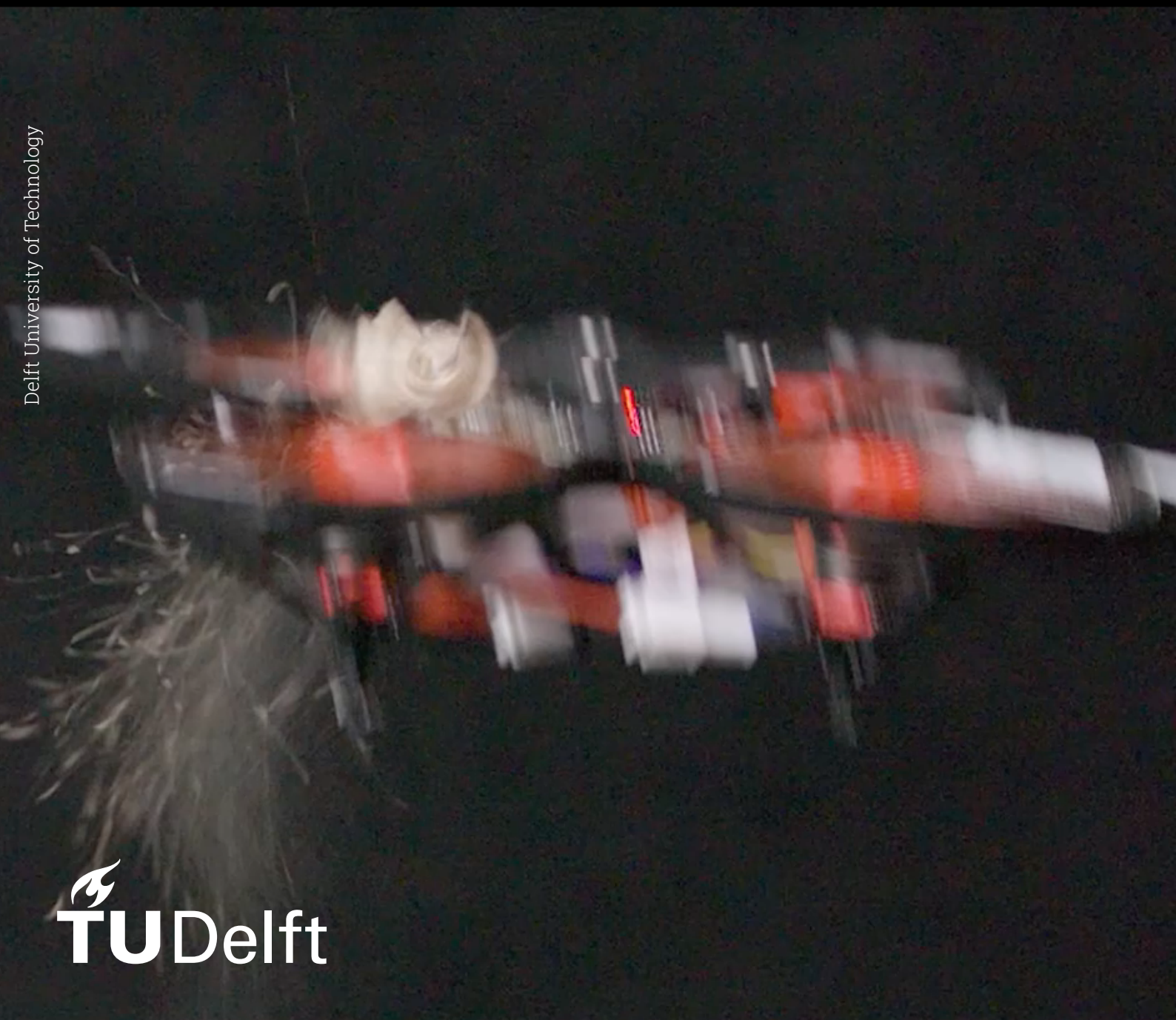
Reinforcement Learning-Based Guidance and Control for Aerial-to-Aerial Pest Interception

MSc Thesis Control & Simulations

Merlijn Broekers



Delft University of Technology

TU Delft

Reinforcement Learning-Based Guidance and Control for Aerial-to-Aerial Pest Interception

Thesis Report

by

Merlijn Broekers

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on December 11, 2025 at 13:00

Thesis Committee:

Chair:	Dr. Ir. C. de Wagter
External Examiner:	Dr. Ir. A. Bombelli
Responsible Supervisor:	Prof. Dr. G.C.H.E. de Croon
Supervisors:	Dr. M. Yedutenko
	Ir. R.W. Vos
Project duration:	February, 2025 - December, 2025
Place:	Faculty of Aerospace Engineering, Delft

TU Delft

Preface

This document marks the pinnacle of my academic achievement. When this project was first presented to me, I was immediately drawn to the breadth of challenges it offered. It seemed like a rare opportunity to test myself across several key domains of aerospace engineering: tackling a demanding control problem, engaging in system design, and ultimately seeing it all come together in a physical system. With the submission of this document, I can also justify to myself that I meet the definition of what it means to be an engineer: *a person who designs and builds*.

I feel extremely privileged to have the network of people around me who have enabled me to carry out this project. This work builds upon that of Reinier Vos, who has been a remarkable and supportive mentor. Together with Guido de Croon and Matthew Yedutenko, I could not have asked for a better team. Thank you for your support, insights, and for making this project extremely enjoyable. I would also like to give special thanks to Raoul Mink for his help in implementing my system on the PATS-X platform; without his support, this would not have been possible.

On a more personal note, I would like to thank my family and friends, who have been here with me throughout my student years. Your ongoing support, companionship, and, above all, your belief in my ability to succeed have been the cornerstone of my journey thus far.

Thank you for reading this work.

Merlijn Broekers
Rotterdam, November 2025

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AIPN	Augmented Integrated Proportional Navigation
ARL	Adversarial Reinforcement Learning
CTCE	Centralized Training with Centralized Execution
CTDE	Centralized Training with Decentralized Execution
DDPG	Deep Deterministic Policy Gradient
DR	Domain Randomization
DRL	Deep Reinforcement Learning
FRPN	Fast Response Proportional Navigation
GAIPN	Global Augmented Integrated Proportional Navigation
GIPN	Global Integrated Proportional Navigation
GPPN	Global Pure Proportional Navigation
GRTPN	Global Robust True Proportional Navigation
GTPN	Global True Proportional Navigation
IPN	Integrated Proportional Navigation
LPN	Linearized Proportional Navigation
MARL	Multi-Agent Reinforcement Learning
MAV	Micro Air Vehicle
PN	Proportional Navigation
POMDP	Partially Observable Markov Decision Process
PPN	Pure Proportional Navigation
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RTPN	Robust True Proportional Navigation
UAV	Unmanned Aerial Vehicle

Symbols

Symbol	Definition	Unit
$\mathbf{r}$	Relative position vector between pursuer and evader	[m]
$\|\mathbf{r}\|$	Relative distance between pursuer and evader	[m]
$\|\dot{\mathbf{r}}\|$	Closing speed	[m/s]
$\dot{\mathbf{r}}$	Relative velocity vector between pursuer and evader	[m/s]
$\ddot{\mathbf{r}}$	Relative acceleration vector between pursuer and evader	[m/s^2]
$\mathbf{r}_p$	Position vector of the pursuer	[m]
$\mathbf{r}_e$	Position vector of the evader	[m]
$\mathbf{v}_p$	Velocity vector of the pursuer	[m/s]
$\mathbf{I}_r$	Unit vector along $\mathbf{r}$	–
$\dot{\phi}$	LOS angular rate	[rad/s]
$\dot{\boldsymbol{\phi}}$	Angular velocity of the LOS vector	[rad/s]
$\ddot{\boldsymbol{\phi}}$	Angular acceleration of the LOS vector	[rad/s^2]
θ, ϕ, ψ	Euler angles representing deviation of pursuer orientation from LOS frame (pitch, roll, yaw)	[rad]
λ	Navigational constant in PN guidance laws	–
t_{go}	Estimated time-to-go until interception	[s]
$\mathbf{a}_p$	Commanded acceleration of the pursuer	[m/s^2]
$\mathbf{a}_{\text{ff}}$	Feedforward acceleration	[m/s^2]
$R_{\text{rot}}(\theta, \phi, \psi)$	Rotation matrix representing deviation from desired alignment using Euler angles	–
$\Lambda(\theta, \phi, \psi)$	Cost function penalizing deviation from ideal interception alignment	–
Γ	CATD-based cost function measuring control aggressiveness-to-deviation tradeoff	–
J	Cumulative reward or performance index	–
π	Policy	–
s	State variable	–
a	Action variable	–
Q	Action-value function	–
γ	Discount factor	–
f	System dynamics function in real environment	–
f'	Approximated dynamics function in simulation	–
g	Observation function in the real system	–
g'	Observation function in the simulator	–
h	Reward function in the real system	–
h'	Reward function in the simulation environment	–
ϕ	Environment mapping operator (real-to-sim)	–
$\widetilde{E}'$	Corrupted simulator	–
ξ', ψ'	Parameters used to randomize physical and sensory properties in simulation	–

1

Introduction

Political pressure on climate change and growing consumer awareness are reshaping agriculture [1]. In this context, pesticide use faces renewed scrutiny for long-term sustainability. Yet pesticides remain a practical necessity. Without pesticide intervention, potential global crop losses are estimated at roughly 50% for wheat and over 80% for cotton [2]. Farmers therefore face a difficult trade-off, balancing ecological impact against economic viability.

Micro air vehicles (MAVs) offer a promising route to reduce chemical inputs. PATS, a Delft-based agri-tech start-up [3], is developing systems for autonomous greenhouses. Its PATS-X concept uses an infrared depth camera to detect and track airborne pests, then dispatches an autonomous MAV to physically intercept and neutralize them, reducing reliance on pesticides. In operation the camera provides real-time target position and velocity, a guidance module computes interception commands, and the MAV executes those commands until contact.

Designing guidance and control for PATS-X is challenging. Targets are slow in top speed but highly agile and often reactive, demanding rapid retargeting and precise proximity control. Greenhouse environments add clutter, tight margins, and field-of-view constraints that limit feasible maneuvers. Sensing introduces noise and delay, and modeling errors create a simulation-to-reality gap that can erode expected performance. Together, these factors make reliable aerial-to-aerial interception a demanding control problem.

1.1. Research Formulation

This thesis contributes to the PATS-X system by evaluating and improving guidance and control for MAV based pest interception. State-of-the-art classical controllers are compared to newly developed reinforcement learned policies trained on insect flight recordings. The study aims to identify best practices for aerial-to-aerial pest interception, and places special emphasis on methods that support deployment on real hardware. The overarching objective guiding this research is,

Research Objective

To enable reliable, pesticide free elimination of highly maneuverable airborne pests in greenhouse environments by developing, benchmarking, and validating both classical and reinforcement learning based guidance and control frameworks for MAVs.

To turn this objective into actionable work, the study is framed by three questions adressed in the scientific article. The research questions are formulated as,

Research Question 1

How do reinforcement learning-based guidance and control policies for MAV pest interception compare to state of the art classical controllers in terms of performance across representative evasion scenarios?

Research Question 2

How do control abstraction level and observation design influence the performance of reinforcement learning-based MAV pest interception policies?

Research Question 3

How do domain randomization and action smoothing influence the zero shot sim-to-real transferability of reinforcement learning based control policies for MAV pest interception?

1.2. Structure of the Report

This report consists of three parts. First, in Part I, the scientific article is presented that addresses the main research questions above. The paper is organized into eight sections. Most importantly, Section III lays out the aerial-to-aerial interception problem formulation. Section IV explains how experiments were conducted and details hardware deployment. Section V presents the main findings relevant to the research questions. Section VI presents a novel PATS-X implementation prototype. Finally, Section VII discusses limitations and Section VIII concludes.

Second, in Part II, the literature review is motivated by the different high-level architectural components of the PATS-X system. The review opens with chapter 4, which formalizes MAV–pest interception kinematics. Then chapter 5 reassesses proportional-navigation variants under MAV constraints. In chapter 6 reinforcement learning and key design choices are motivated. Finally, chapter 7 treats sim-to-real transfer, and chapter 8 synthesizes gaps into research questions and a central objective.

To end, Part III concludes the research presented in this report. Additional results that did not fit directly within the scope of this report are also included here.

Part I

Scientific Article

Reinforcement Learning-Based Guidance and Control for Aerial-to-Aerial Pest Interception

M.S. Broekers, R.W. Vos[1], M. Yedutenko[1], G.C.H.E. de Croon[1]

***Abstract*— PATS-X is a greenhouse pest suppression system that uses a depth camera with an autonomous micro air vehicle (MAV) to detect, track, and physically intercept flying insects. This study targets guidance and control for reliable aerial-to-aerial interception. Reinforcement learning (RL) is used to learn policies from insect flight recordings. We evaluate control policies at increasing levels of abstraction: direct motor commands, collective thrust and body rates (CTBR), and acceleration. In simulation, lower abstraction levels yield better interception performance; moving from acceleration to motor command reduces the median time to first interception by about 41%. A systematic variation of the observation space reveals that the most effective observations are body frame relative position and velocity, and short temporal histories add no benefit beyond noise filtering. Compared with a state-of-the-art classical benchmark, Fast Response Proportional Navigation (FRPN), the best motor level RL policy in simulation achieves a median first interception time of 0.85 [0.76–1.07]s with 99.1% interception rate, compared with FRPN at 1.90 [1.04–2.80]s and 95.6%. To address the reality gap, we compare how well the different control abstractions transfer to hardware. CTBR policies deploy on hardware with the least performance loss relative to simulation. Motor-level policies also transfer when trained with modest domain randomization (DR) plus an action-difference penalty that limits command jitter and thermal load. Acceleration-level policies did not transfer. In a PATS-X proof of concept, an RL controller deployed on the actual system reached a 95.6% interception rate of virtual moths versus 80.0% for the existing controller. Moreover, the RL controller shortened time-to-first-interception by 0.70s, indicating the potential of RL-based guidance for the PATS-X system.**

I. INTRODUCTION

Climate change drives the demand for sustainable agriculture, pushing growers toward technologies that reduce ecological impact while maintaining productivity [1]. Micro air vehicles (MAVs) are a promising part of this transition: by enabling targeted monitoring and intervention, MAV systems have been shown to detect stress and disease earlier, and improve resource efficiency in protected cropping systems [2]. In line with this development, PATS, a Delft based agri-tech start-up is developing the PATS-X concept. This is an integrated greenhouse solution that uses an infrared depth camera to detect and track flying pests. After detection it dispatches an autonomous MAV to physically intercept and neutralize the pests, thereby reducing reliance on chemical pesticides [3].

[1]All three researchers are with Delft University of Technology, Faculty of Aerospace Engineering, Department of Control & Simulation, Micro-Air-Vehicle Laboratory (MAVLab). This scientific article is part of the thesis report for fulfillment of a MSc. in Aerospace Engineering by Merlijn Broekers defended on December 11, 2025

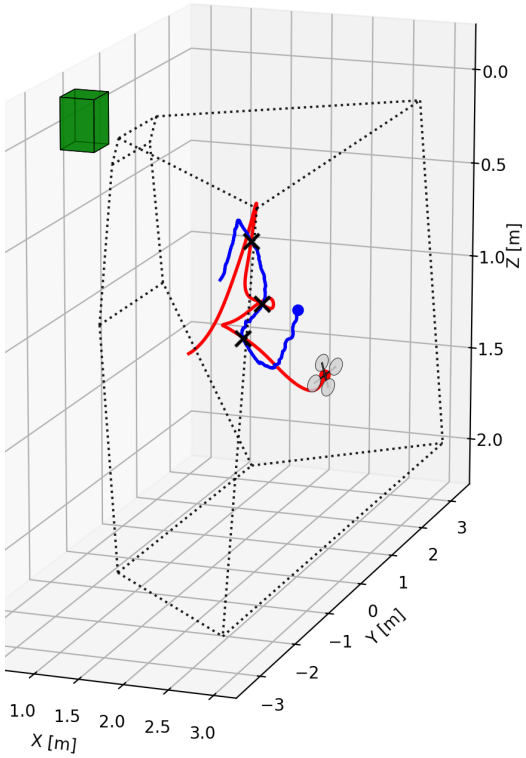


Fig. 1. The figure shows a 3D interception chase. The drone (red) and the evader (blue) are represented by their flight trajectories, with short trailing paths that indicate their positions over the last one second. The evader's path is a replay from a real recorded moth flight. Interception points are marked with a black cross. The viewing frustum of the PATS depth camera (green) is drawn as a black wire-frame box enclosing the region observed.

Intercepting flying pests in this setting is a guidance-and-control problem with tight constraints: targets are small, agile, and erratic; the MAV may be faster in top speed but not in instantaneous maneuverability; the greenhouse imposes hard spatial limits; and camera-based measurements are noisy and delayed. These factors drive the design of the interception controller. This paper therefore benchmarks classical and learned approaches for aerial-to-aerial interception in the PATS-X context. The best practices are identified for the PATS-X system based on interception performance in simulation and transferability to the physical system.

II. RELATED WORK

A. Classical Interception G&C

Classical interception guidance for aerial engagements is dominated by proportional navigation (PN) and its descendant variations. The core working principle of PN is to

minimize miss distance through keeping the line-of-sight (LOS) angle, denoted as ϕ, to the target constant. For interception, the LOS angular rate is driven toward zero while the distance to the target along the LOS decreases, Fig. 2 illustrates this principle.

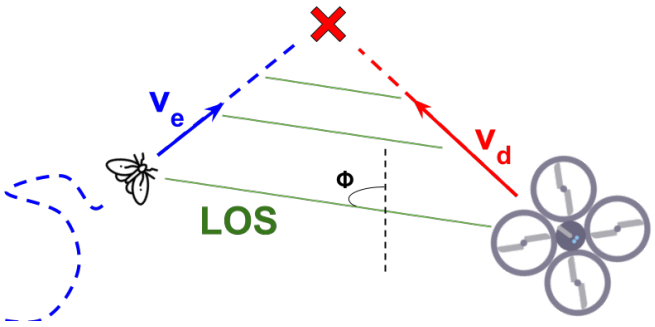


Fig. 2. Working principle of PN. Keeping the LOS angle ϕ constant while decreasing the LOS distance is a sufficient condition for interception.

An attractive quality of PN is that, under certain assumptions, it is optimal with respect to terminal miss distance; therefore, it is a method for precise interception [4]. This emphasis on minimizing miss distance is particularly relevant here, because the targets are small flying insects and minute errors can determine success or failure. PN is widely applied for missile-to-aerial target interception guidance and has seen renewed interest due to the emergence of drones in warfare.

A key limitation identified of PN for drone guidance in aerial interception is the design for a single interception event [5], [6], [7]. Modifications of the PN control laws enable reattempts at interception [4], [8], [9]. Of the methods developed that are able to re-attempt interception, fast response proportional navigation (FRPN) has empirically demonstrated the best performance [9]. FRPN blends classical PN guidance with an additional term to hasten initial convergence without sacrificing terminal accuracy.

Critically, FRPN inherits reliance on the same idealized assumptions that underwrite PN variants. It performs reliably only when target motion is simple, the LOS measurements are accurate, and the vehicle can track commanded accelerations instantaneously. When these assumptions are not met, the performance of PN and, by extension, FRPN degrade [4]. These limitations motivate the investigation of learned controllers. In this paper, FRPN serves as the conventional baseline against which more elaborate learned controllers are compared.

B. Reinforcement Learning Interception G&C

Reinforcement learned (RL) controllers have shown strong performance on aerial interception tasks. In head-to-head simulation studies, RL controllers surpass classical PN variants on representative interception benchmarks [10], [11], [12], [13]. Beyond interception, RL has achieved state-of-the-art results in a closely related, high-agility flight domain. In drone racing, RL policies have outperformed classical stacks, model-based baselines, and human experts [14]. These results suggest that RL is a credible candidate for aerial interception guidance, where fast, reactive control is essential. Despite this promise, several design choices remain unsettled.

1) Control Abstraction: denotes the level in the flight controller at which the policy outputs actions. The optimal controller for the pest interception problem can be viewed as a function mapping observations to actions. In practice, traditional autopilots are built in layers: a guidance layer computes setpoints, an attitude or rate loop tracks them, and a mixer drives the motors [15]. For more layered controllers, the level of abstraction increases. The actions taken by the policy then are more detached from the actuation. Intermediate controllers bridge the gap between the high-level actions and the motor commands. This layered design limits what a learned policy can express depending on which layer of the controller it is implemented in.

Giving the policy direct access to the lowest layer maximizes the behaviors an optimal mapping can realize. This is the motivation for RL policies that map observations directly to the lowest level commands [15]. For MAVs, the lowest layer is direct motor actuation through RPM commands. Comparative studies report that motor-level control can yield large performance gains in simulation [16], but only marginal improvements when deployed on hardware [17].

Learning policies that output actions at lower levels of control abstraction comes at a cost. Taking position as the final outcome, a motor-command dynamic model used in policy training requires a chain of five integrated states from actuation to positional motion. Each layer depends on identified parameters and/or nonlinear maps, so errors and unmodeled effects accumulate across the chain [15]. Consequently, lower levels of abstraction makes faithful deployment on hardware more demanding than at higher-level abstractions.

A systematic comparison across abstraction levels under the same sensing, training, and flight constraints is still missing. This paper fills that gap by evaluating three levels: guidance-level acceleration setpoints, collective thrust and body rates (CTBR), and direct motor commands.

2) Observation Design: Observation spaces for interception policies are not consistent across implementations. Key design choices include whether to provide raw measurements or engineered features and how much temporal history to include. Studies variously use raw signals [18], [19] or processed features [10], [20]. Similarly, some works report substantial gains with augmenting the policy input with a short history of observations [21], [15], [22], whereas others find little benefit from including short histories [23]. These discrepancies leave open questions about which feature set to use. This paper addresses this gap with a controlled comparison, ablating different observation sets.

C. Reality Gap

The reality gap (RG) denotes the discrepancy between performance in simulation and on hardware. The RG's impact is particularly pronounced for RL controllers. Learned policies are tightly coupled to the simulator used during training.

