Learning Vision-based Pursuit-Evasion Robot Policies

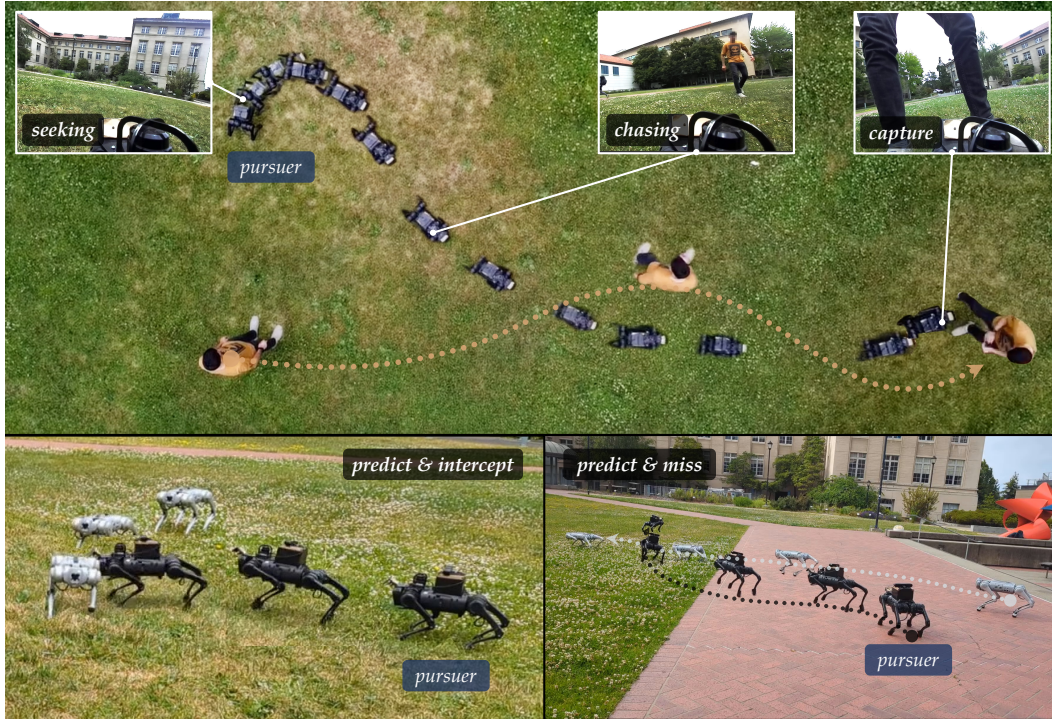**Andrea Bajcsy**[*]   **Antonio Loquercio**[*]   **Ashish Kumar**   **Jitendra Malik**

Figure 1: Our approach deployed in a pursuit-evasion interaction in the wild. Our policy (black robot acting as pursuer) automatically synthesizes behaviors like slowing down and information gathering, accelerating upon detection, and prediction and interception. Video results at https://abajcsy.github.io/vision-based-pursuit/.

Abstract: Learning strategic robot behavior—like that required in pursuit-evasion interactions—under real-world constraints is extremely challenging. It requires exploiting the dynamics of the interaction, and planning through both physical state and latent intent uncertainty. In this paper, we transform this intractable problem into a supervised learning problem, where a fully-observable robot policy generates supervision for a partially-observable one. We find that the quality of the supervision signal for the partially-observable pursuer policy depends on two key factors: the balance of diversity and optimality of the evader's behavior, and the strength of the modeling assumptions in the fully-observable policy. We deploy our policy on a physical quadruped robot with an RGB-D camera on pursuit-evasion interactions in the wild. Despite all the challenges, the sensing constraints bring about creativity: the robot is pushed to gather information when uncertain, predict intent from noisy measurements, and anticipate in order to intercept.

Keywords: Multi-Agent Interaction, Vision-based Robotics

---

[*]denotes equal contribution. All authors are affiliated to UC Berkeley.

1 Introduction

Robot learning has accelerated progress for embodied agents acting "in the wild": quadrupedal and wheeled robots navigate through hard-to-model terrains [1, 2, 3, 4, 5], quadrotors fly at their limits [6], and robotic arms deftly manipulate deformable objects [7]. However, these successes are limited to robots acting in isolation; in reality, robots deployed at scale will inevitably interact with other agents, like people or other robots.

In-the-wild multi-agent interactions raise significant challenges: not only does a robot have to account for perception-induced uncertainty of the physical state (e.g., ego state, positions of others), but it must also account for uncertainty in other agents' future behavior. This problem setting is traditionally modeled by decentralized partially-observable Markov decision processes (Dec-POMDPs) or partially-observable stochastic games (POSGs). While in theory, solutions to these formulations would automatically yield desirable behaviors like information gathering when uncertain, in practice they are notoriously intractable.

Nevertheless, human and animal behavior exhibits these abilities [8]. Pursuit-evasion interactions are a canonical example: the pursuer gathers information about the hidden evader by turning and scanning the environment; upon detection, the pursuer has to continuously strategize about its next move without perfect knowledge of how the evader will react, all from onboard sensors. In this work, we take the first steps towards building similar capabilities into autonomous robots.

Our key idea is to leverage a fully-observable policy to generate supervision for a partially-observable one. However, the classic paradigm of privileged learning [9] does not apply naively to this setting. Namely, privileged information depends not only on the robot, but also on the other agent's behavior, which is dictated by *more* than just physics; it is dictated by the other agent's intent. Therefore, we design a learning procedure to first build a low-dimensional latent representation of intent from future evader trajectories and then learn to estimate this representation from a history of pursuer actions and observations.

Through extensive empirical analysis, we find that the quality of the supervision signal depends on a delicate balance between the diversity of the agents' behavior and the optimality of the interaction. In addition, there are many models for generating the fully-observable supervisor policy (e.g., game-theory [10], multi-agent RL [11]), each with their own potential strengths and weaknesses. We discover that fully-observable policies obtained under strong modeling assumptions (e.g., both agents play under perfect-state Nash equilibrium), are less effective at supervising partially observable ones.

Informed by this analysis, we synthesize a policy that *automatically* takes actions to resolve physical state uncertainty (e.g., looking around to see detect where the other agent is) while also generating predictions about other agents' intent to yield strategic behavior. We deploy this policy on a physical legged robot in a pursuit-evasion game, where it interacts with humans or other legged robots (Fig. 1). Note that the robot only uses onboard sensing, e.g., proprioception and an RGB-D camera, to estimate its state and other agents' physical state and intent.

2 Related Work

Dynamic Games & Multi-Agent RL. Dynamic game theory has a long history of modeling strategic interaction between multiple agents [12, 13, 14, 15] and has influenced fields like robust control [16] and reinforcement learning [17, 18, 19]. Both traditional and modern variants of dynamic games have predominantly assumed knowledge of perfect state. While this has been successful in contexts like robustness to physical disturbances [18, 20, 21], it's a limiting assumption for real-world interaction. Partially observable stochastic games provide a mathematical model of strategic interaction under partial observability [22], where all players have only partial information about environment state. However, they are tremendously computationally expensive to solve, and approximations are an active area of research [23]. To make the optimization tractable, multi-agent

reinforcement learning (MARL) algorithms exploit large-scale simulation and neural network representations [24, 18, 25, 26, 27]. Such approaches have achieved impressive results in simulation interactions like hide-and-seek [28], video games like Starcraft [29], diplomacy [30], and board games like Go [31]. However, to-date, multi-agent RL approaches have not yet scaled to embodied systems acting under real-world sensing constraints. Although these approaches cannot be applied to our physical system out-of-the-box, we do take inspiration from this line of work in the design of privileged fully-observable pursuer policy and a highly strategic evader policy.