The controller's behavior is effectively shaped by the fidelity of the simulated environment. This dependency makes sim-to-real transfer a critical bottleneck in RL-controlled MAV-pest interception. The most consequential failure, an unsuccessful sim-to-real transfer, occurs when a policy that appears successful in simulation cannot generalize to real-world conditions [24].

In many areas of robotics, on-hardware fine-tuning or online learning after simulation pretraining is a practical route to strong real life policy performance. For MAVs, however, real-world training is rarely viable: crash risk, hardware damage, safety incidents, and battery power retrains the amount of learning time. Given these constraints, zero-shot transfer is the only practical strategy, policies are trained entirely in simulation and then deployed on hardware without further learning [25]. Several strategies exist to enhance the probability of a successful zero-shot transfer.

1) System Identification: fits dynamics and sensor models to real flight data, increasing simulator fidelity. Overall system identification is widely stated as a beneficial factor for overcoming the RG [26], [24]. Historically, unmodeled effects were compensated with neural-network residuals on top of system identified dynamic models [17]. More recently, high-agility flight has been achieved using simple rigid-body simulators paired with domain randomization (DR) [23], [14]. Crucially, in this approach accurate system identification is still an important factor in facilitating sim-to-real transfer [25].

2) Domain Randomization: addresses the reality gap by training over a distribution of simulators so that the real world appears as an additional draw from that distribution [27]. Prior work reaches different conclusions about how much and what to randomize. Valassakis et al. [28] find that simple disturbance injection can match the transfer benefits of elaborate parameter randomization at much lower engineering cost. For simple trajectory tracking tasks, it is shown that when the simulator has been well calibrated by system identification, additional randomization can degrade learning and transfer by enlarging the policy search space without adding relevant variability [25], [15]. In contrast, Ferede et al. [23] vary the magnitude of randomization for a drone racing task and report that policies trained without randomization excel in simulation yet fail on hardware. Ferede et al. [23] and Molchanov et al. [29] also find that very heavy randomization can enable zero-shot transfer across different platforms. However, this broad generality comes at a cost; per-platform performance is lower than when training on a well-identified model with light randomization. Overall, while modest randomization is often necessary to achieve zero-shot transfer; heavier randomization slows learning and reduces peak performance, and a measurable sim-to-real gap remains in all cases.

Taken together, there is no one-size-fits-all prescription for domain randomization. Because most findings come from case studies at different abstraction levels, broad generalizations are difficult. To address this, we systematically evaluate domain randomization across guidance-layer, CTBR, and motor-level policies on the same task under identical sensing, training, and deployment conditions.

3) Reward Shaping: involves augmenting the task reward with auxiliary terms that bias learning toward behaviors that are easier to transfer to hardware [25]. It is widely used to improve zero-shot sim-to-real transfer in quadrotor RL, though implementations vary with task and control abstraction. Two forms dominate. First, penalties on body rates [23], [14], [15], which are routinely included but not accompanied by an ablations or an explicit rationale. Second action-difference or smoothness penalties that curb simulator-exploiting aggressive commands and enhance hardware stability [30], [25], [20]. Shaping introduces a stability–agility trade-off: more smooth command aids transferability but can cap performance. Accordingly, this paper systematically evaluates reward shaping, quantifying the stability–agility trade-offs in simulation and under zero-shot transfer to hardware.

III. Methodology

A. Problem Formulation

The MAV–pest interception problem may be formulated as a Partially Observable Markov Decision Process (POMDP). In this work, a single agent formulation is adopted: the MAV is the only decision-making agent and is controlled by the policy, while the pest is treated as part of the environment. The tuple,

$$\mathcal{M}_{\kappa,\psi} = \big(\mathcal{S}_\kappa,\, \mathcal{A}_\kappa,\, O_{\kappa\psi},\, P_\kappa, R,\, \gamma\big)$$

defines the POMDP under control abstraction $\kappa \in \mathcal{K}$ and observation configuration $\psi \in \Psi$. Here, $\mathcal{S}_\kappa$ is the state space, $\mathcal{A}_\kappa$ is the action space, and $O_{\kappa\psi}$ is the observation space. The transition kernel $P_\kappa(s_{t+1} \mid s_t, a_t)$ captures the dynamics under abstraction κ. The reward function $R(s_t, a_t, s_{t+1})$ encodes the MAV-pest interception task objectives, and $\gamma \in (0, 1]$ is the discount factor, which influences wether near term rewards count more than distant ones.

We denote the state and observation at time t by $(s_t, o_t) \in (\mathcal{S}_\kappa, O_{\kappa\psi})$, and the action by $a_t \in \mathcal{A}_\kappa$. Our goal is to learn a policy, $\pi_{\kappa,\psi}(a_t \mid o_t)$, which maps observations from O_ψ to actions in $\mathcal{A}_\kappa$, producing precise, agile maneuvers that intercept pest trajectories. The optimization objective is to maximize the expected discounted return,

$$J(\theta \mid \kappa, \psi) = \mathbb{E}_{\tau\sim\pi_{\theta,\kappa,\psi}}\left[\sum_{t=0}^{T-1} \gamma^t\, R(s_t, a_t, s_{t+1})\right],$$

where $\tau = (s_0, o_0, a_0, \ldots, s_T)$ is a trajectory generated by $\pi_{\kappa,\psi}$.

B. Dynamic Models

To accommodate different RL controller placements in the flight stack, we model the MAV at three control abstraction levels, indexed by $\kappa \in \{\text{Acceleration}, \text{CTBR}, \text{Motor}\}$. Each abstraction defines its own state space $\mathcal{S}_\kappa$, action space $\mathcal{A}_\kappa$, and transition model,

$$x_{t+1} = x_t + \Delta t\, f_\kappa(x_t, a_t), \qquad x_t \in \mathcal{S}_\kappa,\ a_t \in \mathcal{A}_\kappa, \quad (1)$$

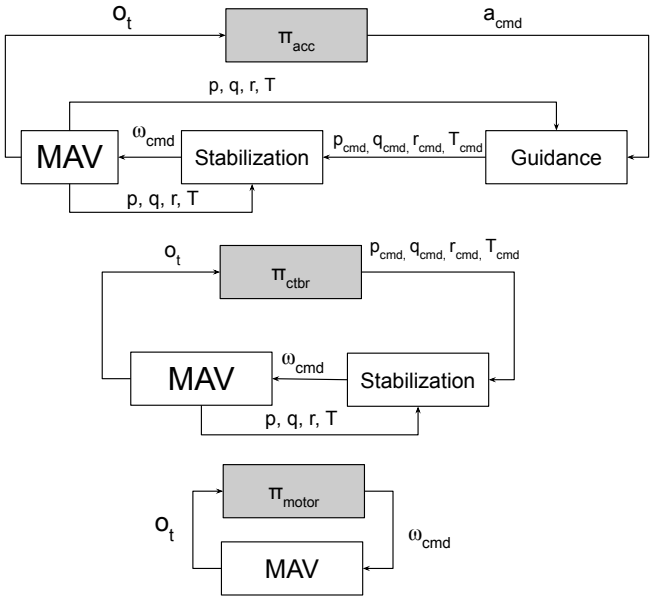


Fig. 3. High-level block diagrams to give an overview of the three control abstraction levels. Top: A guidance-level policy π_{acc} outputs an acceleration setpoint a_{cmd}, which is tracked through the use of two inner loop controllers. Middle: Policy π_{ctbr} outputs $(p_{\text{cmd}}, q_{\text{cmd}}, r_{\text{cmd}}, T_{\text{cmd}})$, which a stabilization controller tracks. Bottom: A motor–level policy π_{motor} outputs rotor–speed commands ω_{cmd} to directly actuate the MAV.

where f_κ denotes the underlying continuous-time dynamics and the environment advances with a fixed step Δt using a forward-Euler scheme during rollouts for learning and evaluation. At the highest level κ = Acceleration, actions correspond to inertial frame acceleration a_{cmd}; at the intermediate level κ = CTBR, actions are collective thrust T_{cmd} and body rates $p_{\text{cmd}}, q_{\text{cmd}}, r_{\text{cmd}}$; and at the lowest level κ = Motor, actions are RPM motor setpoints ω_{cmd}. The full dynamic models, parameter choices, and implementation details are given in Appendix I. A high-level overview of the block diagrams detailing each abstraction level and which inner loop controllers are modeled is shown in Fig. 3.

C. Pest Evaders

Evaders in this study are purely data driven and non–reactive. Evader motion is replayed from logs without feedback from the MAV. The datasets expose the evader's inertial position and velocity time series, denoted by $p_{e,t} \in \mathbb{R}^3$ and $v_{e,t} \in \mathbb{R}^3$.

1) Opogona Moth: A dataset supplied by PATS was recorded with the PATS–X system and contains trajectories of the Opogona moth species. All flights were observed passively; moths were neither pursued by a drone nor otherwise disturbed. The offboard detection system employs an Intel RealSense D455 depth camera producing time-stamped 3D positions at 100 Hz. Raw positions are smoothed with a short moving-average window, velocities are computed via backward Euler differentiation of the raw positions, and a second-order low-pass filter is applied to the derived velocities. This preprocessed data is used throughout the report, matching the preprocessing pipeline of the PATS-X system. To account for temporal mismatches between log playback and the training environment, the nearest available time index in the log is retrieved at each environment step.

2) Pliska Trajectories: A secondary evader source is a dataset from Pliska et al. [9], comprising 100 noise–free, ground–truth trajectories with high–agility maneuvers and a broad range of kinematics, designed to emulate an agile, evasive drone adversary for interception benchmarking. To align magnitudes with greenhouse pests, trajectories are uniformly rescaled so that positions remain within a $3 \times 3 \times 3$ m cube. Unlike the Opogona dataset, ground–truth states are available for Pliska trajectories.

To evaluate the effects of observation noise and preprocessing, measurements are synthesized from ground truth by adding zero-mean Gaussian noise to positions with $\sigma = 0.02$ m per axis to approximate PATS-X sensing noise. Two derived datasets are created: a noisy set, in which noisy positions and velocities derived via backward–Euler are fed directly to the controller; and a preprocessed set, where the PATS-X preprocessing pipeline is applied to the noisy data.

D. Observation Space

The observation space contains a common task–level core that may be shared across all control–abstraction levels. The observation space configurations evaluated in this paper are summarized in Table II. On top of the core, observations that are needed to allow for stable flight under the respective abstraction level are appended. The additional variables per abstraction level are presented in Table I.

Stabilization Feature	Acc	Acc$^+$	CTBR	Motor
Rotation columns $R(\lambda)^\top[:, 1{:}2]$	×	✓	✓	✓
Body rates Ω	×	✓	✓	✓
Specific thrust T	×	✓	✓	×
Normalized rotor speeds ω_{norm}	×	×	×	✓

TABLE I

ABSTRACTION–LEVEL FEATURE BLOCKS NEEDED TO ACHIEVE STABLE FLIGHT APPENDED TO THE CORE OBSERVATION SETS IN TABLE II. ACC$^+$ DENOTES THE ACCELERATION VARIANT THAT INGESTS ONBOARD STABILIZATION CHANNELS. ACC USES ONLY THE OFFBOARD SENSOR INPUTS.

To ease neural policy learning, normalization of the observation space is done. All Cartesian vectors are split into a unit direction vector and a magnitude. The attitude is represented by the first two columns of the world→body rotation matrix $R(\lambda)^\top$, a continuous orthonormal parameterization, rather than Euler angles or quaternions, which are non-unique and introduce discontinuities adverse to neural learning [31].

Finally, a k-step observation history and an m-step action history may be appended, with $k{=}m{=}0$ being the memoryless setting. This yields a consistent interface across abstraction levels while letting us systematically study which feature set and history depth best support learning.

E. Reward

The reward signal explicitly encodes the interception objective and shapes the learning signal so that desired emergent

Input Information	Observation Space Configurations											
	p_d, p_e	p_d, p_e, v_d, v_e	Δp	Δp, Δv	$\Delta p, \Delta v$, $\dot{\phi}$	Δp_b	Δv_b	Δp_b, Δv_b	$\Delta p_b, \Delta v_b$, $\dot{\phi}_b$	p_d, p_e, v_d, v_e, $\Delta p, \Delta v, \dot{\phi}$	p_d, p_e, v_d, v_e, $\Delta p, \Delta v$	p_d, p_e, v_d, v_e, $\Delta p_b, \Delta v_b$
Pursuer position	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
Evader position	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
Pursuer velocity	×	✓	×	×	×	×	×	×	×	✓	✓	✓
Evader velocity	×	✓	×	×	×	×	×	×	×	✓	✓	✓
Relative position world	×	×	✓	✓	✓	×	×	×	×	✓	✓	×
Relative velocity world	×	×	×	✓	✓	×	×	×	×	✓	✓	×
Relative position body	×	×	×	×	×	✓	×	✓	✓	×	×	✓
Relative velocity body	×	×	×	×	×	×	✓	✓	✓	×	×	✓
LOS angular rate	×	×	×	×	✓	×	×	×	✓	✓	×	×

TABLE II

CORE OBSERVATION-SPACE CONFIGURATIONS. SUBSCRIPT b DENOTES BODY-FRAME QUANTITIES; OTHERWISE, INERTIAL FRAME.

behaviors arise during training. The reward as defined in Eq. 2 comprises three elements: a base interception task term, smoothing term that discourages brittle control, and a boundary-adherence reward mechanism that encourages the MAV to stay inside the admissible flight volume. The base reward is briefly introduced here alongside the smoothing reward. Comprehensive motivation and implementation details for the base term are provided in Appendix III. For brevity, the boundary-adherence term is not detailed here; its complete formulation is given in Appendix IV.

$$r_t = r_t^{\text{base}} + r_t^{\text{smooth}} + r_t^{\text{bounds}} \tag{2}$$

1) Base Reward: In the formulation of the base reward, we follow the principle of dense progress shaping from drone racing, where agents are rewarded for per-step approach to the next waypoint [14], [23]. The per-step base reward is defined as,

$$r_t^{\text{base}} = \frac{(d_{t-1} - d_t) - s_t}{d_0} + 10 \cdot \mathbf{1}_{\text{int}}, \tag{3}$$

where, the LOS distance between MAV-evader is $d_t = \|p_{e,t} - p_{d,t}\|$, evader's per-step path length is given through $s_t = \|p_{e,t} - p_{e,t-1}\|$, capture indicator $\mathbf{1}_{\text{int}} = 1$ if $d_t < d_{\text{int}}$ with d_{int} the interception radius, and d_0 is the initial separation between the MAV and pest. An illustration of the base reward is given in Fig. 4.

The reward draws upon a prior work by Vos [32], where the reward is shaped to act as an imperfect proxy for the individual contribution of the agent to the reward. The evader's one-step travel distance is subtracted so that apparent progress caused by coincidental target motion is not falsely attributed to the agent. At each timestep the reward equals the MAV's reduction in range from the previous step minus the evader's distance traveled in that step, normalized by the initial separation, with a fixed bonus upon capture. The step reward is positive when the MAV reduces separation by more than the evader moves; if this persists, the separation decreases and must cross the capture radius in finite time. Therefore, sustained positive reward is a sufficient condition for interception. Subtracting the term s_t also introduces an implicit per-step cost. Longer episodes accrue more penalty, so maximizing return favors trajectories that achieve capture in fewer steps, providing a practical pressure toward time-optimal interception.

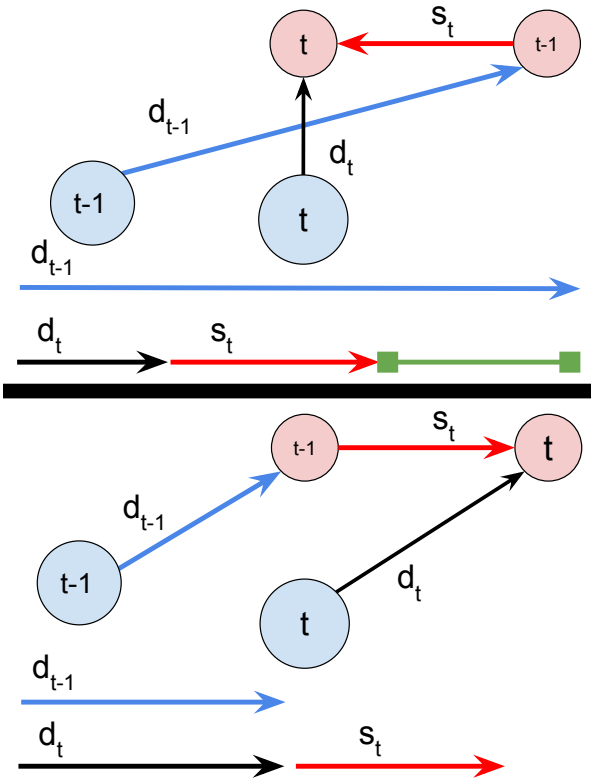


Fig. 4. Base-reward geometry. The pursuer (blue) and evader (red) are shown; d_{t-1} and d_t denote their line-of-sight relative distances. The red arrow indicates s_t, the evader's one-step path length. The top panel depicts a step in which the evader moves towards the pursuer; in this case the credited progress $(d_{t-1} - d_t) - s_t$ (green bracket) is positive and yields a positive base reward r_t^{base}. The bottom panel depicts a step in which the evader moves away from the pursuer; the relative distance does not decrease and the reward is negative.

2) Reward Smoothing: In an effort to enhance policy transferability smoothing terms are appended to the reward. Two terms are optionally included to form,

$$r_t^{\text{smooth}} = -\gamma_s \|\Delta a_t\| - \gamma_r \|\boldsymbol{\Omega}_t\|, \tag{4}$$

where $\Delta a_t = a_t - a_{t-1}$ is the change in the commanded action, $\boldsymbol{\Omega}_t$ are the body rates at time t, and $\gamma_s, \gamma_r \geq 0$ are

small weights.

Action-difference penalties γ_s discourage high-frequency control commands that exploit simulator idealized responses [30], [25]. Similarly, rate penalties γ_r have an analogous role [14]. Conceptually, this reward shaping trades a small amount of agility for smoother actuation.

F. Domain Randomization

Domain randomization (DR) may be systematically applied to each abstraction-level model's parameters. The procedure used here is analogous to what was done in [23]. The parameters of the different abstraction level models as defined in Appendix I, were estimated via least-squares regression with real flight logs. For a complete overview of identified values, consult Appendix VI. Accurate system identification is critical because it anchors DR around an initial best estimate of the dynamic model. This anchoring is necessary for a fair evaluation of DR's effectiveness, otherwise any observed gains or failures may reflect identification bias rather than the benefits of DR [25].

We apply DR by scaling each component of the identified parameter vector $\hat{\boldsymbol{\theta}}_\kappa$ by an i.i.d. uniform factor,

$$(\theta_\kappa)_i \;\leftarrow\; \hat{\theta}_{\kappa,i}\, \mathcal{U}[\,1-p,\, 1+p\,], \tag{5}$$

where, $p \in [0, 1)$ is a user-set percentage that controls the spread. After sampling, we clip each parameter to its physically admissible range, to ensure the randomized set remains feasible. In the experiments conducted DR is applied at four levels, nominal ($p = 0$), 10% ($p = 0.10$), 20% ($p = 0.20$), and 30% ($p = 0.30$). During training, at the start of each rollout, a simulator instance is sampled by drawing one set of parameter multipliers, each environment receives an independent draw.

G. RL Training Algorithm

Policies are learned with Proximal Policy Optimization (PPO) implemented using Stable-Baselines3 [33] on a vectorized Gym environment. The same training pipeline is used across all input configurations. Following prior work in RL-controllers for aerial-to-aerial interception [32], an on-policy method is used despite the inherent lower sample efficiency. In our setting, this reduced sample efficiency is not a major limitation because simulation rollouts are inexpensive. Moreover, PPO's modest hyperparameter sensitivity makes it reliable to train without extensive tuning [34].

Concretely, training runs with 100 independent parallel environments and proceeds in updates that collect 1,200 steps per environment. The simulator advances with a fixed step size $\Delta t = 0.01$. In the case of no early termination, a batch size of 5000 is used, and ten gradient epochs with a discount factor of $\gamma = 0.999$ are performed. The policy and value networks share the same architecture: a three-layer multilayer perceptron (MLP) with 64 units per layer. This compact network is suitable for deployment on lean hardware and, in a comparative study evaluating multiple architectures for drone racing, achieved the best performance [23]. Unless stated otherwise, each run trains for a total of 50 million environment steps, accumulating to a total of 5 days and 19 hours of simulated flight time used for training. The code containing all the different models and environments used for training is available on https://github.com/tudelft/Aerial_To_Aerial_Interception.

H. Fast Response Proportional Navigation

A classical interception guidance controller was also implemented to serve as a benchmark for the RL controllers. The controller implemented here stems from PN. One can recast interception guidance within a linear–quadratic optimal control framework and derive a family of optimal laws of increasing complexity as more target information and actuator dynamics are modeled [4]. Many of these laws subsume classical PN variants, providing principled, Cartesian-space counterparts that explicitly optimize a specific objective.

Of particular relevance here is Linearized PN (LPN), which under certain conditions is optimal with respect to terminal miss distance, a critical objective for pest interception due to the small target size. Note, that in this investigation the conditions for LPN to be optimal with respect to terminal miss distance were not explicitly enforced. Pliska et al. [9] build upon LPN proposing FRPN, which augments LPN with a small pure pursuit (PP) term, position-error based feedback, to hasten the initial response. The FRPN command reads,

$$a_d = G\Bigg((1-W)\, \underbrace{\frac{\Delta p_t + \Delta v_t\, t_{\mathrm{go}}}{t_{\mathrm{go}}^2}}_{\text{LPN term}} + W\, \underbrace{\Delta p_t}_{\text{PP term}} \Bigg). \tag{6}$$

where $\Delta p_t = p_{e,t} - p_{d,t}$ and $\Delta v = v_{e,t} - v_{d,t}$ are relative position and velocity, a_d is the inertial-frame acceleration command, $t_{\mathrm{go}}(t) = \|\Delta p_t\|/\|\Delta v_t\|$ is a time-to-go till interception estimate, G is the navigation gain, and $W \in [0, 1]$ blends the LPN term with the pure pursuit term. Intuitively, as $t_{\mathrm{go}} \to 0$ the LPN term dominates, so the added pure pursuit term accelerates early engagement while preserving terminal precision of LPN.