Latent Intent Modeling. Recent works [32, 33, 34] learn a latent representation of agent intent via reconstructing a dataset of fully-observed states and rewards. This line of works assumes that the latent intent changes only *between* interaction episodes and not during an interaction. To handle intent changes *during* interaction [35] learns an estimator of a human's latent state to predict their immediate next action. These works overwhelmingly assume that the only hidden state in the interaction is the opponent's intent: the physical state of the robot, the opponent's state, and possibly the opponent's action are assumed to be observable. We address the constraints imposed by on-board robot perception, where the evader's physical state, latent intent, and action are all hidden.

Multi-Quadruped Interaction. Progress in low-level control for quadrupedal robots [1, 3, 4] has increased the interest in combining low-level controllers with high-level decision-making [36]. However, multi-agent quadruped interactions have been relatively under-explored. Most relevant is [37] which trains a *centralized* high-level coordination policy with perfect (global) state for two robots pushing a box. [38, 39] present cooperative control of robots via model predictive control and control barrier functions. To the best of our knowledge, our work is the first to demonstrate autonomous interaction between a quadruped and another robotic or human agent truly in the wild.

3 Overview

We seek a robot policy that can strategically interact with another agent in a decentralized fashion (i.e., no explicit communication) and only using proprioception and a single onboard RGB-D camera. Although our technical approach is general, we ground this work in pursuit-evasion games [12], which exhibit core challenges at the heart of real-world multi-agent interaction: partial observability, nonlinear physical dynamics (e.g., quadrupedal dynamics), low-latency decision-making, and a need for strategic planning.

We approach this problem using privileged learning [9]. We found that directly using reinforcement learning to train a policy that reasons *strategically* through *partial observability* was unsuccessful. The key to our approach is to leverage a fully-observable policy to generate supervision for the partially-observable one. During privileged training, we leverage a new type of privileged information: the future state trajectory of the evader. We first learn a **fully-observable policy** (π^*) (top, Fig. 2), that gets access the true future N-step state trajectory of the evader and the current true relative state. This enables π^* to quickly learn actions that account for the evader's behavior by using a learned latent intent, z_t, that encodes the future trajectory of the other agent.

We then distill this policy into a **partially-observable policy** (π^p) which only uses an egocentric video stream from an onboard RGB-D camera (bottom, Fig. 2). Specifically, π^p gets access to a *history* of relative state *estimates and uncertainties* which are generated via a standard Kalman Filter [40]. Even though the Kalman filter is an incredibly coarse approximation of the true system, the uncertainty information captured by the covariance matricies is sufficient for the prediction policy to learn information-gathering behaviors (like turning and looking for the evader), when combined with the teacher policy. In this light, our privileged learning approach can be viewed as an approximation to the optimal policy obtained via solving the underlying, but intractable, decentralized partially-observable Markov decision process (Dec-POMDP).

Our partially-observable policy can be applied zero-shot in the real world using the output of an off-the-shelf object detector [41]. We deploy it in the wild to play a pursuit-evasion game with a human evader and another quadrupedal robot controlled by a human operator.
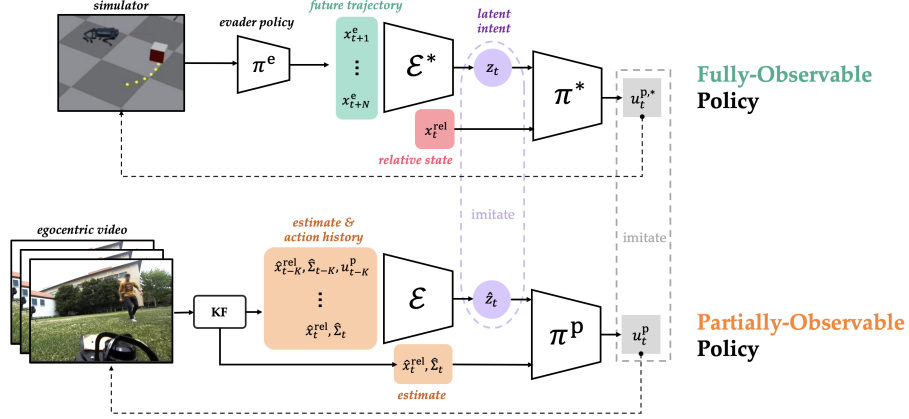
Figure 2: (top) The fully-observable policy knows the true relative state and gets privileged access to the future evader trajectory from which it learns a representation the evader intent. (bottom) The partially-observable policy must plan through physical and latent intent uncertainty.

4 Approach

Given an evader policy π^{e}, we define the pursuer's planning problem as a finite-horizon, discrete-time optimization problem. We seek a policy for the pursuer $\pi^{\mathrm{p}} : \mathcal{O}^{\mathrm{p}} \to \mathcal{U}^{\mathrm{p}}$ which maps from observations to actions that maximizes the following objective:

$$J(\pi^{\mathrm{p}}, \pi^{\mathrm{e}}) = \mathbb{E}_{\tau \sim p(\tau | \pi^{\mathrm{p}}, \pi^{\mathrm{e}})} \Big[\sum_{t=0}^{T} \gamma^t r_t \Big]. \tag{1}$$

Here, $\tau = \{(x_0^{\mathrm{p}}, x_0^{\mathrm{e}}, u_0^{\mathrm{p}}, u_0^{\mathrm{e}}, o_0^{\mathrm{p}}, o_0^{\mathrm{e}}, r_0) \ldots (x_T^{\mathrm{p}}, x_T^{\mathrm{e}}, u_T^{\mathrm{p}}, u_T^{\mathrm{e}}, o_T^{\mathrm{p}}, o_T^{\mathrm{e}}, r_T)\}$ is the joint trajectory of states, actions, observations, and rewards induced by a pair of pursuer and evader policies, drawn from the distribution $p(\tau \mid \pi^{\mathrm{p}}, \pi^{\mathrm{e}})$. The discount factor is denoted by γ. More formally, this optimization defines the solution to a two-agent, finite horizon decentralized partially-observable Markov decision process (Dec-POMDP).

We denote the global physical state of the pursuer as $x^{\mathrm{p}} \in \mathbb{R}^{n_{\mathrm{p}}}$ and the evader to be $x^{\mathrm{e}} \in \mathbb{R}^{n_{\mathrm{e}}}$. Note that the pursuer policy π^{p} does not observe the global state of the agents. The robot's high-level linear and angular velocity commands are denoted by $u^{\mathrm{p}} \in \mathcal{U}^{\mathrm{p}}$ and the pursuer's low-level joint torques are controlled via a pre-computed walking policy. The evader also controls its linear and angular velocity, $u^{\mathrm{e}} \in \mathcal{U}^{\mathrm{e}}$. Both $\mathcal{U}^i, i \in \{\mathrm{p}, \mathrm{e}\}$ are bounded sets, modeling actuation limits. For example, in simulation, the maximum linear speed of the pursuer is 3 m/s, and the evader is 2.5 m/s.

The pursuer is rewarded for minimizing the distance between the two agents at each timestep, and obtains a termination bonus upon capture:

$$\textit{Pursuit: } r_t = -||x_t^{\mathrm{e}} - x_t^{\mathrm{p}}||_2^2, \quad \textit{Capture: } r_T = \alpha \cdot \mathbb{1}\{||x_T^{\mathrm{e}} - x_T^{\mathrm{p}}||_2^2 \leq 0.8\} \tag{2}$$

Asymmetries. Our setting has three asymmetries that induce complexity: 1) *information* (agents have limited FOV and partial state), 2) *dynamics* (e.g., robot quadruped interacting with human), and 3) *control bound* asymmetry (e.g., agents with different maximum speeds).

Ego-Centric State. All agents reason about the *relative* physical state in their own body frame. In a slight abuse of notation, we refer to $x^{\mathrm{rel}} := [p_x^{\mathrm{rel}}, p_y^{\mathrm{rel}}, \theta^{\mathrm{rel}}]^{\top}$ as the true relative planar position and orientation of the exo-centric agent in the ego-centric agent's body frame.

State Estimation. In the wild, the true relative state is not available due to sensing limits. Instead, the pursuer estimates x_t^{rel} from the output of a 3D object detector using the RGB camera [41]. Let $o_t^{\mathrm{p}} \in \mathcal{O}^{\mathrm{p}}$ be the 3D relative position of the evader with respect to the pursuer's camera frame[2]. The pursuer's relative state estimate are the mean and covariance of a Kalman filter:

---

[2]If the evader is out of the field of view, then $o_t^{\mathrm{p}} = \emptyset$ and no measurement update is performed.

$(\hat{x}_t^{\mathrm{rel}}, \hat{\Sigma}_t) = \mathrm{KF}(o_{0:t}^{\mathrm{p}}, u_{0:t}^{\mathrm{p}})$. While a suite of more complex filter designs could be used for even better performance [42], we find that using an unoptimized Kalman filter is sufficient for the robot policy to learn information-gathering behaviors.

Evader Policy. The evader policy is key for enabling the pursuer to learn strategic maneuvers. However, where does the evader policy come from? Datasets of quadrupeds interacting with other agents in the wild do not exist, and simulating human-robot or robot-robot interactions that capture the diversity of the real-world is an ongoing challenge for simulation-based robotics. Instead, we take an investigative approach and study three simulated evader policy models: random motion primitives, multi-agent RL, and dynamic game theory. Across all models, we assume the evader has access to the current true relative state in their own body frame. Details are in Sec. 5.2.

4.1 Fully-Observable Policy: Teacher

To learn the pursuer teacher policy π^*, we must address the challenge that privileged information depends on the evader's behavior which is dictated by their dynamics and intent.

Future Evader Trajectory & Latent State. The fully-observable policy π^* gets access to both the true pursuer relative state, x_t^{rel}, and the future N states of the evader: $x_{t:t+N}^{\mathrm{e}}$. Since the pursuer reasons in a relative coordinate system, the evader trajectory is converted into the pursuer's body frame starting from state at the start of the prediction horizon. This relative state trajectory, $x_{t:t+N}^{\mathrm{rel}} \in \mathbb{R}^{N \times 3}$, is input into an encoder, $\mathcal{E}^*(x_{t:t+N}^{\mathrm{rel}}) = z_t \in \mathbb{R}^8$ which learns a low-dimensional latent representation. Intuitively, z_t should capture low-dimensional information about the evader's near-term behavior: for example, the evader's goal direction, their policy class (e.g., spline coefficients), or control bounds. At each timestep, z_t is re-inferred.

Design. Although this pursuer policy is clearly not deployable in the real world, it enables us to convert the intractable planning problem in Eq. 1 to a Markov Decision Process (MDP), amenable to off-the-shelf RL methods [43]. We use Proximal Policy Optimization [43] for training. The policy π^* and the privileged encoder $\mathcal{E}^*$ are both three-layer MLPs with $[512, 256, 128]$ hidden units.

4.2 Partially-Observable Policy: Student

The partially-observable policy, π^{p}, relies on RGB camera observations $o_t^{\mathrm{p}} \in \mathcal{O}^{\mathrm{p}}$. We use a off-the-shelf 3d object detector [41] to convert from the raw RGB image observable o_t^{p} to an detected relative position and heading in the pursuer's camera frame, $y_t \in \mathbb{R}^3$. We use a Kalman Filter to generate an estimated relative state $\hat{x}_t^{\mathrm{rel}}$ and an associated covariance $\hat{\Sigma}_t$. We use a simplified state transition model $\hat{x}_{t+1}^{\mathrm{rel}} = A\hat{x}_t^{\mathrm{rel}} + Bu_t^{\mathrm{p}}$ which ignores the role of the evader (i.e., $u_t^{\mathrm{e}} \equiv 0$) during the the prediction[3] step (details in Appendix A.5). The history of relative state estimates and pursuer actions are encoded into the estimated lower-dimensional latent intent $\mathcal{E}(\hat{x}_{0:t}^{\mathrm{rel}}, \hat{\Sigma}_{0:t}, u_{0:t-1}^{\mathrm{p}}) = \hat{z}_t$.

Design. We use DAGGER [44] and the fully-observable policy π^* to supervise both the latent intent estimate and the action at each timestep. The policy network is a 3-layer MLP with $[512, 256, 128]$ hidden units, and the encoder $\mathcal{E}$ is a 1-layer LSTM with hidden state 256.

5 Simulation Experiments

We first want to understand the design choices that are important to learn a successful pursuer policy: 1) the ability to learn strategic behavior, 2) the evader policy π^{e} that the robot interacts with at *training* time and 3) the evader interaction at *deployment* time. We perform a set of simulation experiments to ablate the design of the pursuer policy (Sec. 5.1), study the effect of the evader on distillation (Sec. 5.2), and analyze test-time adaptation of the pursuer policy to out-of-distribution opponents (Sec. 5.3). We use Isaac Gym [45] for training and evaluation, and report results over 500 random initial conditions.

---

[3]This removes the need for a velocity estimator which can be hard to design and noisy in reality. While this makes filtering imperfect, we empirically find that the learned policy can compensate for inaccuracies.
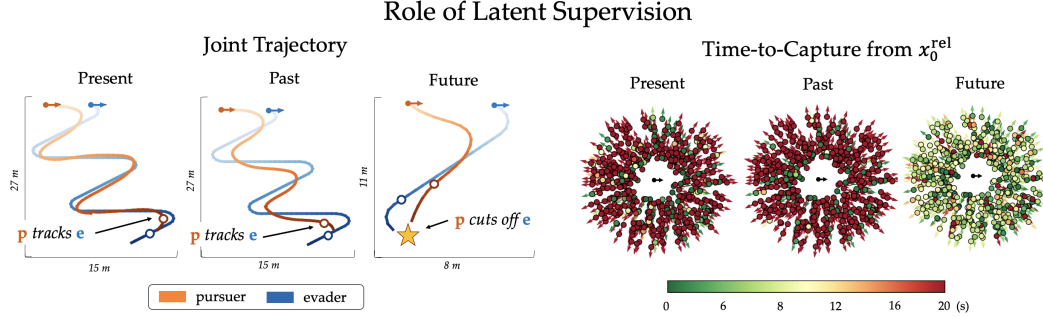
Figure 3: (left) Joint trajectories reveal that using only the present state or directly a history of the past states leads to sub-optimal performance. Supervision from the future enables prediction and fast capture. (right) Time-to-capture as a function x_0^{rel}: future supervision quarters the capture time.

5.1 Predictive representations enable strategic behavior

One of the key types of privileged information we leverage is future trajectory state, which leaks information about the future intent of the other agent. In this section, we ask the question *"What is the value of learning predictive representations for action?"* Here, we fix the evader policy to be highly predictable and investigate alternative approaches to inferring the evader's latent intent. The evader always moves in Dubins' paths [46] with a fixed time duration for turning or going straight determined randomly upon the start of the episode (details in Sec. A.3.1 of the appendix). If the pursuer has a high-quality understanding of the evader's latent intent, it should be able to intercept it along its weaving path. Throughout this section, all policies observe *ground-truth* relative states but not the evader's latent intent.

We consider three approaches: a **reactive** pursuer policy which only observes the present relative state and does not infer any latent intent, a **lookback** policy which must estimate the latent intent from a history of relative states *without supervision from the future* [35, 32] and a **lookahead** policy (ours), which uses a history of relative states and supervision from the future to predict a latent representation of the evader's future trajectory. The **lookback** and **lookahead** policies use identical LSTM architectures for intent estimation.