IV. Experimental Set-up

A. Simulation

Simulation-based evaluation benchmarks all controllers under standardized, repeatable conditions using playback of evader trajectories, specifically, the two evader datasets presented in Section III-C are considered. Policies are evaluated using a held out set of 10 evaders selected at random. For each evader, 100 trials are executed with the pursuer initialized according to Appendix II, yielding broad coverage of relative geometries and engagement conditions across the MAV–pest interception task. Policies are evaluated deterministically to ensure repeatability, the mean action is applied at every step. Unless noted otherwise, dynamics and sensing during evaluation mirror the training configuration. The MAV is simulated with its nominal system-identified parameters. Episodes run until the fixed maximum duration of 6 seconds is reached.

B. CyberZoo Testing

Physical experiments were conducted with a Parrot Bebop 2. The platform's long battery life minimized downtime between test trials, and the IMU exposes per-motor RPM, which is necessary for the system identification pipeline. The onboard CPU is fast enough to run the learned three-layer MLP with 64 units per layer policies directly on the vehicle. While the integrated Wi-Fi module supports telemetry, messaging and logging.

Flights were carried out in the CyberZoo at TU Delft's Faculty of Aerospace Engineering, a flight arena equipped with an OptiTrack motion-capture system. Real-time position and attitude from OptiTrack are fused onboard with IMU measurements using an extended Kalman filter, yielding low-latency state estimation onboard the Bebop 2. The flight software is a MAVLab TU Delft fork of the open-source Paparazzi-UAV autopilot [35], which contains the intermediate controller needed to implement all abstraction levels. The onboard realization of each abstraction is summarized in Fig. 3.

Trials use an offboard evader playback, a held-out Opogona trajectory is streamed to the MAV. Each trial begins by briefly mimicking the target as a stationary point at a randomized setpoint on a circle with a radius of one meter centered at the evader's initial spawn location. The recorded moth flight is then replayed for 6 seconds, and the Bebop 2 attempts to intercept this virtual target. The chosen moth remains well within the arena, keeping the vehicle far from the walls and enabling safe, repeatable testing across policies. The procedure is illustrated in Fig. 5.

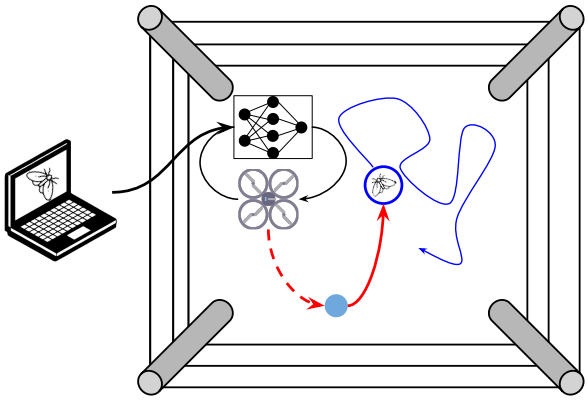


Fig. 5. CyberZoo testing procedure. Virtual target state is streamed from an offboard computer to the drone. The onboard neural network policy fuses the streamed target state with the drone's own state and outputs control commands. Each trial begins by streaming a mock stationary target (light-blue dot) placed 1 m from the trial spawn point to initialize the pursuer; thereafter the real target trajectory is streamed and the interception case is started.

V. Results

A. Evaluation Metrics

Following prior work on aerial-to-aerial interception [9], [32], this work adopts task-level metrics tailored to MAV pest interception to compare controllers on effectiveness, persistence, and spatial behavior. The interception rate p_{int} is the fraction of trials with at least one interception. The time-to-first-interception t_{int} is the elapsed time until the MAV–target distance first drops below 15cm, indicating how quickly interception is achieved. Evaluation trials do not terminate upon an interception, therefore also proximity metrics may be derived. These metrics are secondary in importance. To capture near-successes, near-miss time t_{NM} records the total duration spent within 30cm of the target, while the mean distance $\bar{d}$ is the average separation over the engagement, indicating overall proximity maintenance.

B. Observation Space

In this section the evaluation of how different observation space configurations influence interception performance is presented. Observation space choices are evaluated using the motor pursuer. Among the considered abstraction levels, the motor policy provides the most direct link between observations and actuation, making it the most representative setting for isolating the effect of observation design. The training and evaluation method presented in Section IV-A on the Opogona dataset presented in Section III-C, was used. Performance across all intercepting configurations is presented in Table III.

Relative position Δp_b and velocity Δv_b expressed in the body frame show the best performance with respect to the primary objective, the first interception time, while also remaining competitive across the other metrics. This makes it the best-performing observation space, which will therefore be used for the remainder of the analysis presented in this report. Two clear trends emerge from the observation–space analysis.

Observation Set	$\mathbf{t}_{\text{int}}$ [s] (median [Q1–Q3])	$\mathbf{t}_{\text{NM}}$ [%] (mean $\pm$ std)	$\bar{\mathbf{d}}$ [m] (mean $\pm$ std)	$\mathbf{p}_{\text{int}}$ [%]
$[\Delta p, \Delta v]$	1.77 [1.38–2.32]	61.64 ± 13.43	0.34 ± 0.09	98.82
$[\Delta p_b, \Delta v_b]$	**0.85 [0.76–1.07]**	59.36 ± 11.58	0.32 ± 0.06	**100.0**
$[\Delta p, \Delta v, \dot{\boldsymbol{\phi}}]$	3.11 [1.88–4.43]	11.46 ± 4.33	1.32 ± 0.51	97.61
$[\Delta p_b, \Delta v_b, \dot{\boldsymbol{\phi}}]$	1.18 [0.94–1.85]	33.54 ± 8.33	0.54 ± 0.12	**100.0**
$[p_d, p_e, v_d, v_e,\ \Delta p, \Delta v]$	1.60 [1.19–2.28]	$\mathbf{73.14 \pm 9.70}$	$\mathbf{0.28 \pm 0.06}$	**100.0**
$[p_d, p_e, v_d, v_e,\ \Delta p_b, \Delta v_b]$	1.09 [0.88–1.40]	62.50 ± 11.99	0.29 ± 0.04	99.33
$[p_d, p_e, v_d, v_e,\ \Delta p, \Delta v,\ \dot{\boldsymbol{\phi}}]$	2.35 [1.66–3.57]	17.22 ± 7.43	1.27 ± 0.77	97.11

TABLE III

Task-level metrics by observation-space configuration. Only first-interception time t_{int} uses a skewed summary (median [Q1–Q3]); time in near-misst_{NM} and $\bar{d}$ reports mean $\pm$ std; p_{int} is the interception rate.

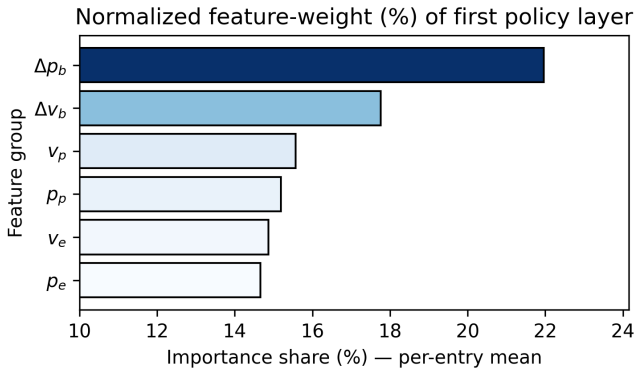


Fig. 6. Relative importance of first-layer policy inputs for the feature set $[p_d, p_e, v_d, v_e, \Delta p_b, \Delta v_b]$. Absolute input weights into the first linear layer are grouped by feature and normalized per entry, yielding comparable importance across observation sets of different sizes.

First, including the LOS rate $\dot{\phi}$ substantially degrades the interception performance. Pooling the inertial and body-frame variants, adding the LOS rate $\dot{\phi}$ increased the median first–interception time t_{int} by $\approx 1.5\times$ and inflated the overall proximity $\bar{d}$ by $\approx 2.8\times$. This is noteworthy because LOS rate is the core feedback signal in classical closed-loop PN guidance laws [36]. The likely cause is the LOS rate's discontinuous behavior. After a near miss, the LOS-rate magnitude spikes. At closest approach the bearing to the target swings rapidly as the MAV passes the target. The LOS direction flips in a few samples, this produces an apparent jump in the measured rate. Theoretical analyses indicate that, for a fixed number of neurons, smoother target functions, those with stronger continuity properties, admit smaller approximation errors [31]. Consequently, under this representation the mapping from LOS rate to the required actuation is not smooth and therefore harder to approximate stably with a neural network.

The second identified trend, is that representing relative features in the body reference frame emerges as a best practice. Pooling the results of the different configurations tested, expressing $\Delta p, \Delta v$ in the body frame reduced median first interception time t_{int} by $\approx 40\%$. Intuitively, aligning target information with the MAV's body axes simplifies the mapping from observations to actuation.

The following observation sets did not achieve interception: inertial-frame positions (p_d, p_e), inertial-frame positions and velocities (p_d, p_e, v_d, v_e), and relative positions (Δp or Δp_b) or velocities (Δv or Δv_b). The minimal observation set that enabled interception was relative position and relative velocity. To quantify observation feature saliency, we inspect the first linear layer of the policy network trained with all features. For each input, we sum the absolute incoming weights, aggregate them by semantic feature group, take a per–entry mean within each group, and normalize across groups to obtain percentage shares of importance. The goal is to identify which observation groups most strongly shape the policy's initial transformation, providing a proxy for feature importance. As shown in Fig. 6, relative position and velocity in the body frame dominate the first-layer importances, indicating that the policy relies most on these signals at the input stage. Although this analysis does not establish causality, it does give some indication of which features are most influential for the interception task.

C. Observation History

The effect of adding observation history was examined to evaluate whether temporal context enables non-intercepting observation sets to learn interception. Observation history length, was varied over $k \in \{1, 2, 4, 8, 16\}$. The same training and evaluation pipeline is used as in the previous analysis, using the Opogona dataset introduced in Section IV-A and evaluating according to what is stipulated in Section IV-A.

All configurations that did not learn to intercept with no history $k{=}0$ remained unable to intercept for every tested history length. This outcome suggests that the network did not reliably infer the required latent variables from the historical observation sequences. This outcome is somewhat surprising, as neural networks are capable of universal function approximation [37], and one might expect the policy to infer omitted latent variables from the historical sequence of observations. For example, when provided only with the last k relative positions, an MLP could in principle approximate the relative velocity by differencing across time, thereby implicitly reconstructing the minimal observation set known to enable interception.

However, the combination of noisy inputs and limited network capacity likely explains why this theoretical potential was not realized in practice. The Opogona measurements are inherently noisy, any velocity estimated from sequences of positions will suffer from amplification of this noise. At the same time, the relatively shallow and narrow MLP lacks the representational capacity to approximate the more

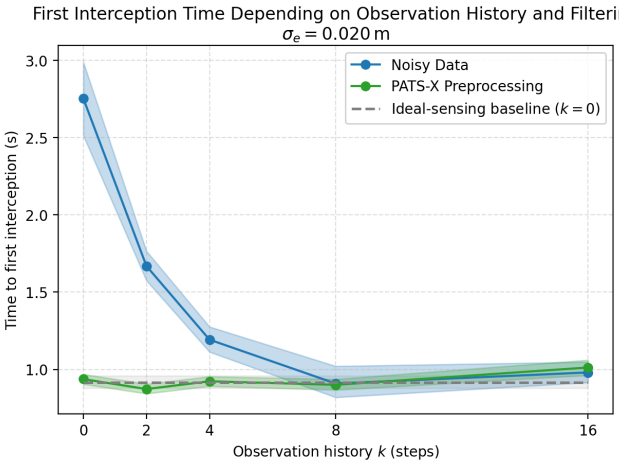


Fig. 7. First-interception time depending on observation history k for the observation set $[\Delta p_b, \Delta v_b]$, evaluated on Pliska trajectories with Gaussian position noise $\sigma = 0.02$ m; metrics computed against ground truth. Results are shown as median and IQR, denoted through the solid line and shaded region, respectively. The ideal-sensing baseline is trained without noise and uses no history ($k = 0$).

complex functions required to recover such latent features under noisy conditions. Scaling the network depth or width might mitigate this, but was not pursued, as the policies are intended for deployment on lightweight flight computers.

A second analysis quantifies how interception performance depends on the amount of observation noise and the available history. For this analysis the best performing observation set, $(\Delta p_b, \Delta v_b)$, was used. The different Pliska trajectories, ground-truth, noisy, and preprocessed as presented in Section III-C were used. The results are reported in Fig. 7 as first-interception time depending on history length k. Under preprocessed inputs, which matches the pipeline of the PATS-X system, increasing k does not improve performance. Under noisy inputs, a longer history is required; approximately the last $k \geq 8$ observations restore performance to the level achieved with preprocessing. These findings indicate that temporal stacking serves primarily as noise filtering when preprocessing is absent, whereas it offers little benefit once light filtering reduces the effective noise floor.

A final analysis evaluates whether using noisy measurements for both training and evaluation, alters the reported interception performance. The Pliska dataset was used and evaluation metrics were computed twice for each policy: once against the preprocessed dataset, and once against the corresponding ground-truth positions. The two evaluation methods produced essentially identical interception metrics, indicating that measurement-only evaluation does not bias conclusions at this noise level.

The implication of this analysis is that training and evaluation can be performed directly on the Opogona dataset using PATS-X preprocessed features. Letting the policy self-filter via observation history provides no advantage over PATS-X preprocessing. With PATS-X preprocessing in place, adding observation history does not improve performance. Evaluation on PATS-X–preprocessed measurements is indistinguishable from evaluation on ground truth. Choosing the preprocessed Opogona data therefore simplifies the pipeline by removing the need for ground-truth labels, while matching the deployed sensing conditions.

D. Action History

We evaluated whether appending recent actions to the observation improves policy performance. The action–history length was swept over $m \in \{1, 3, 5, 10, 20\}$. The resulting evaluation showed no gain from larger histories, see in Fig. 8 the evaluation reward during learning for the motor abstraction level. Performance for was comparable for short histories. Longer histories reduced sample efficiency, the policy needed more environment steps to reach the same return.

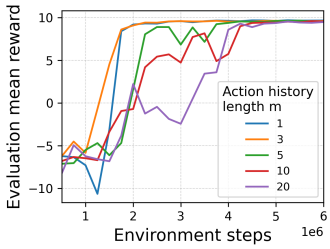


Fig. 8. Mean episode reward depending on action history length during training

A motivation to include the action history is that MAV dynamics and inner–loop controllers introduce lag between commanded action and the effect on the measured state. When such delays are present, learning is more difficult because the reward that is the result of an action may be observed several steps later [15]. Supplying a short window of past actions can, in principle, help the policy disentangle which part of the current response is due to its own previous commands. However, in our experiments no consistent benefit was observed from increasing the action–history length, aligning with the results of [23].

E. Abstraction Level

This section quantifies how control abstraction level affects interception performance. Models at the three abstraction levels, acceleration, CTBR, and motor, were evaluated under the simulation evaluation procedure described in Section IV-A using the Opogona dataset as introduced in Section III-C. Quantitative results are reported in Table IV. Note, the acceleration controller evaluated here is the augmented variant Acc$^+$ which receives the same stabilization channels as CTBR, as stipulated in Table I. This alignment ensures that performance differences reflect abstraction level rather than observation space.

Median first interception time improves consistently as we move to lower abstraction. A performance gain of $\approx 8\%$ can be seen when going from acceleration to CTBR, while $\approx 41\%$ may be gained when moving to the deepest abstraction level. Additionally, variance of the time to the first interception decreases as we go lower in control abstraction, indicating more consistent interception performance. Time in near-miss increases from acceleration to CTBR and motor, consistent with the possibility of more precise control at the lower abstractions.

The performance gain may be attributed to the fact that lower control abstraction increases the policy's expressiveness. When an intermediate controller is present, it enforces control with filtering, limits, latency, and compresses the policy's command space. By enlarging the feasible control envelope, the policy can realize a richer set of behaviors, which in practice yields better interception performance.

Method	$\mathbf{t}_{\mathrm{int}}$ [s]	$\mathbf{t}_{\mathrm{NM}}$ [%]	$\mathbf{p}_{\mathrm{int}}$ [%]
FRPN	1.90 [1.04–2.80]	14.31 ± 4.18	95.6
ACC	1.86 [1.24–2.60]	29.80 ± 8.57	98.2
ACC$^+$	1.44 [1.15–1.76]	32.31 ± 9.29	100.0
CTBR	1.31 [1.01–1.78]	40.04 ± 9.10	100.0
MOTOR	0.85 [0.76–1.07]	55.62 ± 11.58	99.1

TABLE IV

SIMULATION TASK-LEVEL METRICS BY CONTROL ABSTRACTION LEVEL. ONLY FIRST-INTERCEPTION TIME t_{int} USES A SKEWED SUMMARY (MEDIAN [Q1–Q3]); TIME IN NEAR-MISS t_{NM} REPORTS MEAN $\pm$ STD; p_{int} IS THE EMPIRICAL INTERCEPTION RATE.

F. Comparison to Fast Response Proportional Navigation

This subsection evaluates the RL controllers against the classical FRPN method using the standardized simulation

protocol in Section IV-A and the Opogona dataset as presented in Section III-C, with the results summarized in Table IV. The comparison with the acceleration-level RL policy is fair because it operates at the same control abstraction as FRPN and, for this head-to-head, uses the nominal acceleration observation space as defined in Table I. This mirrors FRPN's state requirements and interface, ensuring that any performance differences reflect the guidance law rather than differences in sensing or observation design.

The biggest advantage of the RL acceleration abstraction controller is the ability to retain proximity to the target, as reflected by the near-miss time. The speed at which the first interceptions are done are very comparable. However, FRPN achieves faster fastest interceptions as seen in the first quartile first interception time performance, indicating it can be very quick under favorable conditions. Finally, also a small drop in interception rate is seen for the FRPN controller. Overall, the RL acceleration policy trades some blazing-fast best cases for consistent interception and maintaining tighter proximity to the target.

The drops in performance of the FRPN control law are due to the underlying assumption that the response to an acceleration command is instantaneous. The FRPN control law is optimal for terminal miss distance under this assumption. However, as response lag increases, the terminal miss distance increases under FRPN guidance [4]. The RL controller, while not receiving any direct information about actuator lag, has inherently learned to account for this and therefore enables better proximity control and slightly better overall interception performance.

Formulations of PN exist that can account for non-idealized responses. These laws are able to better retain performance over increasing levels of actuator lag [4]. However, these laws rely on an acceleration feedback signal from the MAV. Given the constraints of the PATS-X system, where differentiation is used to derive an acceleration estimate of the drone from positional data, this signal is too noisy to facilitate these control laws. For this reason, these control laws were not considered within the scope of this investigation.

In real-world deployment, following the testing procedure presented in Section IV-B, the FRPN controller achieved the best first–interception times among all the methods. Across the collected runs, the first–interception time had a median of $1.37[1.01, 1.96]$ s, consistent with the observation in simulation that FRPN achieves fast interception in favorable engagements. However, every physical FRPN test ended in a premature crash; consequently, the data reported here reflect only those trials that reached interception before failure. As a result, the reported performance is biased toward faster outcomes and likely over-represents the fastest quartile behavior noted previously.

A direct real-world comparison to the RL–acceleration controller is not possible. The RL–acceleration controller failed to transfer, and overall the use of acceleration commands led to very poor stability on the Parrot Bebop2 platform. Unfortunately, the problems associated with the acceleration abstraction on the Bebop 2 remain unsolved; therefore, effort during this thesis focused on the CTBR and motor abstraction levels, which transferred more reliably.

G. Domain Randomization

Policies for the motor and CTBR abstraction level trained with different degrees of domain randomization were deployed and tested following the procedure highlighted in Section IV-B and simulation based results were gathered following the procedure stipulated in Section IV-A, using the Opogona dataset defined in Section III-C. The results are presented in Fig. 9.

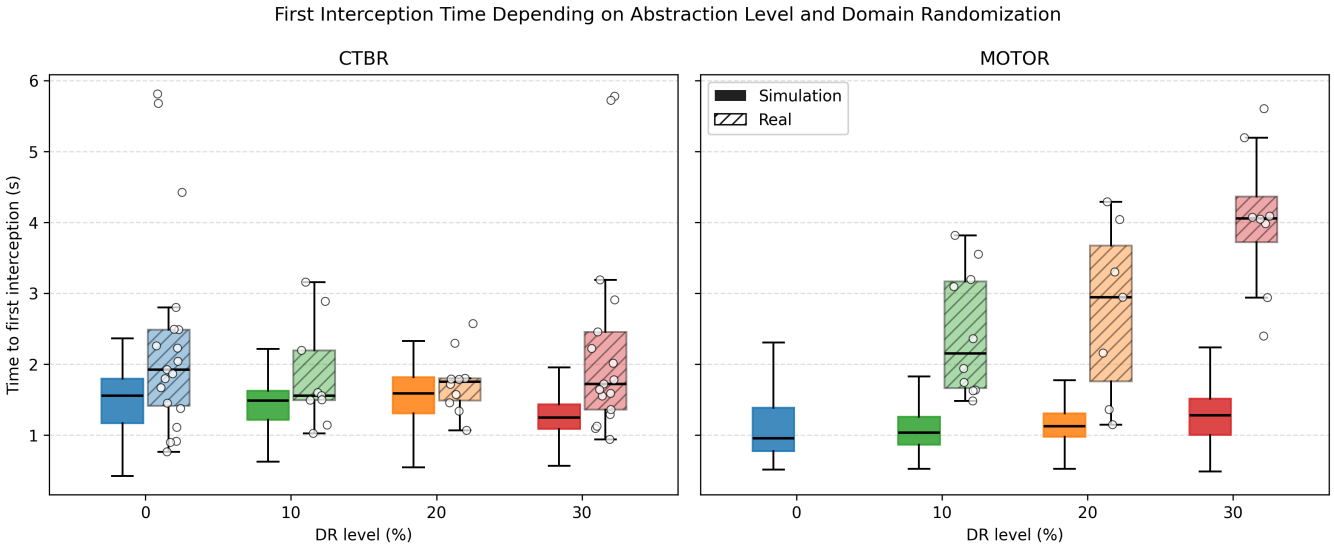


Fig. 9. First-interception time (successes only) for CTBR (left) and Motor (right) pursuers across domain-randomization (DR) levels. For each DR%, distributions are shown for simulation (solid boxes) and real flights (hatched boxes); solid line denoted median, boxes show the interquartile range, and individual real-flight trials are overlaid as points.

In simulation, the motor-level pursuer performs best with no domain randomization, and its performance degrades steadily as randomization increases, as was also observed in [23], [16]. When randomization is off, training and test dynamics coincide, so the policy can fully exploit the simulator's fixed response.

Since the policies are purely Markovian, no context of the current model dynamics is given, nor can it be inferred; the policy chooses the same action for the same observation across all randomized models. Therefore, under domain randomization, actions are conservatively biased, as they must safely operate across all draws of the model distribution, even though the current model may be able to fly more aggressively. This conservatism prevents the controller from using more aggressive, but viable, actions on favorable models, lowering average performance [38].

The effect is especially pronounced at the motor level, where the actions must pass through a long, parameter-sensitive actuation chain [15]; randomizing those parameters broadens the feasible set unevenly and pushes the learned strategy toward risk-averse behavior. By contrast, at the CTBR level the action semantics are stable across models: commanding a roll-rate or thrust setpoint produces the same qualitative motion even if the exact time constant or gain varies. The shorter, more linear actuation chain makes the mapping from action to vehicle motion less parameter-sensitive, so randomization mostly perturbs timing rather than behavior, and the policy does not need to hedge as aggressively. Consequently, the CTBR policy's simulation performance shows no measurable dependence on the randomization level, consistent with prior findings [16].

On the real system, the motor pursuer trained with 0% DR failed to transfer, consistent with the findings of Ferede et al. [23]. The learned policy overfit to the identified dynamics and could not control the real drone. Increasing DR enabled transfer, but raising it further progressively degraded performance for the same reasons as in simulation. This contrasts with Eschmann et al. [15], who reported successful deployment with zero DR. A likely reason is the task difference. Eschmann et al. study trajectory tracking with moderate agility, whereas Ferede et al. [23] and our interception task require racing-level agility. Such aggressive flight drives the vehicle into more extreme states where unmodeled effects are more prevalent, so zero-DR policies fail to generalize, making some level of randomization necessary for transfer.

Overall the motor-level policy performed substantially worse on hardware, falling outside the simulation distribution. This suggests a mismatch between the real vehicle and the dynamics model used for training. One plausible cause is that the model originates from drone-racing platforms with rigid airframes [23], whereas the Parrot Bebop 2 is more flexible, possibly accounting for a larger mismatch.

All CTBR randomization levels transferred successfully; no clear merit is identified of domain randomization for CTBR transfer, consistent with [25]. While CTBR performance on hardware was slightly worse but remained within the bounds of the simulation results distribution. This result highlights the simplicity of sim-to-real transfer at the CTBR level.

H. Reward Smoothing

The goal of the reward smoothing methods is to increase the transferability of a learned policy to the real platform. Policies were trained with either a term γ_s in the reward penalizing the difference between consecutive actions, $\|\Delta a_t\|$ or a term γ_r penalizing the magnitude of the measured rotational rates $\|\Omega_t\|$. Policies were trained across a wide range of γ_s; however, for the rotational-rate penalty only $\gamma_r = 0.001$ was used, in accordance with values reported in [14], [17], [15]. The trained policies were evaluated in simulation and on the Parrot Bebop2, following the procedures stipulated in Section IV-A and Section IV-B, respectively. Note, a domain randomization of 10% was applied to all trained policies to ensure feasibility of transfer.

To quantify the smoothness of different policies, we adopt the frequency-domain smoothness metric proposed by Mysore et al. [30]. To avoid confusion between the name and the meaning of the numerical value, in this work we refer to this quantity as the *roughness measure*, while noting that it is mathematically identical to the smoothness metric S_m in [30]. Concretely, let M_i be the amplitude of the i-th nonzero frequency component of the action signal, and let f_i be the corresponding frequency, with sampling frequency f_s. The roughness measure is then,

$$R_m = \frac{2}{n\, f_s} \sum_{i=1}^{n} M_i f_i, \tag{7}$$

where n is the number of nonzero frequency bins. Intuitively, large weights at high frequencies make R_m large, whereas signals whose energy is concentrated at low frequencies

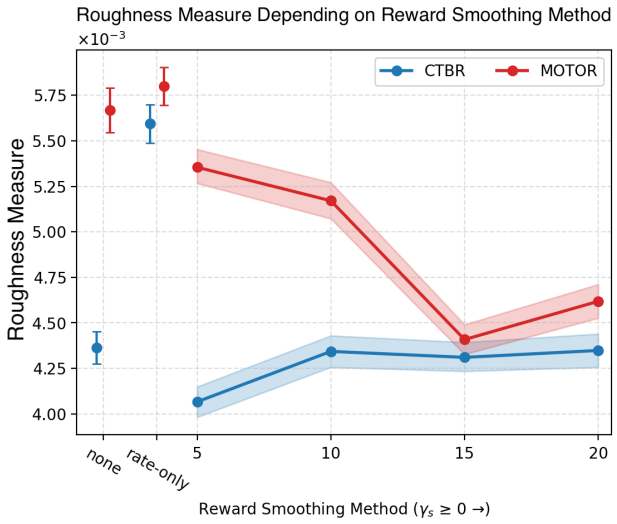


Fig. 10. Roughness measure depending on reward smoothing method: none, rate-only, and $\gamma_s \in \{5, 10, 15, 20\}$, computed over 1000 evaluation trials in simulation using each method's best performing policy. For CTBR (blue) and motor (red). Markers show means; shaded regions indicate 95% confidence intervals.

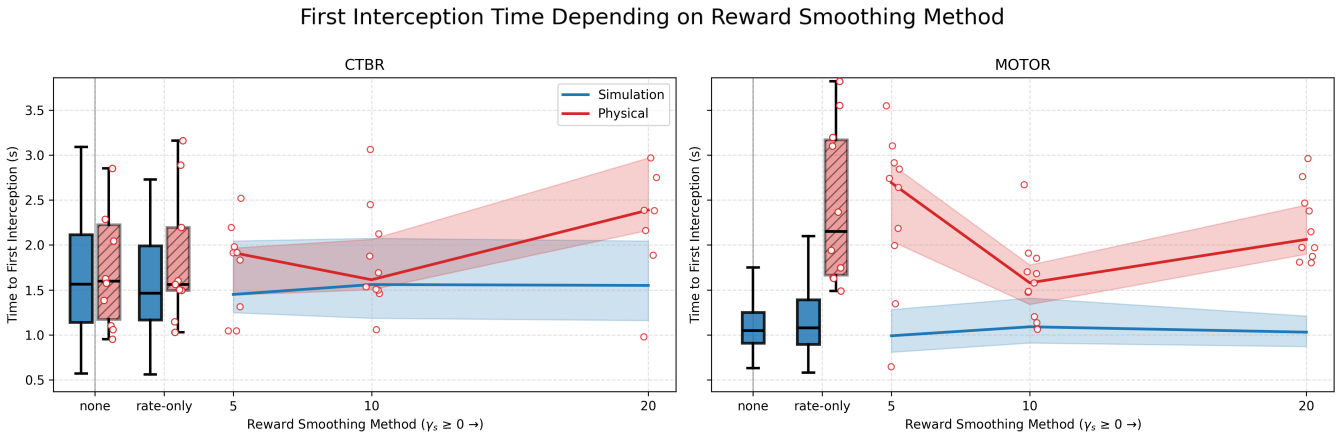


Fig. 11. Time to first interception depending on reward smoothing for CTBR (left) and motor abstraction levels (right) in simulation (blue) and physical (red). No smoothing and rate-only smoothing are shown as box plots; for $\gamma_s \geq 5$ line denotes the median and the shaded band is the IQR. Open red circles indicate individual physical trials

yield a small R_m and smoother commands. See in Fig. 10 the effect of the different reward smoothing options on the roughness measure.

The action-difference penalty γ_s exhibits different effects across abstraction levels. For CTBR, introducing γ_s does not reduce the roughness measure; values remain clustered around the no-smoothing baseline with no identifiable trend. A surprising result, given that Mysore et al. [30] report that action-difference penalties improve the roughness measure across all benchmark tasks and configurations tested.

For the motor level, γ_s lowers roughness measure relative to the no-smoothing baseline, and a weak monotonic tendency appears, larger γ_s generally yields more smoothing, except for a pronounced drop at $\gamma_s = 15$. This irregularity underscores that reward shaping is a soft constraint, it biases learning toward smoother commands but does not guarantee the eventual behavior. Because the policy is obtained via stochastic optimization in a nonconvex setting, smoothing can manifest unevenly, as reflected by the sudden roughness measure decrease at $\gamma_s = 15$.

In contrast, the rate penalty γ_r increases roughness measure for both abstraction levels, especially for CTBR. Reducing the body-rate magnitude $\|\boldsymbol{\Omega}\|$ requires additional control effort; to maintain maneuverability while keeping rates small, the policy issues more frequent, alternating adjustments, which raises roughness measure. The effect is stronger for CTBR because the action directly commands body rates, tightly coupling the penalty to the control channel.

A comparison of the simulation and deployed performance for the smoothed policies is shown in Fig. 11. In simulation, reward smoothing does not change first–interception time for either CTBR or motor abstractions.

At the motor abstraction, lowering roughness measure via γ_s clearly improves transferability up to a point. Without smoothing, rapid voltage and RPM setpoint changes overheated the motors, occasionally melting propellers or coil insulation. This effect diminished markedly for $\gamma_s \geq 5$, consistent with reports that action-regularized quadrotor controllers transfer better to hardware and substantially reduce motor power consumption and heating [30]. Within that safe band, $\gamma_s = 10$ achieved the best deployed performance, combining stable flights with tighter dispersion and faster first–interception times. Pushing smoothing further proved counterproductive. At $\gamma_s = 15$ the policy failed to sustain stable flight; notably, this setting produced a lower roughness measure than at $\gamma_s = 20$, indicating stronger suppression of action changes. The instability at $\gamma_s = 15$ suggests that driving roughness measure too low starves the controller of the bandwidth needed for timely corrections on hardware. Increasing to $\gamma_s = 20$ which has a higher roughness measure, restored stability, but first–interception times degraded, consistent with a more sluggish controller. Together, these results point to a practical control smoothness sweet spot; smoothing must be sufficient to respect actuator thermal limits, yet not so strong to compromise responsiveness and capture agility.

An advantage at the motor abstraction is that reward smoothing achieves the same goal as a low-pass filter, reducing high-frequency command jitter, but does so adaptively rather than imposing fixed attenuation on every command. A fixed low-pass filter enforces uniform smoothing at all times. It guarantees deployable, thermally safe commands, but can be overly restrictive by limiting the controller's bandwidth precisely when brief, decisive adjustments are needed. In contrast, a soft penalty on action differences reduces average command jitter, thereby improving safety and transfer, while still allowing occasional, short bursts of higher action increments when critical. Such sparse, high-difference steps do not accumulate enough thermal load to damage hardware, yet preserve the agility required for timely interceptions.

At the CTBR level, no consistent benefit from reducing the roughness measure was observed: using γ_r produced much smoother commands, but first–interception time remained unchanged relative to the no-smoothing baseline, and setting $\gamma_s \in \{5, 10\}$ likewise yielded no meaningful difference. This contrasts with results on trajectory-tracking tasks, where

action-difference penalties have been shown to improve performance [25]. At $\gamma_s = 20$ a degradation in performance was observed. However, this result was not accompanied by a corresponding change in roughness measure, so no conclusion can be drawn about the underlying cause.

A plausible explanation for the negligible effect of smoothing under the CTBR abstraction is that the policy issues command variations at frequencies higher than the bandwidth of the inner controllers. Those high–frequency components are already rejected by the low-level controllers, so adding an action-difference penalty does not change the realized actuation unless it is strong enough to shift the policy's command spectrum into the trackable range of the inner controller.

VI. PATS-X Development

This section translates the lessons from simulation and CyberZoo testing into the concrete design of the PATS-X system. Wherever feasible, we adopt the identified best practices. The resulting architecture is tailored to fit within the existing PATS-X stack and is a proof of concept designed to showcase the potential of RL within the constraints of the existing system. The objective here is not a production ready controller, but a validated integration path that demonstrates how RL can augment the current stack and motivates the adoption of RL within future PATS-X development. Set-up details of the PATS-X system and how the RL-controller is integrated may be found in Appendix V.

A. Experiments

Each experiment begins with the drone docked at a station mounted near the top plane of the detection camera's viewing frustum, see in Fig. 1 the camera placement and viewing frustum. At the start of a trial, the drone releases from the dock and immediately enters the pursuit phase under offboard guidance. PATS-X exposes only an acceleration command interface, so both the baseline PATS controller and the RL policy operate at the same acceleration abstraction. The PATS controller is a LPN–style guidance law consistent with Eq. 6.

Performance is evaluated using two categories of evader trajectories that capture both natural pest motion and controlled repeatability, recorded moth flights and virtual insects. The first group consists of replays of Opogona moth trajectories as introduced in Section III-C. From the full dataset, five moths were selected whose flight paths remain largely within the observable volume, ensuring that interception attempts are feasible. Because some recordings originate from greenhouse environments with larger allowable volumes, only those that conform to the test facilities space are used. In addition to these five recordings, one artificial replay was generated by modifying an existing moth trajectory into a small figure-eight pattern fully contained within the frustum. This controlled case provides a consistent benchmark for evaluating in-frustum pursuit performance.

The second category of evaders consists of virtual insects exhibiting simple, rule-based reactive behaviors designed to emulate characteristic moth escape responses. Each virtual insect begins its flight along a straight line across the frustum at a constant velocity of 0.5 m/s. Two evasive behaviors are modeled, a startle dive and a proximity-driven evasion. In the startle dive, the insect accelerates downward as soon as the drone's motors spool up, reflecting the tendency of real pests to dive towards cover when startled by noise. In the proximity evasion, the insect executes a sharp U-turn once the drone approaches within a 25 cm radius, simulating a close-range defensive maneuver. Importantly, when the virtual insect is intercepted by the drone the trials terminates. For each control method, five trials were executed for every Opogona replay trajectory, and ten trials were collected for the figure-eight and ten for the virtual insect. Performance was evaluated using the same metrics defined in Section V-A.

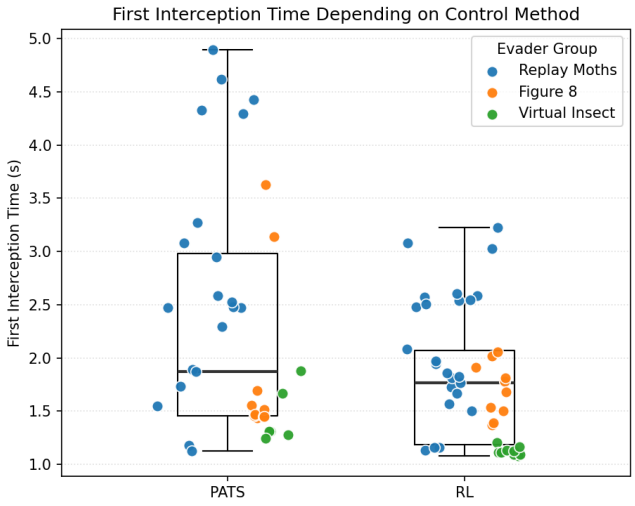


Fig. 12. First interception time for PATS and RL controllers across all evaders. Box plots show the per controller distributions. Points are individual logs, colored by trajectory. Only trials where interception was achieved are included.

B. Results

The overall interception success rate per controller and evader type is presented in Table VI. Across all engagements, the RL controller achieved a higher interception rate than the baseline PATS controller, 95.6% versus 80.0%. An overview of the interception performance metrics comparing the two controllers is given in Table V. In addition to intercepting more often, RL reached the target faster on successful runs. The overall median first–interception time was $1.769\ [1.185, 2.070]\,\mathrm{s}$ for RL versus $1.875\ [1.455, 2.982]\,\mathrm{s}$ for PATS. A visualization of the distribution of the first-interception-times is given in Fig. 12.

Performance varies systematically with engagement difficulty: differences in initial separation, the time at which the evader enters the observable frustum, and the evader's motion class all shift the expected first–interception time. To provide a fair estimate of controller effects with this heterogeneity, we summarize performance per trajectory using within-trajectory medians and then compare controllers via a paired

Method	t_{int} [s]	t_{NM} [%]	d_{mean} [m]
PATS	1.86 [1.46–2.98]	**26.40** [9.20–35.04]	**0.47** [0.42–0.60]
RL	**1.77** [1.19–2.07]	25.14 [15.63–36.04]	0.50 [0.45–0.58]
HL (RL – PATS)	$-0.701[-1.56,\ 0.24]$	$+2.07[-15.95,\ 16.15]$	$+0.03[-0.16,\ 0.10]$

TABLE V

PERFORMANCE METRICS OVER SUCCESSFUL RUNS FOR THE PATS AND RL CONTROLLER. THE BOTTOM ROW REPORTS THE PAIRED HODGES-LEHMANN EFFECT (RL-PATS) WITH A 95% BOOTSTRAP CI.

Hodges–Lehmann estimator across trajectories. This analysis indicates a typical RL speedup of $-0.701\,\text{s}$ with a bootstrap 95% CI $[-1.558,\ 0.239]$, indicating a consistent trend toward faster intercepts, albeit with uncertainty that spans zero. More trajectories would be needed to tighten the interval and make a stronger conclusion about the performance.

Consistent with this variability, several evaders proved especially difficult. Within the replayed Opogona set, replay trajectory 5 was notably hard; the baseline PATS controller intercepted 2 of 5 trials, while the RL controller intercepted 4 of 5. The reactive virtual insect also posed a challenge, with PATS achieving 6 of 10 interceptions and RL achieving 9 of 10. Thus, under harder engagements RL maintains feasibility more consistently.

Method	$\mathbf{p}_{\text{int}}$ [%]	Replay Moths	Figure-8	Virtual Insect
PATS	80.0	20/25	**10/10**	6/10
RL	**95.6**	**24/25**	**10/10**	**9/10**

TABLE VI

INTERCEPTION SUCCESS OF PATS AND RL CONTROLLERS FOR THE DIFFERENT EVADER TYPES.

Secondary kinematic proximity metrics show only small differences and do not change the main conclusions. Time spent in near–miss conditions is similar: the PATS controller reports $26.4\ [9.2, 35.0]\%$ and the RL controller $25.1\ [15.6, 36.0]\%$. The paired Hodges–Lehmann effect for near–miss time indicates no clear systematic difference. Similarly the mean distance between the drone and target are comparable for both controllers.

Taken together, these results meet the goal of this implementation, to demonstrate a functional integration of a learned guidance policy within the existing PATS-X stack. The statistical basis is limited; the number of trials per evader type is small. There is also substantial variation across repeated trials. Even so, the evidence points to a meaningful practical benefit. The RL controller achieves a higher overall interception rate and shows a trend toward faster first–interception times. Thus, as a proof of concept, the study validates both the integration path and the promise of RL for PATS-X, motivating further development.

This paper therefore benchmarks classical and learned approaches for aerial-to-aerial interception in the PATS-X context. The best practices are identified for the PATS-X system based on interception performance in simulation and transferability to the physical system.

VII. LIMITATIONS & FURTHER RESEARCH

This study set out to benchmark classical and learned approaches for aerial-to-aerial interception in the PATS-X context. While we demonstrate strong simulation performance and successful CyberZoo deployments at the CTBR and motor abstraction levels, several limitations constrain the generality of the results and suggest concrete avenues for further work. To start, in the simulation study, all targets were effectively passive. Real pests exhibit reactive, stimulus-driven behaviors and may actively evade the pursuer. A natural extension is to model both pest and pursuer as learning agents in a multi-agent RL set-up, as is done in a prior work by Vos [32], and then re-establish both the benchmark metrics and the best practices under closed-loop interaction.

A further simulation limitation is that, although the RL controllers outperform the FRPN baseline, they operate as black-box policies. Gaining insight into their interception strategies would clarify which geometric or kinematic cues drive successful captures and could inform refinements of existing classical guidance. In particular, it would be valuable to analyze learned behavior through the lens of pursuit-strategy taxonomies and fit metrics such as those proposed by Wei et al. [39], to determine where the policies lie within fundamental pursuit classes and how those principles might be transferred back to classical controllers.

Crossing the reality gap on hardware exposes additional limitations. Deployment under the acceleration abstraction led to frequent crashes. For safety and to avoid hardware damage, we discontinued real-world testing at this level. Within the time frame, the transfer failures could not be resolved. This has two consequences. First, although the FRPN comparison is well established in simulation, a real-world evaluation with the RL acceleration controller was not possible. Second, the omission of the acceleration controller from the experiments on the interplay between abstraction level, domain randomization, and reward smoothing weakens those conclusions. Including this level would have strengthened sensitivity analyses and best-practice recommendations for physical deployment.

To add, the Bebop 2 is an imperfect platform for assessing end-to-end control. Excessive model mismatch, potentially due to airframe flexibility and a less capable IMU leading to weaker state estimation, undercuts the merits of motor-level policies here. Using a more agile drone-racing platform may allow the benefits of end-to-end control to manifest when

crossing the reality gap.

Furthermore, aerial-to-aerial interception is a high-dimensional problem, which complicates controlled evaluation of reality-gap methods. It would be more informative to assess domain randomization and reward smoothing on a task with higher repeatability that still demands agile flight. The agile requirement is important; tracking a slow reference or hovering about a fixed point involves far less nonlinear dynamics and will likely exhibit a smaller reality gap by construction. Drone racing is a suitable candidate for systematically probing the interplay of abstraction level, domain randomization, and reward shaping, and is an avenue for future work.

Within this same reality-gap context, the network class used here is another limitation. The present study uses feed-forward MLPs. More involved architectures that ingest causal sequences are promising. Recent work by Eschmann et al. [38] shows that such networks can facilitate in-context learning for zero-shot transfer to unknown quadcopter platforms. Adopting this approach offers two advantages: zero-shot transfer to new platforms without separate system identification and re-training, and avoidance of the main shortcoming of domain randomization, namely that the policy must select actions viable across all randomized models. With in-context learning, the policy can infer platform dynamics on-the-fly and choose actions that are optimal for the specific platform at deployment. An avenue for future research would be to apply this novel method of overcoming the reality gap to aerial-to-aerial RL interception controllers.