The **reactive** policy fails to predict the evader's behavior and is unable to do better than tracking the evader and trailing behind it (left, Fig. 3). While adding a history improves the pursuer's strategy (center, Fig. 3), it still struggles to estimate the evader's intent reliably. In contrast, the **lookahead** policy, trained to predict a latent representation of the evader's future trajectory, learns effective predictive behaviors (right, Fig. 3). In addition, the **lookahead** policy converges 10 times faster than the **lookback** one (see Appendix Fig. 10). Overall, our experiments show that using the future trajectory as privileged information favors training and enables strategic behaviors.

5.2 Distillation depends on balance of interaction diversity and optimality

Influence of evader model on pursuer policy. Now that we have a teacher policy architecture, we turn to the role of the evader on the teacher pursuer policy. We compute three fully-observable teacher policies, π^*, trained against three different evader models, π^{e}. With a slight abuse of notation, let x^{rel} be the relative state in the evader's body frame. The **random** evader, $\pi_{\mathrm{rand}}^{\mathrm{e}}(t)$, randomly samples a set of controls to apply each 1-3 seconds. The **multi-agent RL** evader, $\pi_{\mathrm{marl}}^{\mathrm{e}}(x_t^{\mathrm{rel}})$, is trained to evade a pre-trained pursuer policy, equivalent to a single iteration of [18]. Finally, assuming perfect relative state, our setting could be modelled by a zero-sum **game theory** model, whose solution characterizes the optimal pair of policies for the pursuer and the evader [16]. We compute $\pi_{\mathrm{game}}^{\mathrm{e}}(x_t^{\mathrm{rel}})$ via an off-the-shelf dynamic game solver [47]. Details on all evaders in Sec. A.3. Top row in Fig. 4 shows that with perfect state, the pursuer capture time is indistinguishable between **random** and **MARL** evaders, while the optimal **game theory** evader maximally exploits the interaction.
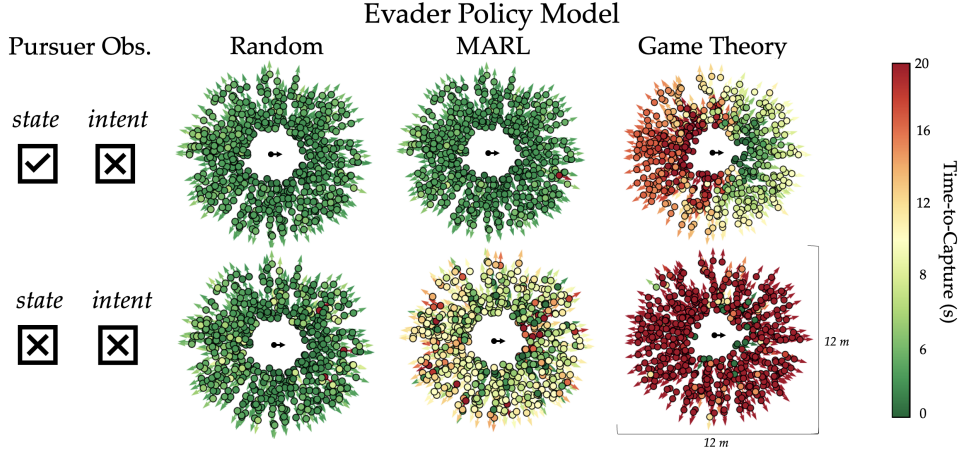
Figure 4: 500 randomly sampled x_0^{rel} initial positions and relative orientation. Colors indicate the normalized time-to-capture starting from the shown initial condition. (top row) Policy knows perfect physical state but infers latent intent from history of relative states and pursuer actions, trained with three different evader models: heuristic, MARL, and zero-sum game-theoretic. (bottom row) Partially-observable state and intent policy is supervised by the corresponding policy above.

Distillation to partially-observable policy. After training the fully-observable pursuer policy against each evader, we supervise the corresponding partially-observable pursuer policy (bottom row, Fig. 4). We find that the **game-theoretic** pursuer policy makes for a poor supervisor because the supervision and interaction data operate under a perfect state-feedback Nash equilibrium assumption that is too hard to satisfy for the partially-observable policy. In contrast, robots trained against noisily-optimal (**MARL**) or extremely diverse (**random**) evaders have smaller in-distribution performance drops. This indicates that the interaction assumptions under which the teacher policy is obtained must be feasible for the partially-observable student.
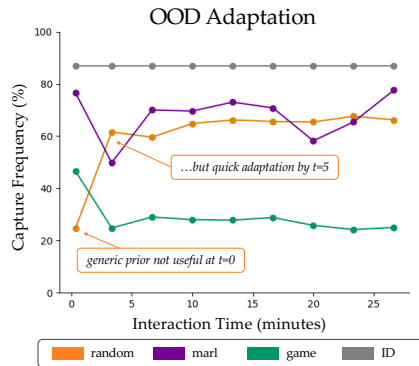

Figure 5: Coverage is more helpful than specialization for quick adaptation.

5.3 For adaptation, coverage is more helpful than specialization

Ultimately, the test-time distribution of the evader is unknown a priori. Thus, we ask *"How quickly and how effectively can different partially-observable pursuer policies adapt to an out of distribution evader?"* We simulate interaction between $\pi_{\text{rand}}^{\text{P}}, \pi_{\text{marl}}^{\text{P}}, \pi_{\text{game}}^{\text{P}}$ and the highly predictable Dubins' agent (Sec. A.3.1). None of the pursuers were trained on this behavior. We collect batches of joint state trajectories, and then finetune the weights of the pursuer's encoder $\mathcal{E}$ by supervising the latent $\hat{z}_t$ at each timestep via the privileged encoder $\mathcal{E}^*$. Since $\pi_{\text{rand}}^{\text{P}}$ has a generic prior on the evader motion, the representation in $\mathcal{E}$ is rich enough to triple its capture frequency in just 5 min. (Fig. 5). $\pi_{\text{marl}}^{\text{P}}$ is good at the start, but, due to its prior on the evader motion, it is less flexible and needs more data to see improvements. $\pi_{\text{game}}^{\text{P}}$, with a stronger prior than $\pi_{\text{marl}}^{\text{P}}$, fails to adapt and reaches a sub-optimal performance with the limited data. This indicates that to quickly adapt to agents "in the wild", which are neither random nor optimal, coverage is more helpful than specialization.

6 Real-World Results

Real-world interactions are out-of-distribution for two main reasons: (1) the behavior of the evader is unscripted and possibly very different to what was observed in simulation, and (2) the physical
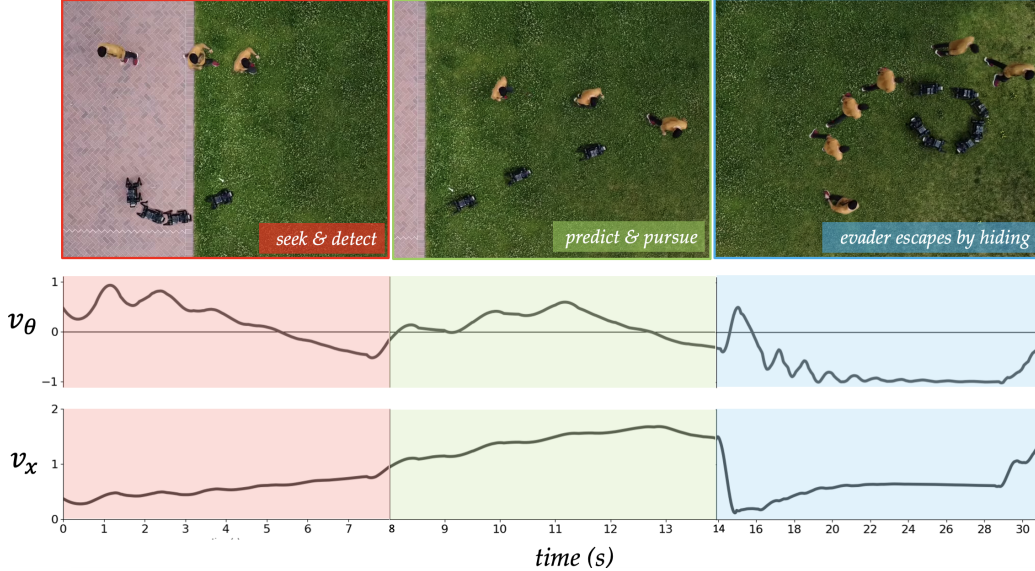
Figure 6: Single-shot interaction between a vision-based pursuer policy and a human. **Left:** Since the human starts outside the FOV of the robot, the latter turns and seeks until it gets the first detection. **Center:** The robot predicts the human will go straight and gallops to where the person will be. **Right:** Human strategically hides outside robot's FOV to escape.