Finally, the PATS-X integration is a proof of concept that leaves room for improvement. Boundary enforcement is implemented via reward shaping as explained in Appendix III, which acts as a soft constraint and does not guarantee that commanded actuation will never drive the vehicle out of bounds. Moreover, training assumed a single boundary configuration; greenhouse deployments can vary substantially in obstacles and admissible volumes, implying that boundary handling should generalize across layouts. A practical next step is to replace soft shaping with constraint-aware control integrated into the RL training and to train across a family of frusta or obstacle layouts so that boundary handling generalizes across deployments.

The benefits of lower abstraction levels could not be evaluated on PATS-X due to limited telemetry bandwidth to the offboard computer. The same limitation prevented system identification for the PATS drone, because telemetry updates are slower than the platform's response to commands. Adding an onboard logging module would enable higher-rate data capture for identification, bypassing telemetry. For lower control abstractions, either telemetry must be increased substantially or control must be moved onboard the vehicle.

VIII. Conclusion

This work systematically examined guidance and control for aerial-to-aerial interception in the context of pest neutralization, comparing the classical state-of-the-art FRPN controller with newly developed RL policies. In simulation, RL consistently surpassed FRPN in reliability and proximity control, with motor-level policies achieving the fastest and most consistent interceptions, followed by CTBR and then acceleration. The best motor-level policy achieved a median first-interception time of 0.85 $[0.76\text{-}1.07]\,\mathrm{s}$, compared to FRPN at $1.90\,\mathrm{s}$ $[1.04\text{-}2.80]\,\mathrm{s}$; this corresponds to a median speedup of $1.05\,\mathrm{s}$, together with higher interception reliability of 99.1% compared to 95.6%. Observation design was decisive. Representing relative position and velocity in the body frame yielded the strongest performance. Adding action histories did not improve results. Short observation histories primarily acted as noise filters when explicit filtering was absent; otherwise, they had no measurable effect. However as the discussion on limitations indicated, all results are due to testing with a passive evader, implementing a reactive evader or a more involved multi-agent RL set-up will require a re-examination of the best practices.

Crossing the reality gap, only the CTBR policies transferred directly. Domain randomization and reward smoothing has no merit for the CTBR abstraction level. The CTBR abstraction level is a sweet spot for RL control: it is expressive enough for the policy to execute the complex maneuvers required for interception, yet abstract enough to transfer easily because low-level stabilization is handled by the inner-loop controllers. A modest amount of domain randomization 10% enabled transfer at the motor abstraction level; any more randomization only worsened performance. Throughout all deployments at the motor abstraction level, performance showed a large gap relative to simulation results, suggesting a significant mismatch with the dynamic model used in simulation. Future work should use a drone better suited to direct motor control, for example a racing drone rather than the hobby-grade Bebop 2. Reward smoothing was beneficial for the motor abstraction level, particularly by allowing trials to be performed without risking overheating the motors and slight smoothing improving performance. However, the motor-level results suggest that too much smoothing worsens performance and decreases flight stability.

The acceleration abstraction level did not transfer at all; however, this is most likely due to limitations of the Bebop 2 and the implementation method, rather than limitations of the abstraction level itself. Notably, the acceleration abstraction did transfer on the PATS-X system, indicating that the level itself is viable when the platform and integration pipeline support it. Prompted by the discussion of the limitations, future research should select a platform to validate RL results at the acceleration level and evaluate methods to overcome the reality gap on a task that better supports repeatable testing.

A proof-of-concept integration on the PATS-X platform showed that the RL controller could be successfully deployed on the real system, a notable result given the sim-to-real gap challenges. Building on that, the integrated RL controller demonstrated higher interception rates, higher interception rates, achieving 95.6% compared to 80.0% for the baseline. RL shows a trend toward faster first–interception times than the LPN–style baseline. A paired Hodges-Lehmann

estimator indicates a typical speedup of $0.701\,\mathrm{s}$ (95% CI $[-1.558, 0.239]$). Because the confidence interval spans zero, additional trials are required to establish a statistically reliable advantage. The RL controller is able to account for the detection system's FOV; however, no guarantee of adherence can be given, as only a soft boundary is imposed. Nevertheless, the potential of RL within the PATS-X framework is established and suggests that future development should focus on better facilitating RL controllers. An initial step would be to include an onboard logging module for improved system identification and to increase the rate at which telemetry is sent to the offboard controller, enabling RL guidance and control at lower abstraction levels.

References

[1] Y. Yang, D. Tilman, Z. Jin, P. Smith, C. B. Barrett, Y.-G. Zhu, J. Burney, P. D'Odorico, P. Fantke, J. Fargione *et al.*, "Climate change exacerbates the environmental impacts of agriculture," *Science*, vol. 385, no. 6713, 2024.

[2] R. Guebsi, S. Mami, and K. Chokmani, "Drones in precision agriculture: A comprehensive review of applications, technologies, and challenges," *Drones*, vol. 8, no. 11, p. 686, 2024.

[3] PATS Indoor Drone Solutions, "Pats-x," https://www.pats-drones.com/patsx, 2025, accessed: 2025-04-05.

[4] N. F. Palumbo, R. A. Blauwkamp, and J. M. Lloyd, "Modern homing missile guidance theory and techniques," Johns Hopkins APL, Tech. Rep., 2010.

[5] M. Mehrandezh, M. N. Sela, R. G. Fenton, and B. Benhabib, "Robotic interception of moving objects using ideal proportional navigation guidance technique," *Robotics and Autonomous Systems*, vol. 28, no. 4, pp. 295–310, 1999.

[6] C.-D. Yang and C.-C. Yang, "Optimal pure proportional navigation for maneuvering targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 3, pp. 949–957, 1997.

[7] D. Ghose, "True proportional navigation with maneuvering target," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 1, pp. 229–237, 1994.

[8] B. Zhu, A. H. B. Zaini, and L. Xie, "Distributed guidance for interception by using multiple rotary-wing unmanned aerial vehicles," in *IEEE Transactions on Industrial Electronics*, vol. 64. Institute of Electrical and Electronics Engineers Inc., 7 2017, pp. 5648–5656.

[9] M. Pliska, M. Vrba, T. Báča, and M. Saska, "Towards safe mid-air drone interception: Strategies for tracking & capture," *IEEE Robotics and Automation Letters*, 2024.

[10] W. Li, Y. Zhu, and D. Zhao, "Missile guidance with assisted deep reinforcement learning for head-on interception of maneuvering target," *Complex and Intelligent Systems*, vol. 8, pp. 1205–1216, 4 2022.

[11] S. He, H.-S. Shin, and A. Tsourdos, "Computational missile guidance: A deep reinforcement learning approach," *Journal of Aerospace Information Systems*, vol. 18, no. 8, pp. 571–582, 2021.

[12] W. Wang, M. Wu, Z. Chen, and X. Liu, "Integrated guidance-and-control design for three-dimensional interception based on deep-reinforcement learning," *Aerospace*, vol. 10, no. 2, p. 167, 2023.

[13] X. Qiu, C. Gao, and W. Jing, "Maneuvering penetration strategies of ballistic missiles based on deep reinforcement learning," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 236, no. 16, pp. 3494–3504, 2022.

[14] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[15] J. Eschmann, D. Albani, and G. Loianno, "Learning to fly in seconds," 2024. [Online]. Available: https://arxiv.org/abs/2311.13081

[16] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 504–10 510.

[17] R. Ferede, C. De Wagter, D. Izzo, and G. C. De Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6172–6177.

[18] B. Gaudet, R. Furfaro, and R. Linares, "Reinforcement learning for angle-only intercept guidance of maneuvering targets," *Aerospace Science and Technology*, vol. 99, 4 2020.

[19] M. Yan, R. Yang, X. Zhao, and L. Yue, "Ballistic missile midcourse intelligent maneuver strategy based on ppo algorithm," in *International Conference on Guidance, Navigation and Control*. Springer, 2022, pp. 5169–5178.

[20] J. Chen, C. Yu, G. Li, W. Tang, X. Yang, B. Xu, H. Yang, and Y. Wang, "Multi-uav pursuit-evasion with online planning in unknown environments by deep reinforcement learning," *arXiv preprint arXiv:2409.15866*, 2024.

[21] A. Dionigi, G. Costante, and G. Loianno, "The power of input: Benchmarking zero-shot sim-to-real transfer of reinforcement learning control policies for quadrotor control," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 11 812–11 818.

[22] S. Gronauer, D. Stümke, and K. Diepold, "Comparing quadrotor control policies for zero-shot reinforcement learning under uncertainty and partial observability," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 7508–7514.

[23] R. Ferede, T. Blaha, E. Lucassen, C. De Wagter, and G. C. de Croon, "One net to rule them all: Domain randomization in quadcopter racing across different platforms," *arXiv preprint arXiv:2504.21586*, 2025.

[24] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, 2021.

[25] J. Chen, C. Yu, Y. Xie, F. Gao, Y. Chen, S. Yu, W. Tang, S. Ji, M. Mu, Y. Wu *et al.*, "What matters in learning a zero-shot sim-to-real rl policy for quadrotor control? a comprehensive study," *IEEE Robotics and Automation Letters*, 2025.

[26] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada: IEEE, 2017, pp. 23–30. [Online]. Available: https://doi.org/10.1109/IROS.2017.8202133

[28] E. Valassakis, Z. Ding, and E. Johns, "Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5372–5379.

[29] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 59–66.

[30] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1810–1816.

[31] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[32] R. Vos, "Hunt like a dragonfly and strike like a drone: Simulating games of pursuit and evasion to optimize quadcopter control for insect pest interception in greenhouses," Master's thesis, Delft University of Technology, Delft, The Netherlands, October 2024, master's thesis, Faculty of Aerospace Engineering. [Online]. Available: http://repository.tudelft.nl/

[33] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[35] B. Gati, "Open source autopilot for academic research-the paparazzi

system," in *2013 American Control Conference*. IEEE, 2013, pp. 1478–1481.
[36] C.-D. Yang and C.-C. Yang, "A unified approach to proportional navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 2, pp. 557–567, 1997.
[37] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
[38] J. Eschmann, D. Albani, and G. Loianno, "Raptor: A foundation policy for quadrotor control," *arXiv preprint arXiv:2509.11481*, 2025.
[39] E. Wei *et al.*, "Pursuit and an evolutionary game," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 465, no. 2105, pp. 1539–1559, 2009.
[40] E. J. Smeur, Q. Chu, and G. C. De Croon, "Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 3, pp. 450–461, 2016.
[41] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.
[42] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," *Neurocomputing*, vol. 497, pp. 64–75, 2022.
[43] J. Eschmann, "Reward function design in reinforcement learning," *Reinforcement learning algorithms: Analysis and Applications*, pp. 25–33, 2021.

APPENDIX I
DYNAMIC MODELS

This appendix expands Section III-B and Fig. 3 by giving the full transition models for the three control abstraction levels $\kappa \in \{\text{Motor}, \text{CTBR}, \text{Accel}\}$ used in the main text. For each level we specify the state composition x, and the continuous-time dynamics $\dot{x} = f_\kappa(x, a_\kappa)$ together with the simplifications used to obtain a simulator that is lightweight enough for RL rollouts but still captures the relevant MAV response and inner-loop controller behavior.

A. Motor

The motor abstraction model is taken from [23]. The policy outputs an action vector $a_t \in [-1, 1]^4$, which is linearly rescaled element-wise to motor inputs by

$$u_i \;=\; \frac{a_{t,i}+1}{2}, \qquad i = 1, \dots, 4.$$

The state and input are,

$$x = \begin{bmatrix} p_d,\ v_d,\ \lambda,\ \Omega,\ \omega \end{bmatrix}^T, \qquad u = \begin{bmatrix} u_1, u_2, u_3, u_4 \end{bmatrix}^T, \quad (8)$$

where $p_d \in \mathbb{R}^3$ is the MAV position, $v_d \in \mathbb{R}^3$ velocity, $\lambda \in \mathbb{R}^3$ the Euler angles, $\Omega \in \mathbb{R}^3$ body rates, and $\omega \in \mathbb{R}^4$ the rotor angular speeds. The continuous-time equations of motion are,

$$\begin{aligned} \dot{p}_d &= v_d, & \dot{v}_d &= g\, e_3 + R(\lambda)\, F, \\ \dot{\lambda} &= Q(\lambda)\, \Omega, & \dot{\Omega} &= M, \\ \dot{\omega}_i &= \big(\bar{\omega}_{ci} - \omega_i\big)/\tau_\omega, \quad i = 1, \dots, 4. \end{aligned} \quad (9)$$

where $R(\lambda)$ maps body to world coordinates, $Q(\lambda)$ transforms body rates to Euler-angle derivatives, $e_3 = [0\ 0\ 1]^T$, and $\tau_\omega > 0$ is the motor reaction time constant. The commanded steady-state motor speed is modeled by

$$\bar{\omega}_{ci} = (\omega_{\max} - \omega_{\min}) \sqrt{k_t\, u_i^2 + (1 - k_t)\, u_i} \;+\; \omega_{\min}, \quad (10)$$

with $k_t \in [0, 1]$ shaping the thrust curve and $\omega_{\min}, \omega_{\max}$ the idle and saturation speeds. The specific force applied by the motors $F \in \mathbb{R}^3$ and the angular acceleration $M \in \mathbb{R}^3$ are parameterized as,

$$F = \begin{bmatrix} -\sum_{i=1}^4 k_x\, v_x^B\, \omega_i \\ -\sum_{i=1}^4 k_y\, v_y^B\, \omega_i \\ -\sum_{i=1}^4 k_\omega\, \omega_i^2 \end{bmatrix} \quad (11a)$$

$$M = \begin{bmatrix} -k_{p1}\omega_1^2 - k_{p2}\omega_2^2 + k_{p3}\omega_3^2 + k_{p4}\omega_4^2 \\ -k_{q1}\omega_1^2 + k_{q2}\omega_2^2 - k_{q3}\omega_3^2 + k_{q4}\omega_4^2 \\ -k_{r1}\omega_1 + k_{r2}\omega_2 + k_{r3}\omega_3 - k_{r4}\omega_4 + \\ \dots - k_{r5}\dot{\omega}_1 + k_{r6}\dot{\omega}_2 + k_{r7}\dot{\omega}_3 - k_{r8}\dot{\omega}_4 \end{bmatrix} \quad (11b)$$

where v_x^B, v_y^B are the body-frame components of v. The final row of $\mathbf{M}$ admits additional linear terms to capture the inertial effect of the propellors.

B. CTBR

The CTBR abstraction policy issues collective-thrust and body-rate setpoints. The model presented here is taken from [17]. The same kinematic blocks $R(\lambda)$ and $Q(\lambda)$ used in Eq. 9 are retained, but the actuation layer is changed by assuming an incremental non-linear dynamic inversion (INDI) inner loop that tracks the commanded body-rates and thrust. As shown in [40], the INDI controller's response to set-points may be modeled as a first-order delay. This reduces fidelity relative to the motor abstraction by removing the rotor speed state and the detailed moment map, and by lumping reaction and gyroscopic effects into simple lags.

The policy outputs an action vector $a_t \in [-1, 1]^4$, split into rate and thrust channels as,

$$a_t \;=\; \begin{bmatrix} (a_t^\Omega)^T & a_t^T \end{bmatrix}^T, \qquad a_t^\Omega \;=\; \begin{bmatrix} a_{t,p} \\ a_{t,q} \\ a_{t,r} \end{bmatrix}, \quad (12)$$

where, a_t^Ω are the three body-rate channels and a_t^T is the thrust channel, all normalized to $[-1, 1]$. These actions are linearly rescaled to platform-specific commands using the actuator limits $p_{\max}, q_{\max}, r_{\max}, T_{\max}, T_{\min}$,

$$\Omega_{\text{cmd}} = \begin{bmatrix} p_{\max} & 0 & 0 \\ 0 & q_{\max} & 0 \\ 0 & 0 & r_{\max} \end{bmatrix} \mathbf{a}_t^\Omega, \quad (13)$$

$$T_{\text{cmd}} = \tfrac{1}{2}(T_{\max} - T_{\min})\, a_t^T + \tfrac{1}{2}(T_{\max} + T_{\min}). \quad (14)$$

Here, Ω_{cmd} are the commanded body rates and T_{cmd} is the commanded specific thrust along the body z-axis. The state and input to the CTBR model is,

$$x = \begin{bmatrix} p_d,\ v_d,\ \lambda,\ \Omega,\ T \end{bmatrix}^T, \qquad u = \begin{bmatrix} \Omega_{\text{cmd}},\ T_{\text{cmd}} \end{bmatrix}^T, \quad (15)$$

where T denotes specific thrust along the body z-axis, T_{cmd} the commanded specific thrust, and $\Omega_{cmd} \in R^3$ the commanded angular rates. The continuous–time dynamics are,

$$\begin{aligned} \dot{p}_d &= v_d, & \dot{v}_d &= g\, e_3 + R(\lambda)\, F^B, \\ \dot{\lambda} &= Q(\lambda)\, \Omega, & \dot{\Omega} &= \operatorname{diag}\!\big(\tfrac{1}{\tau_p}, \tfrac{1}{\tau_q}, \tfrac{1}{\tau_r}\big)\, (\Omega_{\text{cmd}} - \Omega), \\ \dot{T} &= \tfrac{1}{\tau_T}\, (T_{\text{cmd}} - T), \end{aligned} \quad (16)$$

with a simplified approximation of specific force in the body frame,

$$F^B = \begin{bmatrix} -d_x\, v_x^B \\ -d_y\, v_y^B \\ -T \end{bmatrix}, \tag{17}$$

where aerodynamic effects are simplified to linear in–plane drag with coefficients d_x and d_y. The lateral body frame velocity is denoted through v_x^B and v_y^B.

C. Acceleration

The acceleration abstraction model exposes an outer interface for which the policy output is a normalized world–frame specific acceleration $a_t \in [-1,1]^3$. This action is scaled by the platform's maximum specific acceleration $a_{\max}$ to form a_{cmd}. Because the vehicle cannot track a_{cmd} directly, a pipeline converts this command into feasible setpoints and tracked inputs using three stages: an incremental nonlinear dynamic inversion (INDI) allocator that maps apparent–acceleration errors to attitude–thrust setpoints, a PD attitude loop that outputs body–rate commands, and the same first–order response approximators for body rates and thrust as used in the previously defined CTBR model.

Let $a_m = \dot{v}$ denote the measured acceleration, and let $\mathrm{clip}(\bullet;\, \underline{\bullet}, \overline{\bullet})$ denote elementwise clipping. A clipped acceleration increment is computed as

$$\delta a \;=\; \mathrm{clip}\big(a_{\mathrm{cmd}} - a_m;\; \delta a_{\min},\, \delta a_{\max}\big). \tag{18}$$

Here, $\delta a_{\min}, \delta a_{\max} \in \mathbb{R}^3$ are the per–axis bounds on the increment δa per step to adhere to INDI linearization assumptions. The acceleration increment is passed through an INDI allocator. Using the local control effectiveness matrix $G_a(\lambda, T_{\mathrm{est}})$, the attitude–thrust increments satisfy

$$\delta u_v \;=\; \big[\delta\theta,\; \delta\phi,\; \delta T\big]^T \;=\; G_a(\lambda, T_{\mathrm{est}})^{-1}\, \delta a. \tag{19}$$

Here, δu_v stacks the attitude and thrust increments. $G_a(\lambda, T_{\mathrm{est}}) \in \mathbb{R}^{3\times 3}$ is the control–effectiveness matrix that maps small (θ, ϕ, T) changes to acceleration changes at the current Euler angles $\lambda = [\phi\ \theta\ \psi]^T$ and thrust estimate T_{est}. The control effectiveness matrix is defined as,

$$G_a \;=\; \begin{bmatrix} c_\theta c_\phi T_{\mathrm{est}} & -s_\theta s_\phi T_{\mathrm{est}} & s_\theta c_\phi \\ 0 & -c_\phi T_{\mathrm{est}} & -s_\phi \\ -s_\theta c_\phi T_{\mathrm{est}} & -c_\theta s_\phi T_{\mathrm{est}} & c_\theta c_\phi \end{bmatrix}, \tag{20}$$

where, T_{est} is the local thrust estimate taken as $-g$. The resulting setpoints are

$$\theta_{\mathrm{sp}} = \mathrm{clip}\big(\theta + \delta\theta;\; -\theta_{\max},\, \theta_{\max}\big), \tag{21}$$
$$\phi_{\mathrm{sp}} = \mathrm{clip}\big(\phi + \delta\phi;\; -\phi_{\max},\, \phi_{\max}\big), \tag{22}$$
$$\psi_{\mathrm{sp}} = 0, \tag{23}$$
$$T_{\mathrm{sp}} = \mathrm{clip}\big(T + k_T\, \delta T;\; T_{\min},\, T_{\max}\big). \tag{24}$$

Here, where $\theta_{\max}, \phi_{\max}, T_{\min}, T_{\max}$ are respective platform specific limits. The yaw angle is left unactuated, as it is held to track zero throughout the engagement. A diagonal PD controller maps attitude errors to rate commands,

$$\Omega_{\mathrm{cmd}} = \mathrm{clip}\Big(K_p(\lambda_{\mathrm{sp}} - \lambda) - K_d\,\Omega;\, -\Omega_{max},\, \Omega_{max}\Big), \tag{25}$$
$$\lambda_{\mathrm{sp}} = \big[\phi_{\mathrm{sp}}\ \theta_{\mathrm{sp}}\ \psi_{\mathrm{sp}}\big]^T. \tag{26}$$

where, $K_p = \mathrm{diag}(k_{p,\phi}, k_{p,\theta}, k_{p,\psi})$ and $K_d = \mathrm{diag}(k_{d,\phi}, k_{d,\theta}, k_{d,\psi})$ are the proportional and derivative gains, respectively. $\Omega_{max} \in R^3$ are per–axis rate limits which is platform dependent.

The thrust setpoint from the INDI allocation is directly set as the command, $T_{\mathrm{cmd}} = T_{\mathrm{sp}}$. Body–rate and thrust response to commanded values are approximated with first–order lags. This is the same as was done for the CTBR abstraction model as given in Eq. 16.

Appendix II
Initialization

Initialization randomizes the initial geometry and states to expose the policy to a wide range of encounter conditions. This is necessary for insensitivity to different starting points and to post–interception resets, where the MAV may need to re–engage a new target from a dynamic, non–hovering state.

Let $\zeta \in \mathbb{R}^3$ be a random unit vector obtained by normalizing a 3-D Gaussian draw, and let $\xi_0 \sim \mathcal{U}(1,2)\,\mathrm{m}$ be a random initial standoff radius. For a given center $p_{d_{\mathrm{init}}}$,

$$p_{d0} \;=\; p_{d_{\mathrm{init}}} + \xi_0 \cdot \zeta, \qquad v_{d0} \sim \mathcal{U}\big([-0.5, 0.5]^3\big)\ \mathrm{m/s}.$$

Euler angles and body rates are initialized as,

$$\lambda_0 = \begin{bmatrix} \phi_0 \\ \theta_0 \\ \psi_0 \end{bmatrix}, \quad \phi_0, \theta_0 \sim \mathcal{U}\Big[-\tfrac{\pi}{4},\, \tfrac{\pi}{4}\Big], \quad \psi_0 \sim \mathcal{U}[-\pi, \pi],$$

$$\Omega_0 = \begin{bmatrix} p_0 \\ q_0 \\ r_0 \end{bmatrix} \sim \mathcal{U}\big([-2.0, 2.0]^3\big)\ \mathrm{rad/s}.$$

For the Motor abstraction, rotor speeds are initialized by drawing $w_0 \sim \mathcal{U}([w_{min}, w_{max}]^4)$. For CTBR and acceleration models, the specific thrust is initialized as $T_0 \sim \mathcal{U}(0, T_{\max})$.

Appendix III
Base Reward

The objective is to achieve interception in the shortest possible time. A reward that directly encodes this objective penalizes the agent for every timestep that elapses prior to interception and terminates the episode on interception. Let Δt be the simulation step, d_{int} the interception radius, and $d_t = |\Delta \mathbf{p}(t)|$ the distance between the MAV and pest where $\Delta p_t = \mathbf{p}_{e,t} - \mathbf{p}_{p,t}$. A time–optimal reward is,

$$r_t^{\mathrm{direct}} \;=\; -\,\Delta t\,\big(1 - \mathbf{1}_{\mathrm{int}}\big), \tag{27}$$

where $\mathbf{1}_{\mathrm{int}} = 1$ if $d_t < d_{\mathrm{int}}$, else 0. Maximizing $\sum_t r_t^{\mathrm{direct}}$ is equivalent to minimizing the expected capture time.

In practice this reward is extremely sparse, most early policy rollouts never lead to interception events. As a result,

the reward signal is almost always zero, giving the policy no intermediate guidance about which actions actually shorten time-to-capture or lead to an interception. Moreover, this reward structure suffers from poor "credit assignment". The credit assignment problem refers to the difficulty of assigning reward to the agent's actions [41]. Several factors make this problem hard for the specific task of aerial-to-aerial interception. The dynamics of the MAV are delayed, the effects of an action become clear only later, leading to a weaker learning signal. Noisy sensing makes the reward signal unreliable. Finally, as noted in a prior work by Reinier Vos [32], the independent motion of the target can lead to reward being attributed to the interceptor's actions when it actually results from coincidental target motion.

Motivated by these limitations, a reward is adopted here that serves as an imperfect proxy for time-optimal interception while providing step-wise credit for progress caused by the action's of the MAV. The per–step reward is defined as,

$$r_t^{\text{base}} \;=\; \frac{(d_{t-1} - d_t) - s_t}{d_0} \;+\; 10 \cdot \mathbf{1}_{\text{int}}, \tag{28}$$

where, Δd_t and $\mathbf{1}_{int}$ are as defined in Eq. 27. The evader's one-step path length is given through $s_t = |p_{e,t} - p_{e,t-1}|$. The initial range between the MAV and pest is given through Δd_0.

At each timestep, the reward equals the MAV's reduction in LOS range from the previous step minus the evader's distance traveled in that step, normalized by the initial separation, with a fixed bonus added when the capture condition is met. The step reward is positive when the MAV reduces the separation by more than the evader moves during that step. If this holds throughout an engagement, the separation decreases at every step and must cross the capture radius in finite time; sustained positive reward is therefore a sufficient condition for interception.

To reduce sparsity, we draw upon the reward shaping used in drone racing [14], [23]. In that setting, the agent receives per step credit for closing the distance to the next waypoint. We apply the same per step credit principle. At each timestep the MAV earns reward for its own per step reduction of range to the target, so progress toward interception is attributed continuously and remains aligned with the terminal objective. This links each step to the terminal objective and provides informative feedback throughout the episode.

Addressing the credit assignment problem, the reward draws upon the reward formulation of a prior work by Reinier Vos [32], where the reward is shaped to act as an imperfect proxy for the individual contribution of the agent to the reward. Inspired by his formulation, evader's per-step travel is subtracted from the LOS distance. This improves the credit assignment by attributing progress only to the MAV's own motion. For example, if the MAV is stationary while the pest drifts toward it, the raw range decreases, but the shaping term subtracts the pest's travel, yielding no reward; positive reward is granted only when the MAV's action produces closing beyond what the pest contributes.

A second benefit of subtracting the evader's per-step travel is that it adds an implicit cost each timestep, so longer episodes systematically accrue more penalty. Maximizing return therefore favors trajectories that reach capture in fewer steps, therefore a practical stimulant for time-optimal interception.

Finally, the capture bonus is set bigger than any total of the step-wise rewards that could be accumulated without intercepting. Therefore even the most favorable non-capturing episode remains inferior to any episode that achieves capture. In practice, this means the single terminal bonus dominates the return, ensuring the learner always prefers policies that actually intercept.

APPENDIX IV
BOUNDARY ADHERENCE REWARD

To keep the MAV within an admissible flight volume, the reward is augmented with a proximity penalty and a hard out-of-bounds sanction. The flight region is represented as a convex frustum defined through multiple planes. The signed distance is computed to each plane $s_i(p_{d,t})$, which is nonnegative inside the frustum. Following a prior boundary-shaping via distance-to-constraint penalty [42], a soft force-field penalty is applied only on the interior side of each plane and grows linearly within a small boundary margin $m > 0$,

$$\text{ramp}_i(p_{d,t}) \;=\; \text{clip}\left(\frac{m - s_i(p_{d,t})}{m},\, 0,\, 1\right) \tag{29}$$

and is zero otherwise. The per–step boundary activation is then the maximum ramp across all boundary defining planes, and the soft penalty subtracted from the reward is,

$$R_{\text{bnd}}(p_{d,t}) \;=\; \gamma_{\text{bnd}} \;\max_i \text{ramp}_i(p_{d,t}), \tag{30}$$

where γ_{bnd} is a term that scales the penalty. Crossing a boundary triggers an immediate episode termination with a hard penalty γ_{OOB}, as done in several out-of-bounds schemes for drone control problems [23], [20].

To mitigate "learning to terminate", which is a destructive behavior where the agent discovers that ending an episode early gives higher return over a short horizon [43], the interception aligned base reward r_t^{smooth} is gated by the evader being in bounds. The base reward r_t^{base} is applied only if the evader is inside the inner margin, that is if all signed distances to the boundary planes satisfy $s_i(p_{e,t}) \geq m$. Otherwise the base reward is set to zero and replaced by a small waiting reward that decays with more control effort,

$$R_{\text{wait}} \;=\; \gamma_{\text{wait0}} + \gamma_{\text{wait1}}\, \exp\!\big(-\beta\, \|a_t\|^2\big), \tag{31}$$

where a_t is the pursuer action at time t and $\gamma_{\text{wait0}}, \gamma_{\text{wait1}}$, and $\beta > 0$ are constants defined in Table VII.

m	0.20	γ_{bnd}	0.25	γ_{OOB}	5.0
γ_{wait0}	0.001	γ_{wait1}	0.01	β	2.0

TABLE VII
HYPERPARAMETER VALUES FOR BOUNDARY ADHERENCE.

This gating addresses the failure mode that occurs when the evader is out of bounds. The pursuer cannot physically capture the target and, because the base reward subtracts the evader one step path length $s_t = |p_{e,t} - p_{e,t-1}|$, each step tends to yield negative reward while there is no opportunity to earn the positive capture bonus. In those circumstances an unconstrained learner may decide that ending the episode early is preferable. The shaped step return is

$$r_t = \mathbf{1}_{\text{ev-in}}\, r_t^{\text{base}} + r_t^{\text{smooth}} + \big(1 - \mathbf{1}_{\text{ev-in}}\big)\, R_{\text{wait}} - R_{\text{bnd}}(p_t) \;-\; \gamma_{\text{OOB}} \cdot \mathbf{1}_{\text{OOB}}. \tag{32}$$

where $\mathbf{1}_{\text{ev-in}} \in \{0, 1\}$ indicates that the evader is inside the inner margin, $\mathbf{1}_{\text{OOB}} \in \{0, 1\}$ flags an out of bounds pursuer, $R_{\text{bnd}}(p_t)$ is the soft boundary penalty at pursuer position p_t, and γ_{OOB} is the hard termination penalty. In words, the interior force field penalizes proximity to walls, the base interception reward is active only when the evader is legitimately in bounds, a low effort waiting term encourages holding position when the evader is outside, and any boundary violation produces an immediate penalty of γ_{OOB} and ends the episode. This shaping preserves the time to capture objective while discouraging crossing the boundaries.

APPENDIX V
PATS-X SET-UP

A. Sensors & State Estimation

PATS-X senses both the drone and airborne targets with an Intel RealSense depth camera and derives a real-time state stream at roughly 90 Hz for offboard guidance. Raw 3D detections are smoothed with a short moving-average filter; velocities are obtained by first-order backward-Euler differentiation of the smoothed positions and then low-pass filtered.

B. Actuation Interface

Acceleration is the interface through which the drone is actuated. Acceleration set-points are decomposed into attitude and thrust set-points which are issued offboard. Onboard, an in-house built flight controller tracks these set-points.

C. Flight Volume

Because state estimation is vision-based, the aircraft must remain within the camera field of view. The admissible flight region is therefore represented as a convex frustum constructed from the camera's FOV, orientation, and depth limits. During flight, this frustum is used to test feasibility of candidate maneuvers and, when needed, to reshape trajectories so that the vehicle remains observable. See in Fig. 1 a visualization of an engagement within the observable frustrum.

D. Baseline Guidance

PATS-X's baseline guidance is a LPN-style command consistent with Eq. 6, augmented in two practical ways. First, instead of the heuristic $t_{\text{go}} = \|\Delta p\|/\|\Delta v\|$, the system solves for t_{go} iteratively. For a candidate t_{go}, the target is extrapolated to,

$$p_e^{\text{pred}} \;=\; p_{e,t} \;+\; v_{e,t}\, t_{\text{go}},$$

and the constant-acceleration maneuver that brings the drone from (p_d, v_d) to p_e^{pred} in time t_{go} is computed as,

$$a_p^{\star}(t_{\text{go}}) \;=\; \frac{2\Big(p_e^{\text{pred}} - p_d - v_d\, t_{\text{go}}\Big)}{t_{\text{go}}^2}.$$

A simple search algorithm updates t_{go} until $\|a^{\star}\|$ falls within the platforms feasible flight envelope. This produces a self-consistent time-to-go that respects actuation limits. Second, boundary gating is layered on the command; if the resulting acceleration would carry the drone outside the frustum, the most constraining plane is identified and an intermediate via-point just inside the volume is inserted; a PID position feedback loop tracks this via-point to produce bounded acceleration commands while continuing toward interception.

Parameter	Value
$\hat{\tau}_{p,q,r,T}$ (s)	0.05
Rate limits p, q, r (rad/s)	$[-10,\ 10]$
Specific thrust range T (m/s^2)	$[0.0,\ 18.4]$
$k_x,\ k_y$ (—)	0.10
Max bank angle (deg)	unbounded
$\Delta a_{\min}$ (m/s^2)	$[-6.00,\ -6.00,\ -9.00]$
$\Delta a_{\max}$ (m/s^2)	$[6.00,\ 6.00,\ 9.00]$
$\mathbf{K_p}(\phi, \theta, \psi)$	$[20.0,\ 20.0,\ 20.0]$
$\mathbf{K_d}(\phi, \theta, \psi)$	$[1.0,\ 1.0,\ 1.0]$

TABLE VIII
ESTIMATED PATS-X MODEL PARAMETERS USED FOR RL POLICY TRAINING.

E. RL Deployment

For the RL experiments on PATS-X, only the acceleration command interface is available, so the policy was trained at the same abstraction level using the acceleration model in Section III-B. Due to the low-rate telemetry, high-fidelity system identification of the PATS hardware was not feasible; approximate parameters were estimated and are reported in Table VIII. Training used the Opogona dataset as defined in Section III-C and included the frustum planes so that boundary awareness is embedded in the policy. At deployment, the trained policy runs offboard in the intercept phase and, by design, respects the frustum without relying on PATS's classical boundary safety switches.

While simulation results presented in Section V-B identified relative features as best practice, enforcing frustum adherence requires absolute geometry. Accordingly, the observation set augments relative terms with inertial states, using $(p_d, p_e, v_d, v_e, \Delta p, \Delta v)$ computed from the preprocessed RealSense stream and without action or observation history. Because parameters were estimated, and not obtained via dedicated system identification, domain randomization was set to 30% during training to hedge model error. Reward smoothing was applied with γ_s=10 to curb command jitter.

APPENDIX VI
BEBOP 2 PARAMETERS

CTBR & Acceleration			
$[\hat{p}_{\min}, \hat{p}_{\max}]$ (rad/s)	$[-3.00,\ 3.00]$	$[\hat{q}_{\min}, \hat{q}_{\max}]$ (rad/s)	$[-3.00,\ 3.00]$
$[\hat{r}_{\min}, \hat{r}_{\max}]$ (rad/s)	$[-2.00,\ 2.00]$	$[\hat{T}_{\min}, \hat{T}_{\max}]$	$[1.41,\ 18.4]$
$\hat{g}$ (m/s^2)	9.81		
Acceleration			
$\hat{\phi}_{\max}$ (deg)	20.0	$\Delta a_{\min}$ (m/s^2)	$[-6.00,\ -6.00,\ -9.00]$
$\Delta a_{\max}$ (m/s^2)	$[6.00,\ 6.00,\ 9.00]$	$\hat{\mathbf{K}}_p\ (\phi, \theta, \psi)$	$[20.0,\ 20.0,\ 20.0]$
$\hat{\mathbf{K}}_d\ (\phi, \theta, \psi)$	$[1.0,\ 1.0,\ 1.0]$		

TABLE IX

ACTUATOR LIMITS, DESIGN PARAMETERS, AND CONSTANTS OF THE DIFFERENT ABSTRACTION LEVEL MODELS FOR THE BEBOP 2.

Motor			
$\hat{k}_\omega$	-3.70×10^{-6} ($\pm0.31\%$)	$\hat{k}_x$	-1.02×10^{-4} ($\pm5.85\%$)
$\hat{k}_y$	-1.05×10^{-4} ($\pm4.20\%$)	$\hat{\omega}_{\min}$ (rad/s)	2.99×10^{2} ($\pm0.78\%$)
$\hat{\omega}_{\max}$ (rad/s)	1.14×10^{3} ($\pm0.06\%$)	$\hat{k}_t$ (–)	1.00 ($\pm0.64\%$)
$\hat{\tau}_\omega$ (s)	6.00×10^{-2} ($\pm0.37\%$)	$\hat{k}_{p1}$	1.10×10^{-4} ($\pm0.88\%$)
$\hat{k}_{p2}$	-1.04×10^{-4} ($\pm1.10\%$)	$\hat{k}_{p3}$	-1.03×10^{-4} ($\pm0.92\%$)
$\hat{k}_{p4}$	1.06×10^{-4} ($\pm1.11\%$)	$\hat{k}_{q1}$	8.93×10^{-5} ($\pm0.77\%$)
$\hat{k}_{q2}$	8.70×10^{-5} ($\pm0.93\%$)	$\hat{k}_{q3}$	-8.29×10^{-5} ($\pm0.82\%$)
$\hat{k}_{q4}$	-8.95×10^{-5} ($\pm0.97\%$)	$\hat{k}_{r1}$	7.21×10^{-3} ($\pm9.96\%$)
$\hat{k}_{r2}$	-7.44×10^{-3} ($\pm7.34\%$)	$\hat{k}_{r3}$	1.07×10^{-2} ($\pm6.03\%$)
$\hat{k}_{r4}$	-7.80×10^{-3} ($\pm8.27\%$)	$\hat{k}_{r5}$	1.56×10^{-3} ($\pm2.04\%$)
$\hat{k}_{r6}$	-1.39×10^{-3} ($\pm2.47\%$)	$\hat{k}_{r7}$	1.50×10^{-3} ($\pm2.63\%$)
$\hat{k}_{r8}$	-1.31×10^{-3} ($\pm2.42\%$)		
CTBR & Acceleration			
$\hat{\tau}_p$ (s)	1.65×10^{-1} ($\pm2.17\%$)	$\hat{\tau}_q$ (s)	1.51×10^{-1} ($\pm2.19\%$)
$\hat{\tau}_r$ (s)	4.44×10^{-1} ($\pm5.43\%$)	$\hat{\tau}_T$ (s)	8.60×10^{-2} ($\pm4.67\%$)
$\hat{k}_x^{\text{ctbr}}$	3.10×10^{-1} ($\pm6.66\%$)	$\hat{k}_y^{\text{ctbr}}$	3.28×10^{-1} ($\pm4.67\%$)

TABLE X

IDENTIFIED PARAMETERS OF THE DIFFERENT ABSTRACTION LEVEL MODELS FOR THE BEBOP 2. 95% CONFIDENCE INTERVALS ARE SHOWN AS RELATIVE HALF-WIDTHS ($\pm\%$).

Part II

Literature Review

3

Introduction

The drastic effects of climate change have heightened the importance of sustainability, placing greater pressure on the agricultural sector. As a result, farmers have begun adopting innovative methods to increase yields while reducing environmental impact. At the same time, consumers have become more conscious of sustainability, driving a surge in demand for eco-friendly technologies [1].

Micro air vehicles (MAVs) have emerged as a versatile technology with applications spanning numerous industries, including agriculture. For example, MAV-based crop monitoring systems help reduce water usage, detect diseases early, and improve overall resource management [4]. In line with these trends, PATS has positioned itself as an innovative contributor to sustainable agriculture. Their overarching goal is to accelerate the transition toward autonomous greenhouses and more eco-friendly farming practices. A novel product under development is the PATS-X system, which uses an array of infrared cameras to detect and track pests. Once a pest is located, a MAV is dispatched to intercept and eliminate it, thereby reducing reliance on toxic and expensive pesticides [3].

As pest control remains a critical challenge in sustainable farming, technologies such as the PATS-X system are gaining attention for their potential to reduce dependency on chemical pesticides. Research by E.-C. Oerke [2] highlights the severity of pest-related losses, estimating global potential crop losses ranging from approximately 50% in wheat to a maximum of over 80% in cotton. Therefore leaving pests unchecked is detrimental to ones yield. Farmers must constantly balance the amount of pesticide which is economically and ecologically feasible.

However, realizing the PATS-X system presents several significant challenges. One of the most critical difficulties is the interception of the pests. Although pests are typically limited in their top speed, they exhibit exceptional agility, frequently changing direction with little warning. This agility makes them difficult targets for MAVs, which may be capable of flying faster but often lack the maneuverability needed to respond effectively. Additionally, most pests exhibit reactive behavior; when pursued, they actively engage in evasive maneuvers, further complicating interception.

Moreover, while the PATS-X camera system provides high-frequency updates of a pest's location, accurately predicting its future behavior remains a major challenge. The high maneuverability and erratic flight patterns of pests make state estimation difficult, which in turn complicates real-time control and interception. These difficulties are further amplified by real-world greenhouse conditions, where dense vegetation, structural obstacles, and narrow spaces offer pests ample opportunities for escape and concealment. Finally, since autonomous control strategies are often developed and validated in simulated environments, there is a critical need to ensure that these solutions transfer to the unpredictable and complex conditions of real-world deployment. To this end, the following main research question has been developed to guide the subsequent literature review,

Literature Review Research Question

What integrated guidance and control strategies are effective for MAV interception of agile pests, and what strategies are used to aid transition from simulation to deployment in real-world environments?

To effectively address the research question, this literature review is structured around the key components required for a reliable MAV-based pest interception guidance and control. Each chapter is motivated by a distinct challenge or opportunity inherent to aerial-to-aerial pest interception. An overview of a preliminary system architecture is provided in Figure 3.1, which serves as a foundation for identifying critical research areas.

The first chapter, chapter 4, addresses the need for a formal framework to model MAV-to-pest interception dynamics. It defines the core kinematic relationships governing aerial engagement and highlights the challenge of evaluating interception strategies—particularly when dealing with black-box controllers. To support systematic comparison, the chapter motivates the need for both geometric adherence metrics and task-level performance indicators.

The chapter 5 chapter tackles the problem of designing classical guidance laws suitable for MAV-based interception of agile aerial pests. Traditional proportional navigation (PN) methods were developed under assumptions that do not hold in MAV contexts—such as high interceptor speed, ideal actuation, and limited need for re-engagement. This chapter investigates whether PN-based strategies exist or can be adapted to meet the challenges of MAV-pest interception.

Chapter 6 addresses the limitations of classical guidance laws in handling the complex, reactive, and highly dynamic nature of MAV-to-pest interception. Fixed-form strategies like PN lack the adaptability needed for varying engagement phases, asymmetric maneuvering capabilities, and unpredictable pest behavior. This chapter motivates the need for learned control strategies that can overcome these constraints, and investigates what reinforcement learning (RL) algorithms and design consideration may be used to produce high-performance interception policies.

In chapter 7 a central barrier to real-world deployment of RL-based MAV interception controllers, the reality gap, is addressed. Policies trained in simulation often fail when transferred to physical hardware due to modeling inaccuracies. This problem is particularly acute for RL methods, which overfit to their training environments. The chapter explores how targeted design strategies, including system identification, domain randomization, and reward smoothing, can improve zero-shot transferability and mitigate sim-to-real degradation.