dynamics of the evader do not follow the unicycle model as in simulation. We run two sets of experiments to study how our policies react to such conditions.

First, we ablate the pursuer policy and deploy $\pi^{\text{P}}_{\text{rand}}$, $\pi^{\text{P}}_{\text{marl}}$, $\pi^{\text{P}}_{\text{game}}$ on a physical quadruped robot to interact with a human. We observe that $\pi^{\text{P}}_{\text{rand}}$ and $\pi^{\text{P}}_{\text{marl}}$ perform qualitatively similarly. They showcase information-seeking motions when the evader is not in the field of view and predictive strategies when the evader is visible, i.e., heading towards where the evader *will be*, not where *it is* (Fig. 6). However, $\pi^{\text{P}}_{\text{marl}}$ shows slightly better anticipation and faster reaction times. Conversely, $\pi^{\text{P}}_{\text{game}}$ shows inefficient information-seeking motions, taking long detours to reach the evader. Such performance, inline with the experiments from Sec. 5, confirms that the diversity of interaction data collected by a game-theoretic supervisor is not high enough to be robust to real-world interactions.

Second, we keep the pursuer policy fixed and ablate the dynamics of the evader. Concretely, we compare the performance of $\pi^{\text{P}}_{\text{rand}}$ when interacting against a human or another quadruped teleoperated by a human (Fig. 1). In both cases, we observe aspects of strategic behavior. However, such behavior is more apparent during interaction with another robot. This is due to the robot's dynamics being closer to the unicycle model the policy was trained on in simulation. See website for videos.

7   Discussion

Limitations. Our current approach does not model the affordances of the environment, like obstacles, that could be strategically used by the pursuer. However, to achieve this, the pursuer needs to sense the environment geometry, potentially making optimization much more challenging. Additionally, the limited FOV assumption could be alleviated with different sensor designs (e.g., high-resolution 360 FOV camera), but introduce additional challenges like the computational burden of processing high-resolution images, that are interesting future work.

Conclusions. Our paper takes the first steps toward learning vision-based robot policies that can reason strategically through partially-observable physical state and latent intent. We find interesting pursuer behaviors when deployed on a physical quadruped robot with an RGB-D camera: information gathering under uncertainty, intent prediction from noisy state estimates, and anticipation of agents' motion.

Acknowledgments

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[2] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast traversability estimation for wild visual navigation. *Robotics: Science and Systems*, 2023.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[4] A. Loquercio, A. Kumar, and J. Malik. Learning Visual Locomotion with Cross-Modal Supervision. In *ICRA*, 2023.

[5] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[6] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[7] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *Robotics: Science and Systems*, 2022.

[8] A. M. Wilson, J. Lowe, K. Roskilly, P. E. Hudson, K. Golabek, and J. McNutt. Locomotion dynamics of hunting in wild cheetahs. *Nature*, 498(7453):185–189, 2013.

[9] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[10] T. Başar and G. J. Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[11] S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2022.

[12] R. Isaacs. Differential games i: Introduction. Technical report, RAND CORP SANTA MONICA CA SANTA MONICA, 1954.

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

[14] L. S. Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953.

[15] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[16] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50 (7):947–957, 2005.

[17] K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[18] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[19] L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[20] Y. Tang, J. Tan, and T. Harada. Learning agile locomotion via adversarial training. In *2020 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*, pages 6098–6105. IEEE, 2020.

[21] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. *L4DC*, 2022.

[22] Z. Zhang and J. F. Fisac. Safe occlusion-aware autonomous driving via game-theoretic active perception. *arXiv preprint arXiv:2105.08169*, 2021.

[23] W. Schwarting, A. Pierson, S. Karaman, and D. Rus. Stochastic dynamic games in belief space. *IEEE Transactions on Robotics*, 37(6):2157–2172, 2021.

[24] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[25] M. Woodward, C. Finn, and K. Hausman. Learning to interactively learn and assist. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2535–2543, 2020.

[26] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[27] D.-K. Kim, M. Riemer, M. Liu, J. Foerster, M. Everett, C. Sun, G. Tesauro, and J. P. How. Influencing long-term behavior in multiagent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:18808–18821, 2022.

[28] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent interaction. *Machine Learning, Cornell University*, 2019.

[29] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[30] M. F. A. R. D. T. (FAIR)†, A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

[31] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[32] A. Xie, D. Losey, R. Tolsma, C. Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*, pages 575–588. PMLR, 2021.

[33] S. Parekh, S. Habibian, and D. P. Losey. Rili: Robustly influencing latent intent. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 01–08. IEEE, 2022.

[34] W. Z. Wang, A. Shih, A. Xie, and D. Sadigh. Influencing towards stable multi-agent interactions. In *Conference on robot learning*, pages 1132–1143. PMLR, 2022.

[35] J. Z.-Y. He, Z. Erickson, D. S. Brown, A. Raghunathan, and A. Dragan. Learning representations that enable generalization in assistive tasks. In *Conference on Robot Learning*, pages 2105–2114. PMLR, 2023.

[36] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. *arXiv preprint arXiv:2210.04435*, 2022.

[37] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *Conference on Robot Learning*, 2019.

[38] R. T. Fawcett, L. Amanzadeh, J. Kim, A. D. Ames, and K. A. Hamed. Distributed data-driven predictive control for multi-agent collaborative legged locomotion. *arXiv preprint arXiv:2211.06917*, 2022.

[39] J. Kim, J. Lee, and A. D. Ames. Safety-critical coordination for cooperative legged locomotion via control barrier functions. *arXiv preprint arXiv:2303.13630*, 2023.

[40] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[41] Zed 2 camera. https://www.stereolabs.com/zed-2/. Accessed: 2023-05-08.

[42] M. A. Lee, B. Yi, R. Martín-Martín, S. Savarese, and J. Bohg. Multimodal sensor fusion with differentiable filters. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10444–10451. IEEE, 2020.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[44] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[45] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[46] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79 (3):497–516, 1957.

[47] I. M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35:300–329, 2008.