To synthesize the insights gained from the literature review, the final chapter 8 presents an integrated summary of key findings across all thematic areas—engagement kinematics, classical guidance, RL, and sim-to-real transfer. It revisits the overarching research question, consolidates lessons learned from each chapter, and clearly identifies the remaining gaps. Based on this synthesis, the conclusion motivates a targeted set of research questions and defines the central research objective of this thesis.

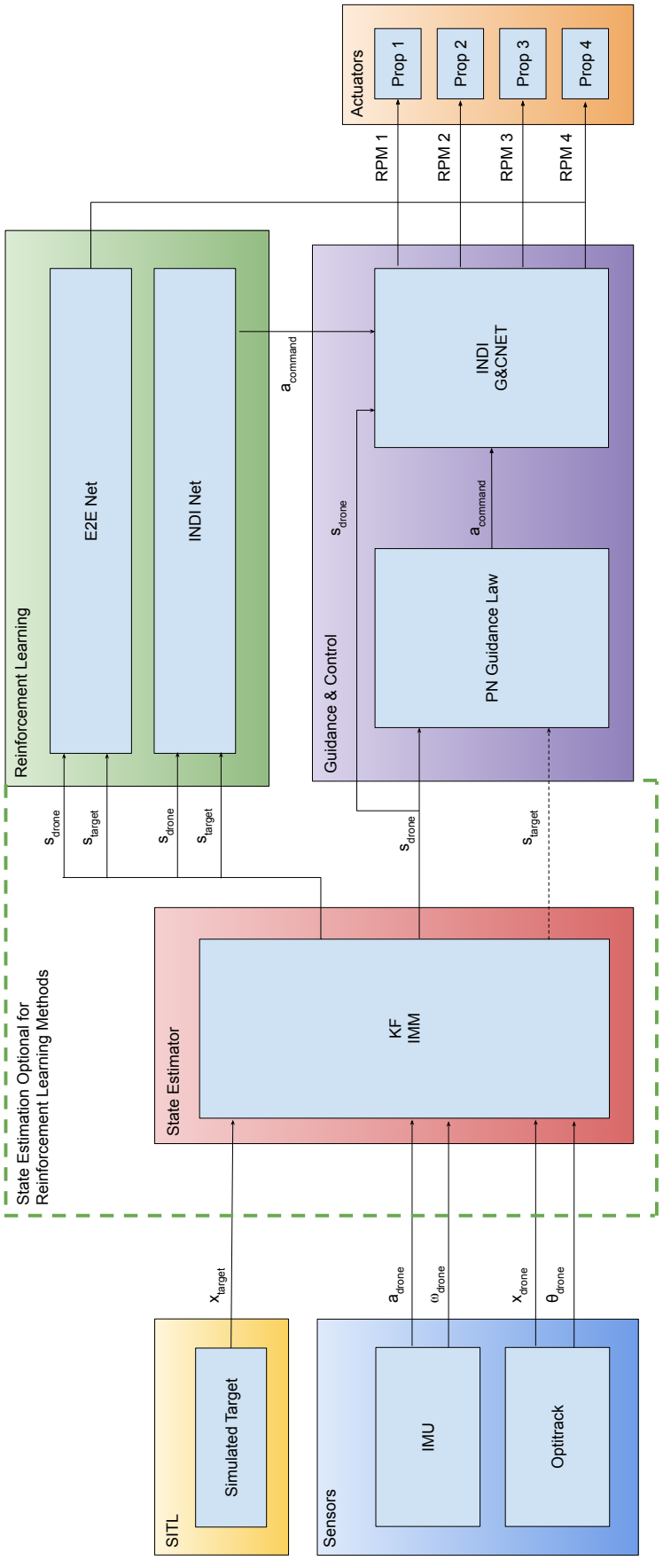


Figure 3.1: Overview of MAV system architecture options, featuring sensor inputs, state estimation, guidance and control networks, actuator management, and optional RL and software in the loop components.

4

Engagement Kinematics

The problem of pest interception with a MAV represents a challenging instance of the broader class of aerial-to-aerial interception scenarios. In this context, interception requires that a MAV successfully closes the gap and makes contact with a moving, evasive aerial target within a constrained environment. To design, evaluate, and compare guidance and control strategies for such interception tasks, it is essential to establish a formal kinematic framework that accurately describes the geometry and relative motion between the MAV (pursuer) and the pest (evader). A well-defined engagement model enables the derivation of guidance laws and the formulation of control objectives that are grounded in the physics of motion and the spatial relationship between agents. Without this foundation, it would be difficult to reason about trajectory feasibility, guidance behavior, or expected interception outcomes. Therefore, the following research question is posed,

Kinematic Modeling Research Question

What are the kinematic equations describing the relative motion between a MAV and an agile aerial pest during interception?

Equally important is the development of metrics that can quantitatively assess interception performance. For MAV-based pest control, it is not sufficient to determine whether interception occurs; one must also evaluate how it occurs. Quantitative performance indicators provide crucial insight into the quality and efficiency of a guidance and control solution. These metrics also allow for meaningful comparisons between classical and learned control approaches, particularly when the controller operates as a black box and its internal logic is not directly interpretable. To guide the development of such metrics the following research question is formulated,

Performance Assessment Research Question

How can MAV-based pest interception performance be quantitatively assessed?

Firstly, section 4.1 presents the kinematic equations of motion that describe the relative dynamics between a MAV and an agile aerial pest. This section directly addresses the first research question by establishing the mathematical framework needed to model MAV-to-pest engagement dynamics.

Next, section 4.2 surveys three canonical pursuit strategies—classical pursuit, constant bearing pursuit, and constant absolute target direction (CATD)—and introduces quantitative metrics for evaluating how closely a given controller adheres to each. Finally, the chapter concludes with section 4.3 where a set of interception performance metrics that extend the analysis from geometric adherence to practical effectiveness. These metrics allow for a detailed assessment of how reliably and efficiently a controller performs the interception task. Together, the second and third sections directly contribute to answering the second research question.

4.1. Engagement Kinematic Equations of Motion

The kinematic equations of motion for the aerial-to-aerial interception problem are given in this section. These serve as the baseline for the derivation of the respective aerial-to-aerial interception guidance laws. The derivation of the kinematics equations of motion and frames defined here are akin to what is presented in the technical report of Palumbo et al. [5].

The LOS coordinate frame is a reference system defined along the direction of the relative position vector between the pursuer and the evader. In this context, the pursuer refers to the MAV attempting to intercept the target, while the evader represents the agile pest attempting to escape. As explained by Palumbo et al. [5], the LOS frame is particularly useful because it isolates the line-of-sight angle, the angle between the relative position vector and a fixed inertial frame. This angle plays a central role in many guidance laws, as it directly characterizes the direction of interception. In addition to this, the LOS frame simplifies the analysis of relative motion by enabling more direct calculation of relative velocity and acceleration components along and transverse to the LOS, which are critical for formulating effective guidance laws.

While the LOS coordinate frame is particularly well-suited for analyzing interception dynamics, it is not the only option. Alternative frames, such as an inertial frame, express motion in absolute coordinates but obscure critical interception quantities like line-of-sight angle and closing velocity, requiring additional transformations. A body-relative LOS frame, which rotates with the MAV, can simplify onboard control design by aligning with the MAV's sensing and actuation systems, but it complicates interpretation of relative motion from an external perspective. To aid interpretability in this analysis, the LOS frame will be defined relative to a fixed coordinate system. However, it is worth noting that for control implementation, a body-fixed LOS frame offers advantages in terms of integration with onboard systems.

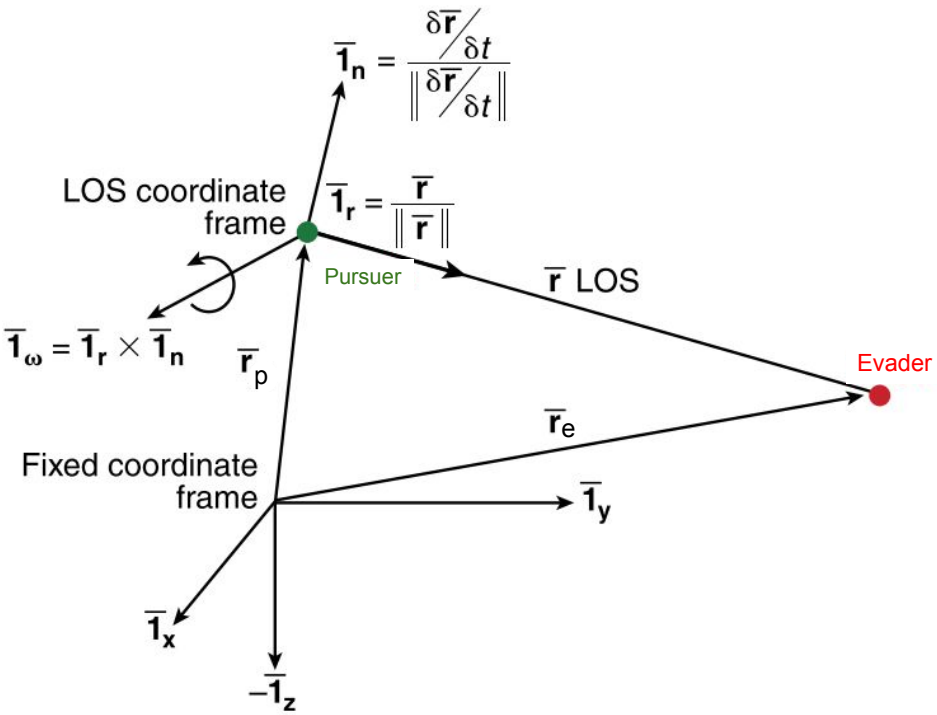


Figure 4.1: LOS coordinate frame for aerial-to-aerial engagement. Image retrieved from [5], with symbolism modified for the convention of this report

Consider the engagement geometry shown in Figure 4.1. The relative position vector $\mathbf{r}$ between the evader and the pursuer is defined as Equation 4.1,

$$\mathbf{r} = \mathbf{r}_e - \mathbf{r}_p \tag{4.1}$$

where $\mathbf{r}_e$ and $\mathbf{r}_p$ are the position vectors of the evader and the pursuer, respectively, both expressed in a fixed inertial coordinate frame. The magnitude of this vector, denoted $\|\mathbf{r}\|$, represents the scalar distance between the agents. Alternatively, the relative position can be expressed using a LOS unit vector $\mathbf{I}_r$, defined as Equation 4.2,

$$\mathbf{r} = \|\mathbf{r}\|\mathbf{I}_r \tag{4.2}$$

where $\mathbf{I}_r$ is a unit vector pointing from the pursuer to the evader. This formulation isolates the direction of interception and provides a foundation for expressing relative velocity and acceleration in LOS-aligned coordinates. Taking the time derivative of Equation 4.2 yields the relative velocity $\dot{\mathbf{r}}$ between the two agents given through Equation 4.3,

$$\dot{\mathbf{r}} = \|\dot{\mathbf{r}}\|\mathbf{I}_r + \|\mathbf{r}\|\frac{d}{dt}\mathbf{I}_r \tag{4.3}$$

where $\|\dot{\mathbf{r}}\|$ is the time derivative of the range, and $\frac{d}{dt}\mathbf{I}_r$ is the time derivative of the LOS direction vector (both as introduced in Equation 4.2). The second term in Equation 4.3, $\frac{d}{dt}\mathbf{I}_r$, can be rewritten using the angular velocity of the LOS, which yields Equation 4.4,

$$\dot{\mathbf{r}} = \|\dot{\mathbf{r}}\|\mathbf{I}_r + \|\mathbf{r}\|\left(\dot{\boldsymbol{\phi}} \times \mathbf{I}_r\right) \tag{4.4}$$

where $\dot{\boldsymbol{\phi}}$ is the angular velocity vector of the LOS with respect to the inertial frame. Taking the second time derivative of $\mathbf{r}$, as defined in Equation 4.1, results in the relative acceleration $\ddot{\mathbf{r}}$ defined in Equation 4.5,

$$\ddot{\mathbf{r}} = \|\ddot{\mathbf{r}}\|\mathbf{I}_r + 2\|\dot{\mathbf{r}}\|(\dot{\boldsymbol{\phi}} \times \mathbf{I}_r) + \|\mathbf{r}\|(\ddot{\boldsymbol{\phi}} \times \mathbf{I}_r) + \|\mathbf{r}\|\left[\dot{\boldsymbol{\phi}} \times (\dot{\boldsymbol{\phi}} \times \mathbf{I}_r)\right] \tag{4.5}$$

here, $\|\ddot{\mathbf{r}}\|$ is the second derivative of the range, $\dot{\boldsymbol{\phi}}$ and $\ddot{\boldsymbol{\phi}}$ are the angular velocity and angular acceleration of the LOS vector as introduced in Equation 4.4. $\mathbf{I}_r$ is as defined in Equation 4.2. This formulation captures radial, tangential, and centripetal components of the relative acceleration and forms the basis for deriving guidance laws in subsequent sections.

4.2. Taxonomy of Pursuit Strategies

In this section, a taxonomy of fundamental pursuit strategies is introduced that serves as both a foundation for analyzing advanced guidance laws and a diagnostic tool for evaluating black-box methods such as learned controllers. By establishing clear metrics to measure adherence to these strategies, one can infer whether a controller approximates or diverges from established pursuit paradigms, even when the internal decision-making is opaque. Since pursuit and interception guidance are closely related—both aiming to reduce the separation between pursuer and target over time—these strategies are directly applicable to the task of MAV-based pest interception. Consequently, this allows for more informed assessments of whether complex or learned guidance control laws approximate or diverge from well-understood pursuit paradigms.

4.2.1. Classical Pursuit and Constant Bearing Pursuit

According to the work of Wei et al. [6], three fundamental pursuit strategies may be identified. The first two, classical pursuit and constant bearing pursuit, are grouped under a shared analytical framework with a common metric.

Firstly, classical pursuit describes a strategy in which the pursuer continuously steers directly toward the target's current position. Secondly, constant bearing pursuit is when the pursuer maintains a constant

relative heading error instead of perfect alignment with the LOS. Instead of constantly adjusting to directly aim toward the target's instantaneous position, the pursuer adjusts its path to follow the target with a fixed angle between its direction of motion and the relative position vector.

To evaluate how well a given motion trajectory adheres to these strategies, Wei et al. [6] introduce a cost function. This cost function is extended to a three dimensional case and given through Equation 4.6,

$$\Lambda(\theta, \phi, \psi) = -\frac{\mathbf{r}}{\|\mathbf{r}\|} R_{rot}(\theta, \phi, \psi) \frac{\dot{\mathbf{r}}_p}{\|\dot{\mathbf{r}}_p\|} \tag{4.6}$$

where $R_{rot}(\theta, \phi, \psi)$ is a rotation matrix representing the desired angular deviation between the pursuer's heading and the LOS, parameterized by yaw ψ, pitch θ, and roll ϕ. The parameters $\mathbf{r}$, and $\mathbf{r_p}$ follow the definitions given in Equation 4.1.

The value of the cost function $\Lambda(\theta, \phi, \psi)$ is bounded in the interval $[-1, 1]$, where a value of -1 indicates perfect adherence to the desired pursuit strategy. For instance, when $(\theta, \phi, \psi) = (0, 0, 0)$, Equation 4.6 reduces to a form that quantifies alignment with classical pursuit. More generally, nonzero values of (θ, ϕ, ψ) define constant deviation angles used in constant bearing pursuit. Intuitively this cost function measures the alignment of the velocity of the pursuer with the LOS to the evader.

4.2.2. Constant Absolute Target Direction

The third strategy, CATD, identified by Wei et al. [6] involves the pursuer moving in such a way that it always appears to the evader as if the pursuer is staying on the same bearing or direction in space. Intuitively, the pursuer does not appear to move sideways relative to the evader. This results in a stable directional vector toward the target. For this strategy Wei et al. [6] also introduced a cost function given through Equation 4.7,

$$\Gamma = -\left(\frac{\mathbf{r}}{|\mathbf{r}|} \cdot \frac{\dot{\mathbf{r}}}{|\dot{\mathbf{r}}|}\right) \tag{4.7}$$

where $\mathbf{r}$ and $\dot{\mathbf{r}}$ follow the definitions given in Equation 4.1 and Equation 4.2, respectively.

This cost function Γ may be understood as measuring the extent to which relative motion occurs along the LOS directional vector. The value of Γ is bounded in the interval $[-1, 1]$, a value of $\Gamma = -1$ indicates that the relative motion occurs entirely along the LOS, meaning the pursuer motion is consistent with CATD behavior.

Both cost metrics, $\Lambda(\theta, \phi, \psi)$ and Γ, are diagnostic tools to assess the extent to which a pursuer follows the intended strategy. While they do not measure interception effectiveness directly, they provide valuable insight into the geometric and behavioral consistency of a complex or learned controller with these fundamental pursuit strategies.

4.3. Metrics for Evaluating Interception Performance

While the previous section establishes metrics for quantifying a controller's adherence to idealized pursuit strategies, this section defines a complementary set of metrics aimed at evaluating the overall performance of an MAV in executing the interception task. These performance metrics provide a quantitative basis for comparing different controllers or guidance strategies in terms of effectiveness, persistence, and spatial behavior throughout the engagement. The metrics are similar in spirit to those used by Pliska et al. [7], and tailored here to suit the specific requirements of MAV-based pest interception.

The *time to first interception* measures the elapsed time between the start of the engagement and the first successful interception event, defined as the moment when the Euclidean distance between the MAV and the target drops below 15 cm. This threshold is specified by PATS as a sufficient condition for a successful interception. This metric captures how quickly the controller is able to initiate a successful interception. The *interception rate* reflects the percentage of trials in which at least one interception occurs, serving as a measure of robustness, especially in more challenging scenarios where interception is not guaranteed.

To assess near-successes, the *near-miss time* records the total duration the MAV remains within a 30 cm radius of the target. This metric provides insight into the MAV's ability to consistently close the distance to the evader, even if an interception does not occur. The *mean distance* gives an overall measure of the spatial relationship between the pursuer and evader by averaging their separation across the full engagement. Lower values reflect stronger proximity maintenance and tracking capability. Finally, the *distinct interceptions* metric counts the number of separate interception events. Given trials do not terminate after the first interception, this metric evaluates the controller's ability to repeatedly re-engage and successfully intercept the target multiple times.

These metrics offer a multi-dimensional view of interception performance. They capture not only whether interception occurs, but how efficiently, reliably, and persistently it is achieved. When used in combination with the pursuit adherence metrics from the previous section, they enable a comprehensive evaluation of both the guidance strategy and its execution in real-time MAV-based pest interception tasks.

4.4. Conclusion

This chapter established the foundational framework necessary for analyzing and evaluating MAV-based pest interception. First, the kinematic equations of motion were derived in a line-of-sight coordinate frame, capturing the relative dynamics between the MAV and the evading pest. This formulation directly addressed the first research question by providing a basis for describing the interception geometry.

Building on this foundation, a taxonomy of classical pursuit strategies was presented, along with diagnostic cost functions for quantifying a controller's adherence to each strategy. These tools enable the evaluation of both interpretable and black-box guidance systems based on their geometric behavior.

Finally, a suite of interception performance metrics was introduced to assess the practical effectiveness of a controller, including its ability to initiate, maintain, and repeat successful interceptions. These metrics respond to the second research question by offering a quantitative framework for evaluating not just whether interception occurs, but how well it is performed. Together, the formulation, strategies, and metrics developed in this chapter lay the groundwork for the comparative evaluation of advanced and learned interception controllers explored in the chapters that follow.

5

Proportional Navigation

This chapter explores the application of PN for guiding MAVs toward agile aerial targets. As indicated in the preliminary system architecture in Figure 3.1, guidance logic forms a crucial intermediate layer, processing the estimated state of the target and issuing commands to the MAV's control systems. In this context, effective guidance strategies must translate uncertain, rapidly evolving target trajectories into interception commands, even under the constraints of onboard processing and flight dynamics.

Among the strategies available, PN stands out due to its long-standing success in missile interception guidance systems. The fundamental idea is to steer the interceptor so that the LOS angle to the target remains constant. By keeping the LOS angle rate constant, the relative geometry between the interceptor and the target does not change, effectively forcing them onto a collision course.

Palumbo et al. [5] highlight several reasons for the widespread use of PN in interception guidance. Firstly, PN requires only minimal target information, primarily the LOS rate, thereby reducing the complexity of onboard sensors required. Secondly, PN guidance laws are computationally light allowing them to be easily integrated into onboard computing hardware. Finally, under certain simplifying assumptions about target and interceptor dynamics, PN laws are optimal with respect to the terminal interception miss distance, an especially valuable property in MAV pest interception, where the small physical size of the targets demands extremely precise guidance.

In recent years, the rapid growth of drone technology has motivated a re-examination of PN within the context of drone-to-drone interception. Drones operate under markedly different conditions compared to missiles; they fly at lower speeds, exhibit greater maneuverability, and encounter more complex environmental dynamics. These differences necessitate a re-evaluation of classical PN guidance laws to ensure their suitability and effectiveness in aerial platforms where flight dynamics and operational constraints diverge significantly from traditional missile systems. To this end the following research question is posed,

PN Research Question

Which of the available PN guidance laws are most effective for aerial-to-aerial interception of highly maneuverable targets, given the unique dynamics and operational constraints of MAVs?

To address this question, the chapter is organized as follows. In section 5.1 the fundamentals of PN are introduced, including the mathematical formulation and essential conditions for interception. Here different adaptations of the PN are presented within the framework. Next, section 5.2 evaluates these guidance laws through the means of a preliminary simulation study conducted. Finally, section 5.3 summarizes the findings, suggesting which PN guidance law is most suitable for use in the system architecture and highlighting future areas for research.

5.1. Fundamentals of Proportional Navigation

The core working principle of PN is to minimize miss distance by ensuring that the LOS to the target remains constant. This fundamental idea, grounded in the kinematic relationships presented in Equation 4.5, forms the basis for a range of PN guidance laws. Palumbo et al. [5] identify a pair of conditions that, if satisfied, guarantee interception. First, the rate of change of the LOS angle must be zero ($\dot{\phi} = 0$), ensuring that the pursuer remains on a collision course with the target. Second, the pursuer must be able to reduce the range along the LOS. This requires that its acceleration projected onto the LOS eventually exceeds that of the evader, particularly when the relative velocity is not initially closing.