A Appendix

A.1 Pursuer capture time as a function of train vs. test evader
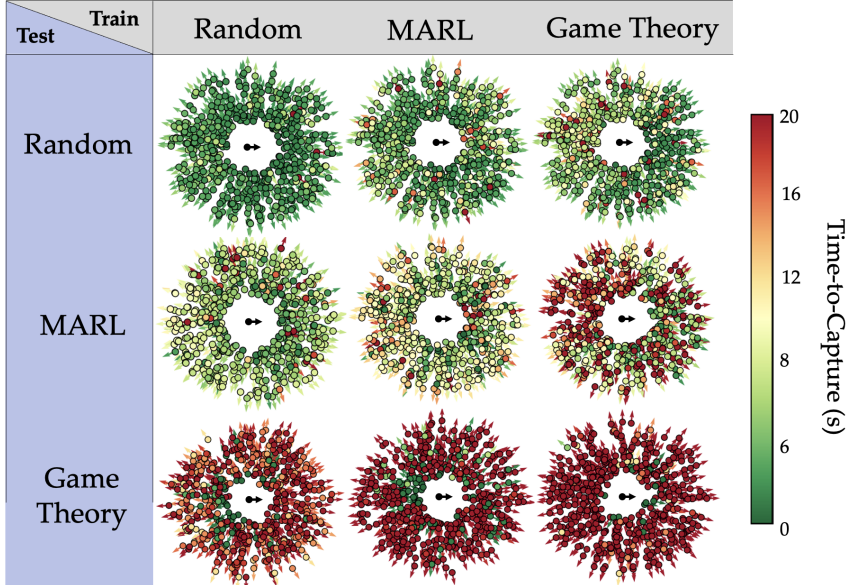

Figure 7: Each pursuer policy is trained on a unique distribution of evader behavior—random, MARL, or zero-sum game-theoretic—and deployed to interact with each other possible evader. The plots reveal how pursuer policies trained on more optimal but less diverse behavior "overfit" to those types of maneuvers. For example, consider the top-right where the pursuer policy is trained against a game theory evader but tested against the random evader. The random evader does not explicitly strategize, but the capture time distribution of the game-theory policy mirrors the distribution when interacting with the game-theoretic evader (Fig. 4).

A.2 Simulation & training details

Quadruped High & Low-level Control. The quadruped uses a pre-computed low-level controller for walking, and is just learning a high-level strategic policy. In this way, our method is agnostic to the specific low-level controller. We use an off-the-shelf RL-based approach to train a walking policy [45]. We use this policy during all simulation experiments and during training the high-level strategic policy. The high-level policy we present in this work controls linear and angular velocity, $u^{\mathrm{p}} = [v_x^{\mathrm{p}}, v_\theta^R] \in \mathcal{U}^{\mathrm{p}}$. The control inputs are bounded: $v_x^{\mathrm{p}} \in [0,3]\frac{m}{s}$ and $v_\theta^{\mathrm{p}} \in [-2,2]\frac{rad}{s}$. Note that the pursuer has a $\sim 17\%$ linear speed advantage over the evader (detailed in Appendix A.3).

Optimization Details. The episode length for interactions in the Isaac Gym simulator is always $T = 20s$. The high-level quadruped policy runs at $5\,Hz$, and simulates a new control applied every $\delta t = 0.2s$. The low-level quadruped policy runs at $50\,Hz$. We use the Adam optimizer for both PPO and for DAGGER. We use Exponential Linear Unit (ELU) activations for the policy network and for the encoders. The future state trajectory of the evader is always $N = 4s = 8$ steps. We train the teacher policies with $3,000$ environments, and the student policy with $8,000$ environments. We always train the high level policy on flat terrain in an infinite plane. Otherwise, we use all the default parameters from [45] but remove the cross-entropy loss during PPO.

12

A.3 Evader simulation & policy design

We simulate the evader as a unicycle dynamical system, controlling linear and angular velocity, $u^{\mathrm{e}} = [v_x^{\mathrm{e}}, v_\theta^{\mathrm{e}}] \in \mathcal{U}^{\mathrm{e}}$. The control inputs are bounded: $v_x^{\mathrm{e}} \in [0, 2.5]\frac{m}{s}$ and $v_\theta^{\mathrm{e}} \in [-2, 2]\frac{rad}{s}$. The discrete-time dynamics model we use is:

$$
\begin{bmatrix} p_x^{\mathrm{e}} \\ p_y^{\mathrm{e}} \\ p_z^{\mathrm{e}} \\ \theta^{\mathrm{e}} \end{bmatrix}^{t+\delta t} = \begin{bmatrix} p_x^{\mathrm{e}} \\ p_y^{\mathrm{e}} \\ p_z^{\mathrm{e}} \\ \theta^{\mathrm{e}} \end{bmatrix}^{t} + \delta t \begin{bmatrix} v_x^{\mathrm{e}} \cos(\theta^{\mathrm{e}}) \\ v_x^{\mathrm{e}} \cos(\theta^{\mathrm{e}}) \\ 0 \\ v_\theta^{\mathrm{e}} \end{bmatrix}. \tag{3}
$$

The evader's initial condition is randomly initialized at the start of each episode and after each environment reset. Specifically, the initial xy-position of the evader is offset from the pursuer initial xy-position via

$$
\begin{bmatrix} p_x^{\mathrm{e}} \\ p_y^{\mathrm{e}} \\ p_z^{\mathrm{e}} \\ \theta^{\mathrm{e}} \end{bmatrix}^{0} = \begin{bmatrix} p_x^{\mathrm{p}} \\ p_y^{\mathrm{p}} \\ 0 \\ 0 \end{bmatrix}^{0} + \begin{bmatrix} r^0 \cos(\psi^0) \\ r^0 \sin(\psi^0) \\ 0.001 \\ \tan^{-1}(p_y^{\mathrm{e},0}/p_x^{\mathrm{e},0}) \end{bmatrix},
$$

where $r^0 \sim \mathrm{Unif}[2, 6]$ is drawn uniformly at random between 2 and 6 meters, and $\psi^0 \sim \mathrm{Unif}[-\pi, \pi]$ so that the evader spawns at random angles relative to the pursuer. The z-position is always fixed at 0.001 and the yaw angle θ^{e} is chosen so the evader is facing away from the pursuer at the start of the episode.

A.3.1 Dubins' policy

This evader always moves in Dubins' paths [46] with a fixed time duration for turning or going straight determined randomly upon the start of the episode. Let $v_x^{\mathrm{e}} \in [\underline{v}_x, \bar{v}_x]$ be the min and max linear velocity commands and $v_\theta^{\mathrm{e}} \in [\underline{v}_\theta, \bar{v}_\theta]$ be the min and max angular velocity commands. Let τ_{fwd} be the number of seconds for which the evader goes forward, and τ_{turn} be the number of seconds to turn. At the start of the episode, each parameter is sampled from the corresponding uniform distribution:

$$
\tau_{fwd} \sim \mathrm{Unif}[2, 4], \quad \tau_{turn} \sim \mathrm{Unif}[0.6, 1.4].
$$

The Dubins' policy switches between two maneuvers depending on if the current time t matches the time interval τ_{fwd} or τ_{turn}:

- τ_{fwd}: evader goes forward at max linear velocity, applying $u^{\mathrm{e}} = [\bar{v}_x, 0]$
- τ_{turn}: evader turns while moving at max linear velocity. With 70% probability the evader turns the *opposite* direction as the last turn, and with 30% turning the *same* direction as last time): $u^{\mathrm{e}} = [\bar{v}_x, \bar{v}_\theta]$ or $u^{\mathrm{e}} = [\bar{v}_x, \underline{v}_\theta]$

At the start of the episode, the evader uniformly at random chooses to turn left or right.

A.3.2 Random policy

This evader, $\pi_{\mathrm{rand}}^{\mathrm{e}}(t)$, randomly samples a motion primitive to apply for a fixed number of seconds, before re-sampling another motion primitive. The set of motion primitives $\mathcal{MP}$ are the Cartesian product of regularly discretized linear and angular velocities:

$$
V_x = [\underline{v}_x, 0.5\underline{v}_x, 0, 0.5\bar{v}_x, \bar{v}_x], \quad V_\theta = [\underline{v}_\theta, 0.5\underline{v}_\theta, 0, 0.5\bar{v}_\theta, \bar{v}_\theta], \quad \mathcal{MP} := V_x \times V_\theta
$$

At the start of each episode, the duration for applying a motion primitive is randomly sampled to be between $1 - 3$ seconds. When it is time to switch motion primitives, $\pi_{\mathrm{rand}}^{\mathrm{e}}(t) \equiv (v_x, v_\theta) \sim \mathrm{Unif}[\mathcal{MP}]$.

13

A.3.3 MARL policy

This evader, $\pi^{\mathrm{e}}_{\mathrm{marl}}(x^{\mathrm{rel}}_t)$, is trained to evade a pre-trained pursuer policy, equivalent to a single iteration of [18]. The evader is rewarded for maximizing the distance between the two agents at each timestep, and obtains a termination penalty upon capture:

$$\textit{Evasion: } r_t = 2 \cdot ||x^{\mathrm{e}}_t - x^{\mathrm{p}}_t||^2_2, \quad \textit{Capture: } r_T = -80 \cdot \mathbb{1}\{||x^{\mathrm{e}}_T - x^{\mathrm{p}}_T||^2_2 \le 0.8\}. \tag{4}$$

The evader trains against a pre-trained, fully-observable pursuer policy: $\pi^*(x^{\mathrm{rel}}, z_t)$. This pursuer was originally trained against a random evader, $\pi^{\mathrm{e}}_{\mathrm{rand}}(t)$. Since the pursuer is pre-trained, its policy is already highly capable of capturing the evader making it difficult for the MARL agent to learn. To tackle this, we use a curriculum set at where at each fixed iteration, the pursuer speed in increased by 20%. The curriculum is set to $[1800, 2000, 2400, 2800, 3100, 3800]$ iterations.
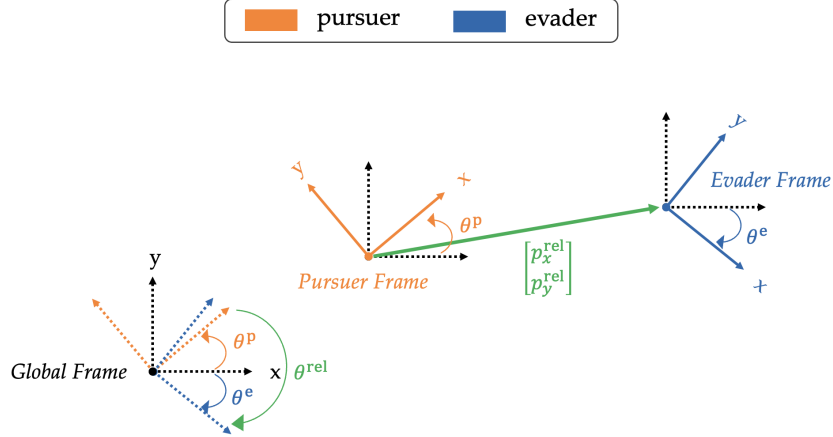

Figure 8: Global and relative coordinate system with respect to the pursuer's coordinate frame.

A.4 Zero-sum game policies: pursuer and evader

For one of our candidate fully-observable supervisor policies, we use an off-the-shelf zero sum differential game solver [16, 47]. It computes the optimal value of the two-player game, $V(x^{\mathrm{rel}}, t)$ at each time $t \in [0, T]$ via solving the Hamilton-Jacobi Isaacs Partial Differential Equation, whose solution yields optimal game-theoretic policies for both the pursuer and the evader. Each agents' state is their global position and their heading angle: $x^i := [p^i_x, p^i_y, \theta^i]^\top$, $\quad i \in \{\mathrm{p}, \mathrm{e}\}$ (see Fig. 8). Each agent controls their linear and angular yaw velocity: $u^i := [v^i_x, v^i_\theta] \in \mathcal{U}^i$ and the dynamics evolve via a unicycle model:

$$\begin{bmatrix} \dot{p}^i_x \\ \dot{p}^i_y \\ \dot{\theta}^i \end{bmatrix} = \begin{bmatrix} v^i_x \cos(\theta^i) \\ v^i_x \sin(\theta^i) \\ v^i_\theta \end{bmatrix}. \tag{5}$$

Each agent's policy is defined in their respective relative coordinate system (see Fig. 8 for relative coordinate system in pursuer's body frame). From the perspective of the pursuer, the relative state $x^{\mathrm{rel}} := [p^{\mathrm{rel}}_x, p^{\mathrm{rel}}_y, \theta^{\mathrm{rel}}]^\top$ consists of the relative xy-position and heading:

$$\theta^{\mathrm{rel}} := \theta^{\mathrm{e}} - \theta^{\mathrm{p}}, \quad \begin{bmatrix} p^{\mathrm{rel}}_x \\ p^{\mathrm{rel}}_y \end{bmatrix} := \begin{bmatrix} \cos(\theta^{\mathrm{p}}) & \sin(\theta^{\mathrm{p}}) \\ -\sin(\theta^{\mathrm{p}}) & \cos(\theta^{\mathrm{p}}) \end{bmatrix} \begin{bmatrix} p^{\mathrm{e}}_x - p^{\mathrm{p}}_x \\ p^{\mathrm{e}}_y - p^{\mathrm{p}}_y \end{bmatrix} \tag{6}$$

The continuous-time relative dynamics of the evader in the pursuer's body frame is defined by:

$$\begin{bmatrix} \dot{p}^{\mathrm{rel}}_x \\ \dot{p}^{\mathrm{rel}}_y \\ \dot{\theta}^{\mathrm{rel}} \end{bmatrix} = \begin{bmatrix} -v^{\mathrm{p}}_x + v^{\mathrm{e}}_x \cos(\theta^{\mathrm{rel}}) + v^{\mathrm{p}}_\theta p^{\mathrm{rel}}_y \\ v^{\mathrm{e}}_x \sin(\theta^{\mathrm{rel}}) - v^{\mathrm{p}}_\theta p^{\mathrm{rel}}_x \\ v^{\mathrm{e}}_\theta - v^{\mathrm{p}}_\theta \end{bmatrix} := f(x^{\mathrm{rel}}, u^{\mathrm{p}}, u^{\mathrm{e}}). \tag{7}$$

Note that when solving the game, the dynamics models are assumed to be deterministic.

The game is solved over the same time horizon as the episode length in the Isaac Gym simulator: $T = 20s$. The pursuer's optimal policy is defined as:

$$\pi^{\mathrm{p}}_{\mathrm{game}}(x^{\mathrm{rel}}, t) := \arg \min_{u^{\mathrm{p}} \in \mathcal{U}^{\mathrm{p}}} \max_{u^{\mathrm{e}} \in \mathcal{U}^{\mathrm{e}}} \nabla V(x^{\mathrm{rel}}, t)^{\top} f(x^{\mathrm{rel}}, u^{\mathrm{p}}, u^{\mathrm{e}}), \tag{8}$$

The evader's optimal policy is defined similarly with two changes: 1) the relative state x^{rel} is of the pursuer's state in the evader's body frame, and 2) the min and the max are swapped in Eq. 8.

$$\pi^{\mathrm{e}}_{\mathrm{game}}(x^{\mathrm{rel}}, t) := \arg \max_{u^{\mathrm{e}} \in \mathcal{U}^{\mathrm{e}}} \min_{u^{\mathrm{p}} \in \mathcal{U}^{\mathrm{p}}} \nabla V(x^{\mathrm{rel}}, t)^{\top} f(x^{\mathrm{rel}}, u^{\mathrm{p}}, u^{\mathrm{e}}). \tag{9}$$

Note that because the joint dynamics are affine in the pursuer and evader controls, the value of the game from the pursuer and evader's perspective is identical (also known as *Isaacs' condition* [13]).

Supervising the partially-observable policy. Recall that our approach uses supervision from the fully-observable policy. The partially-observable policy's intent estimate, $\hat{z}_t$, is supervised with the teacher's intent encoding z_t and also the pursuer actions, u^{p}_t, are supervised with the teacher policy actions, $u^{\mathrm{p},*}_t$. However, the game-theoretic policy does not generate an explicit latent state z_t the way that the MARL and Random policy architectures do. Thus, we only supervise partially observable policy's with the game theory policy $\pi^{\mathrm{p},*}(x^{\mathrm{rel}}_t, t)$ at each timestep and we update the latent intent encoder $\mathcal{E}$ and the policy π^{p} weight's. In the continual learning of Sec. 5.3 we follow this same paradigm: finetune the weights of the encoder and the policy from just action supervision. Otherwise the architecture of the partially observable student policy is identical (LSTM for encoding with 3-layer MLP for action network).

A.5 Kalman Filter

We use a linear state transition model,

$$x^{\mathrm{rel}}_{t+\delta t} = A x^{\mathrm{rel}}_t + B u^{\mathrm{p}}_t + w$$

which ignores the role of the evader (i.e., $u^{\mathrm{e}}_t \equiv 0$) during the the prediction step, and a linear observation model,

$$y_t = H x^{\mathrm{rel}}_t + v.$$

Note that while the Kalman filter ignores the motion of the evader, the student policy does not. Indeed, the student policy uses a history of relative states to predict the future evader's motion (Fig. 2). We intentionally ignore the evader's motion in the Kalman filter prediction step to waive the requirement for a velocity estimator (which can be cumbersome to obtain and noisy in the real world). While this makes filtering imperfect, we empirically found that the learned policy can cope with the resulting inaccuracies.

The state transition model assumes the relative state evolves via a single integrator model, assuming $u^{\mathrm{p}} := [v_x, v_y, v_\theta]$, which directly influence the time derivatives of the relative x, y and θ states. The quadruped we used in the experiments cannot control v_y directly, so at deployment time this entry of the control input is set to zero. The process covariance $w \sim \mathcal{N}(0, Q)$, the measurement covariance $v \sim \mathcal{N}(0, R)$, initial error covariance matrix P_0, observation matrix H, and state transition matricies are:

$$Q = 0.01 \cdot I_{3\times3}, \quad R = \mathrm{diag}(0.2, 0.2, 0.1), \quad P_0 = I_{3\times3}, \quad H = I_{3\times3}, \tag{10}$$

$$A = I_{3\times3}, \quad B = \mathrm{diag}(-\delta t, -\delta t, \delta t), \quad \delta t = 0.2 \tag{11}$$
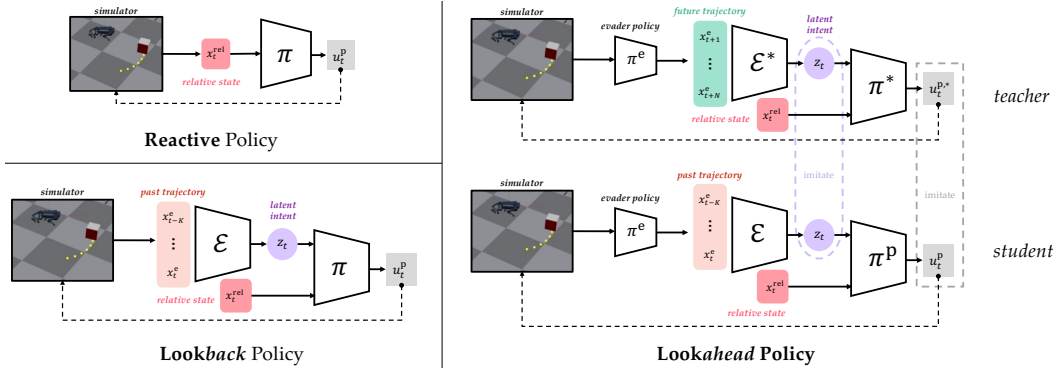
A.6 Real-world experiments

Figure 9: Policy architectures for reactive (up left), lookback (low left) and lookahead policy (right).

We use an Unitree A1 robot and a Unitree Go1 robot for experiments. The policy controls the linear speed of the robot on the x-axis in the body frame, i.e., in the forward direction (v_x), and the angular yaw velocity (v_θ). Such high-level commands are executed by an off-the-shelf low-level policy [4]. We use a Zed 2 camera for object detection in 3D. The object detection pipeline and the Kalman filter run onboard in a Jetson Tx2, while the low-level controller runs on an Intel UP Board. Communication happens through ROS using the implementation of [4].

Our policy only controls the pursuer Unitree A1 robot, while the evader is either a human or a Go1 robot teleoperated by a human. In both cases, the evader policy is neither pre-scripted nor explicitly strategic. The evader tries to avoid the pursuer with short-horizon evasion maneuvers, but it is not strategic in a long horizon to keep the interaction spatially constrained. We switch between tracking a person and another robot by changing the tracking class from PERSON to ANIMAL in the ZED 2 camera configs. We assume that only a single evader is present in the scene; therefore, we do not keep track of object identity. We empirically find that the ANIMAL class has more mis-detections than the PERSON class since the visual features of the quadruped differ from a real dog. Still, the detection performance was sufficient for continuous interaction.

A.7 Section 5.1 details: Reactive, Lookback, Lookahead policies

All policies use the same simulation setup as in Sec. A.2. The main differences are in the inputs and in the latent intent learning. Fig. 9 visualizes the policy architectures for the **reactive**, **lookback** and **lookahead** policies. We also compare training time of our **lookahead** approach–which uses future trajectory information as privileged info to learn the latent intent–to a **lookback** policy which must estimate the latent intent only from a history of relative states via an LSTM. We see that future trajectory supervision enables 10 times faster training compared to the **lookback** policy (Fig. 10) as well as higher overall reward.
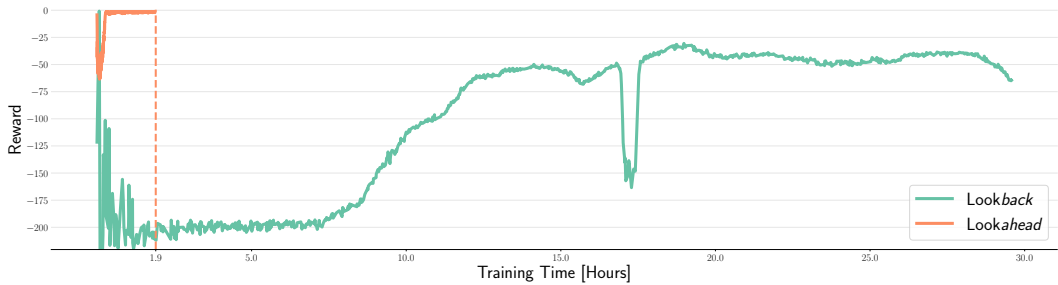

Figure 10: The lookahead policy (orange) converges to a higher reward 10 times faster than the lookback policy (green)

16

A.8 Note on directly learning a strategic pursuer policy with PPO

We tried to directly learn a policy with partial observations, $\pi(\hat{x}_t, \Sigma_t)$, using PPO. However, our policies never got a performance better than random. We believe the main reason to be the difficulty of the optimization problem. Indeed, such a policy needs to learn at the same time: (1) strategy, i.e., learning to predict where the evader will be, (2) control, i.e., learning to move towards a desired location on the x-y plane, and (3) estimation, i.e., learning to reason over the noisy states from the Kalman filter to get an estimate of the real position of the evader. We empirically found that PPO fails to do all these tasks simultaneously. However, we found that PPO could easily solve (1) and (2) if provided with a short-horizon future trajectory of the evader: this corresponds to our fully-observable training. To reduce sample complexity, we learn (3) using distillation learning, even though it could as well be trained with a further stage of reinforcement learning. An interesting venue for future work is to see how to learn such policy directly.