These conditions, if met, ensure that an interception will occur based solely on the geometry and dynamics of the system. Therefore, PN guidance laws can be designed to generate acceleration commands that actively drive the system toward satisfying these conditions. In the works of Yang and Yang [8] they suggest a unified framework that encapsulates all the basic methods of PN. They highlight that the commanded acceleration of the pursuer can be expressed by the scheme shown in Equation 5.1,

$$\mathbf{a_p} = \lambda \mathbf{L} \times \dot{\phi} \tag{5.1}$$

where λ is a navigational constant, and $\mathbf{L}$ is the normal direction of the command acceleration. The definition of $\dot{\phi}$ is as stipulated in Equation 4.4. Different PN guidance laws employ different forms of $\mathbf{L}$, there are two main variations. The first, true PN (TPN) has LOS referenced laws [9]. In this case the normal direction of the acceleration takes the form $\mathbf{L} = \mathbf{I_r}$. The latter, is pure PN (PPN) in which control laws are referenced relative to the pursuer velocity vector [10], therefore yielding $\mathbf{L} = \mathbf{v_p}$, where $\mathbf{v_p}$ is the velocity of the pursuer in an inertial frame.

As highlighted by Shukla et al [10] each formulation offers distinct trade-offs. TPN has been shown to be optimal in minimizing the terminal miss distance, defined as the closest point of approach between the pursuer and the evader [5]. However, TPN requires accurate estimation of the LOS direction $\mathbf{I_r}$, which may not always be readily measurable. In contrast, PPN is more practical for onboard implementation, as the pursuer's own velocity $\mathbf{v_p}$ is directly available from inertial sensors, making it simpler to implement. As such, while TPN offers theoretical performance benefits, PPN may be preferred in hardware-constrained MAV applications where sensor limitations are a factor. All subsequent variations of classical PN stem from one or a combination of these fundamental methods [8, 11, 12]; for their guidance laws and working principles, refer to Table 5.2.

5.1.1. Augmented Proportional Navigation

In classical PN, the acceleration command relies purely on feedback from the LOS rate $\dot{\phi}$. However, in practical scenarios where the target maneuvers aggressively, a feedforward control term, $\mathbf{a_{ff}}$ can be introduced to compensate for target acceleration and improve interception performance [12]. This term depends on higher-order derivatives of target motion, such as acceleration. Thus, modifying Equation 5.1 to include feedforward control. The inception of this control scheme is called augmented PN (APN), and the general form of the commanded acceleration due to PN guidance is shown in Equation 5.2,

$$\mathbf{a_p} = \lambda \mathbf{L} \times \dot{\phi} + N\mathbf{a_{ff}} \tag{5.2}$$

where N is the gain applied to the feedforward control term $\mathbf{a_{ff}}$ and λ, $\mathbf{L}$, $\dot{\phi}$ are defined per Equation 5.1. Note, APN relies on higher-order kinematic information and in many practical scenarios, such information may not be directly measurable and must instead be estimated. This process is highly susceptible to noise amplification, potentially degrading the quality of the estimated acceleration and, in turn, the effectiveness of the APN guidance law. As a result, while APN can offer improved performance in theory, its practical implementation is often constrained by the quality of measurements avaliable.

5.1.2. Global Proportional Navigation

PN methods typically assume that the pursuer is initially closing in on the evader and that the relative speed remains negative throughout the engagement ($\|\dot{\mathbf{r}}\| < 0$). To add, classical PN formulas rely on

$\dot{\phi} \neq 0$ to produce meaningful acceleration commands. However, if these conditions are not satisfied, classical PN yields negligible control effort.

Zhu et al. [13] tackle this limitation by introducing Global Integrated PN (GIPN). Rather than relying solely on feedback from the rate of LOS angle ($\dot{\phi}$), GIPN adds a velocity-convergence term that enforces the pursuer's relative speed with respect to the evader $\|\dot{\mathbf{r}}\|$ to be a predetermined negative value. This additional term gives the pursuer another mean to re-accelerate toward the evader, independent of the LOS angle rate feedback.

Although Zhu et al. [13] specifically demonstrated their velocity-convergence term for GIPN, the same conceptual modification may be similarly applied to other PN variants. This highlights the broader applicability of the global correction approach to a wide range of PN laws. In general a global modification of the PN law may be formulated per Equation 5.3,

$$\mathbf{a_p} = \lambda \mathbf{L} \times \dot{\phi} + N\mathbf{a_{ff}} + \left[k_2(|\dot{\mathbf{r}}| - v_r) + \dot{\mathbf{r}} \cdot \dot{\boldsymbol{\phi}}\right] \mathbf{I}_r \tag{5.3}$$

where v_r is the desired relative closing speed, and λ, $\mathbf{L}$, and $\dot{\phi}$ are defined as in Equation 5.1, while the definitions of $\dot{\mathbf{r}}$, $\mathbf{I}_r$, and $\dot{\boldsymbol{\phi}}$ are as previously provided in Equation 4.4. As explained in the work of Zhu et al. [13], it is important to note that the two components of this equation act in orthogonal directions. The classical PN term, $\lambda \mathbf{L} \times \dot{\phi}$, generates lateral acceleration perpendicular (or approximately perpendicular depending on $\mathbf{L}$) to the LOS vector, guiding the pursuer in angular space to ensure minimal miss distance. In contrast, the added velocity-convergence term acts along the LOS vector, adjusting the rate at which the pursuer closes in on the evader. Because these control efforts are decoupled, they do not interfere with one another. As such, the global correction term does not compromise the core convergence guarantees of the classical PN law.

While enforcing a predefined closing velocity helps prevent stalling, it introduces notable trade-offs. If the pursuer is already closing in faster than the desired speed, the guidance law may issue decelerative commands to match v_r, despite the fact that this faster convergence may be advantageous. In addition, this correction term applies force along the LOS, which can lead to control saturation: the system may prioritize matching the closing speed at the expense of lateral acceleration needed to adjust heading. As a result, lateral corrections driven by angular guidance may be diminished or delayed.

5.1.3. Modern Guidance Laws

Interception strategies can be developed using optimal control theory. In particular, linear quadratic optimal control provides a framework in which guidance laws can be derived analytically to optimize a specific performance metric. Using this methodology, Palumbo et al. [5] derive a family of optimal guidance laws of varying complexity. The complexity of each law corresponds to the amount of target state information assumed to be available and whether actuator dynamics are modeled. Notably, many of these optimal laws can be viewed as generalizations or supersets of classical PN formulations.

One such law, referred to as Linearized PN (LPN), is evaluated by Pliska et al. [7] in the context of drone-based aerial target interception. The LPN guidance law is expressed in Equation 5.4,

$$\mathbf{a_p} = \lambda \left(\frac{\mathbf{r} + \dot{\mathbf{r}} \cdot t_{\text{go}}}{t_{\text{go}}^2} \right) \tag{5.4}$$

where t_{go} refers to the estimated time till interception, defined as $t_{\text{go}} = \frac{||\mathbf{r}||}{||\dot{\mathbf{r}}||}$. The definitions of $\mathbf{r}$ and $\dot{\mathbf{r}}$ are as defined previously in Equation 4.4. Note, that the metric optimized to derive LPN is the terminal interception distance. Palumbo et al. [5] show that the LPN control law is a linearized and cartesian version of the classic TPN control law. A key advantage of the LPN control law lies in its formulation. It operates directly in terms of relative position and velocity, avoiding the need to transform into the LOS coordinate frame. This leads to a simpler and more intuitive expression for the commanded acceleration.

However, the derivation of LPN relies on several simplifying assumptions: constant speeds for both target and pursuer, negligible LOS angles, a non-maneuvering target, ideal actuation with no delay,

and full observability of all relevant states [5]. Only under these assumptions is the optimality of LPN guaranteed. These assumptions are unlikely to hold in practical MAV interception scenarios, where high maneuverability, partial observability, and noisy sensing are the norm.

To improve the initial responsiveness of the guidance system, Pliska et al. [7] propose a modified version of LPN called Fast Response PN (FRPN). This hybrid method combines the precision of LPN with the rapid engagement characteristics of classical pursuit guidance by introducing a weighted feedback term. The FRPN law is defined in Equation 5.5,

$$\mathbf{a_p} = \lambda \left((1 - W) \frac{\mathbf{r} + \dot{\mathbf{r}} \cdot t_{\text{go}}}{t_{\text{go}}^2} + W \cdot \mathbf{r} \right) \tag{5.5}$$

where $W \in [0, 1]$ is a tuning parameter that controls the influence of the classic pursuit component relative to the LPN term. By blending these two strategies, FRPN enables faster engagement initiation while maintaining precise terminal guidance.

5.2. Evaluation of Proportional Navigation Guidance Laws

While classical PN laws have traditionally been evaluated in the context of missile guidance [8, 11, 12, 5], the significantly different dynamic and operational characteristics of MAVs necessitate a re-evaluation of these laws for aerial-to-aerial interception. In order to assess the viability of different PN laws in this context, a comparative analysis was conducted.

A preliminary simulation environment is set up where the control laws may be tested across scenarios, an overview is presented in subsection 5.2.1. The evaluation framework builds directly on the interception performance metrics introduced in section 4.3. Subsequent sections delve into these specific metrics, and a summary of the performance with respect to these metrics is presented in Table 5.3. To this end, subsection 5.2.2 investigates the time to first interception of the pursuer to the evader, subsection 5.2.3 evaluates the ability to re-attempt interception, subsection 5.2.4 looks into the dependency on initial conditions, with results summarized in Table 5.2, and finally subsection 5.2.5 explores the robustness to system dynamics and noise.

5.2.1. Simulation Set-up

The simulation models a simplified engagement scenario between a pursuer and an evader in a three-dimensional space, free from gravity and aerodynamic forces. The evader follows predefined trajectories: straight-line, circular, and figure-eight paths, chosen to represent different levels of maneuverability. The velocity and acceleration of the evader are computed via numerical differentiation of its position data. Interception is classified as occurring when the pursuer comes within 15 cm of the evader. This extremely small interception radius reflects the minute physical size of flying pests and the high spatial precision required in such applications and follows from the criteria stipulated by PATS.

The pursuer responds to acceleration commands generated by the PN law under evaluation. These commands are filtered through a first-order transfer function to simulate actuator response delay as well as command communication latency. The resulting acceleration is clipped to a maximum magnitude of 10 m/s², serving as a performance cap to reflect conservative physical limits on the MAV's maneuvering capabilities. Additionally, Gaussian noise can be introduced to the evader's position to emulate sensor noise in the perception system.

Several limitations apply to this setup. No state estimation techniques are employed, meaning that any noise added to the position data propagates unfiltered into velocity and acceleration estimates. The pursuer model is also highly abstracted; while it includes a first-order response delay, it does not consider actuator saturation, control authority, or flight stability constraints. Moreover, the fixed acceleration cap of 10 m/s², while reasonable, may underestimate the capabilities of some high-performance MAV platforms. Finally, coarse gain tuning was performed for each PN law based on time to first interception in a nominal straight-line scenario with favorable geometry. As this tuning was not exhaustive and optimized for only a single case, some PN laws may operate with suboptimal gains. Despite these limitation, the simulation provides a controlled environment to isolate and compare the important aspects of the different PN guidance strategies.

5.2.2. Time to First Interception

All PN methods under consideration are capable of achieving interception when the evader follows a straight-line path, a circular trajectory, or figure-eight motion. This highlights the general applicability of PN-based guidance schemes to a variety of motion profiles, including cases where the evader demonstrates continuous acceleration or aggressive maneuvering.

Among these methods, the globally modified guidance laws exhibit significantly faster time to first interception. This performance can be attributed to the relative velocity-convergence term, which explicitly encourages a consistently negative closing velocity between the pursuer and evader. As a result, the pursuer accelerates not just laterally but also longitudinally along the LOS, reducing the time required to intercept.

FRPN also exhibits outstanding performance in this metric. In addition to inheriting the properties of linearized PN, it includes a weighted proportional-position feedback term. This term directs the pursuer's acceleration more aggressively toward the evader, further reducing the time required to close the gap.

By contrast, classical PN laws such as TPN, PPN, and their augmented variants primarily rely on lateral acceleration components that are orthogonal, or approximately orthogonal, to the LOS vector. This design stems from the original intent of these laws, which focused on minimizing miss distance rather than optimizing time to intercept. In addition, this reliance on lateral acceleration in TPN and PPN stems from their origin in missile guidance, where longitudinal acceleration is typically assumed to be saturated or not independently controllable due to propulsion constraints. As a consequence, they lack the capability to directly propel the pursuer along the LOS toward the evader, which slows down convergence and leads to higher initial interception times.

5.2.3. Reattempting Interception

In drone interception scenarios, a single miss does not render the MAV expendable, which distinguishes it from traditionally studied missile engagements. Consequently, the MAV's guidance must enable re-engagement after an initial failure. Most PN-based research, however, focuses on a single intercept event [8, 11, 12, 14, 9], where the LOS rate $\dot{\phi} \approx 0$ and the closing velocity $\dot{R} < 0$ persist until impact. After passing the target, the LOS rate remains approximately zero, $\dot{\phi} \approx 0$. As a result, classical PN laws which are all dependent on the feedback of $\dot{\phi}$, yield negligible commanded acceleration, leaving the drone on its pre-interception trajectory. In this case, the control input is minute, and the pursuer diverges from the evader.

Two modern PN frameworks circumvent this limitation. Firstly, global PN laws introduce a velocity convergence term that ensures a minimum negative closing velocity at all times, thereby letting the pursuer correct its heading even with $\dot{\phi} \approx 0$ after a missed interception. Secondly, linearized PN laws formulate the guidance in Cartesian coordinates and thus are not explicitly dependent on $\dot{\phi}$. By avoiding dependence on the LOS angle, linearized methods enable reattempts at interception.

To quantify each law's capacity for multiple interceptions, the following metrics is used. The fraction of total time the pursuer remains within a 30 cm radius from the evader, classified as being within a near-miss range. A higher value here implies stronger post-miss recovery and consistent proximity to the evader. The second metric is the number of distinct interceptions observed, a higher count indicates the controller successfully drives the pursuer back into contact at multiple distinct points.

As summarized in Table 5.3, FRPN and LPN exhibit the best performance with respect to re-engagement metrics. Among these, the linearized guidance laws demonstrate a distinct advantage in re-interception scenarios due to their ability to produce more refined and stable control near the target. In contrast, globally modified PN laws, while effective at re-initiating pursuit, enforce a persistent non-zero closing velocity objective, which tends to reduce control precision as the pursuer nears the evader. This often leads to overshoot after an interception attempt, highlighting a key trade-off between re-engagement capability and fine-grained terminal accuracy for these methods. The re-interception trajectories for some PN laws are given in Figure 5.1 to visualize their performance.

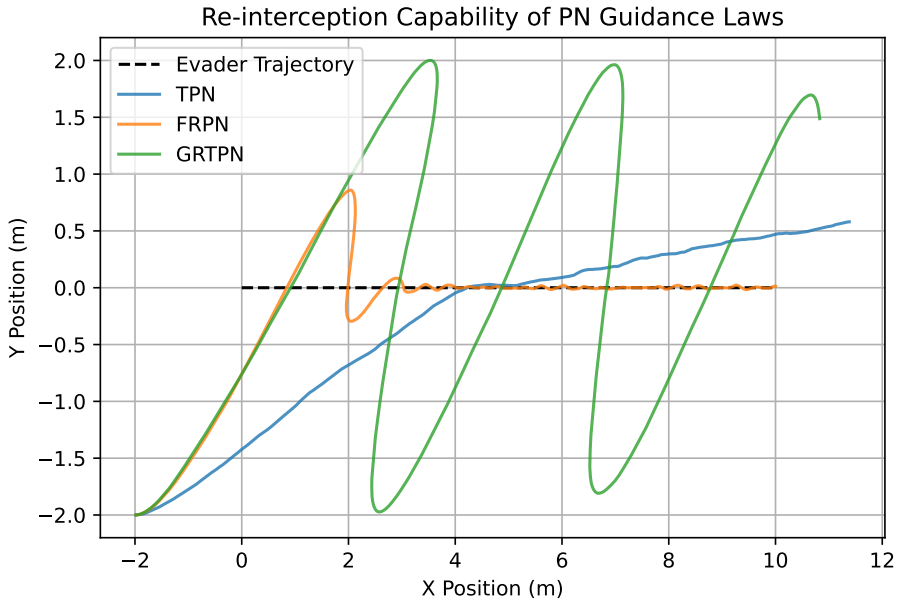


Figure 5.1: Trajectories of the FRPN, TPN, and GRTPN guidance laws and a straight line evader, highlighting re-interception performance

5.2.4. Dependency on Initial Conditions

To assess how each PN variant copes with diverse engagement geometries, multiple initial conditions were explored. Including scenarios where the evader moves twice as fast as the pursuer, the pursuer's initial velocity is oriented away from the evader, the pursuer attempts a tail chase with a slower initial velocity than the evader, or the pursuer starts from rest. Table 5.2 summarizes the performance of each guidance law under these distinct setups.

A key insight is that PN laws, which rely solely on the LOS angular rate ($\dot{\phi}$), are the most dependent on the initial conditions. The reason is that, the pursuer under guidance of these laws may converge to a path that yields a constant LOS angular rate ($\dot{\phi} \approx 0$), however this path will not lead to an interception. This divergent path is aliased with the path to interception from the perspective of the control law as both achieve a $\dot{\phi} \approx 0$. An example of such a path is shown in Figure 5.2. In particular these aliased paths occur when the initial closing velocity is negative ($V_c < 0$). When on such a path the zero angular rate induces no acceleration toward the evader, leaving the pursuer effectively stalled on a diverging trajectory.

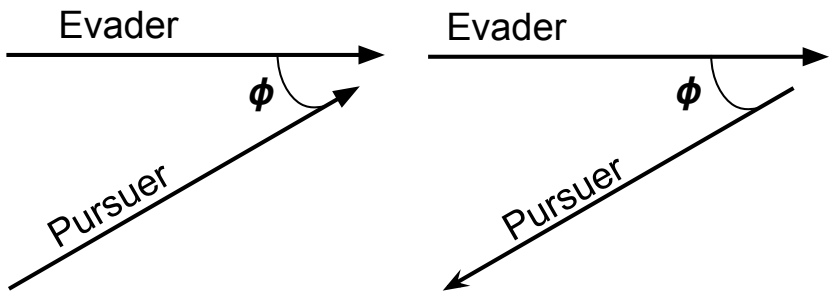


Figure 5.2: Aliased trajectories for control laws that rely on $\dot{\phi}$ feedback only.

Similarly, if the evader travels significantly faster than the pursuer, a law that provides little or no longitudinal acceleration cannot force the pursuer to catch up. This shortcoming is especially evident when the pursuer begins with suboptimal initial velocity. Lateral guidance alone is insufficient in those cases, since the pursuer needs a mechanism to increase its closing speed rather than just maneuver around

the evader's trajectory. This is the reason why the TPN, RTPN, and PPN methods all fail to intercept if the pursuers starts from an initial rest position.

Among these different tests, the tail chase of a non-maneuvering target emerged as particularly demanding for traditional PN designs. If the pursuer is already behind the target but starts with $\dot{\phi} = 0$, an angular-rate-based controller will register no guidance cue, thus failing to accelerate even if the guidance law operates longitudinally. More advanced methods, whether through global modifications or linearized frameworks, implicitly address the issue discussed above and are as a result robust to this initial condition.

5.2.5. Robustness to Actuator Delay and Noise

In practical applications, both actuator delay and sensor noise can significantly degrade the performance of guidance laws. In our simulations, actuator delay (τ) is modeled as a first-order lag representing the real system's response delay to commanded accelerations, while noise (σ) is introduced on the evader's observed position. As shown in Table 5.1, interception rate of all guidance laws worsen as noise and delay increase. Notably, it was observed noise is particularly detrimental, likely because the velocity and acceleration estimates are obtained through differentiation schemes, which inherently amplifies noisy signals.

Table 5.1: Interception Rate (%) for Different Guidance Laws at Varying Measurement Noise and Delay Levels. Each scenario is evaluated over 100 trials. The evader moves at a constant velocity of 1 m/s in a straight line, while the pursuer moves at 1.25 m/s, with randomized initial positions on a sphere of radius 2 m centered at the evader's initial position, and initial velocity directed toward the evader.

Guidance Law	$\sigma = 0.010\,\text{m}$ $\tau = 0.01\,\text{s}$	$\sigma = 0.025\,\text{m}$ $\tau = 0.01\,\text{s}$	$\sigma = 0.050\,\text{m}$ $\tau = 0.01\,\text{s}$	$\sigma = 0.010\,\text{m}$ $\tau = 0.5\,\text{s}$	$\sigma = 0.010\,\text{m}$ $\tau = 1.0\,\text{s}$
TPN	**100**	**60**	**14**	**53**	**29**
RTPN	**100**	**60**	**15**	**60**	**25**
PPN	**100**	**86**	**16**	**47**	**17**
IPN	**100**	**33**	**19**	**54**	**17**
LPN	**100**	**100**	**100**	**91**	**34**
FRPN	**100**	**100**	**100**	**63**	**31**
AIPN	**100**	**36**	**12**	**25**	**11**
GTPN	**100**	**98**	**31**	**61**	**28**
GRTPN	**100**	**99**	**43**	**63**	**29**
GPPN	**100**	**73**	**17**	**43**	**35**
GIPN	**100**	**45**	**15**	**52**	**18**
GAIPN	**100**	**32**	**12**	**26**	**11**

As actuator delay increases, the response of the pursuer progressively lags behind the commands generated by the guidance law. This lag reduces the pursuer's ability to promptly adjust its trajectory toward the evader, which in turn degrades interception performance. Initially, moderate levels of delay can be tolerated, as the pursuer is still able to correct its motion sufficiently to achieve interception. However, beyond a certain threshold, the accumulated lag prevents effective realignment, causing the pursuer to miss the target. This highlights the importance of fast system response for interception tasks, as greater delays diminish the controller's ability to track the intended guidance path accurately and maintain a successful pursuit.

The impact of actuator delay is amplified when intercepting maneuvering targets. Palumbo et al. [5] quantitatively investigate this by evaluating the final miss distance of LPN-guided missiles under varying levels of target acceleration and interceptor delay. As shown in Figure 5.3, increased delay leads to significantly larger miss distances, especially against highly agile targets. While ideal interceptors with negligible delay maintain consistent accuracy.

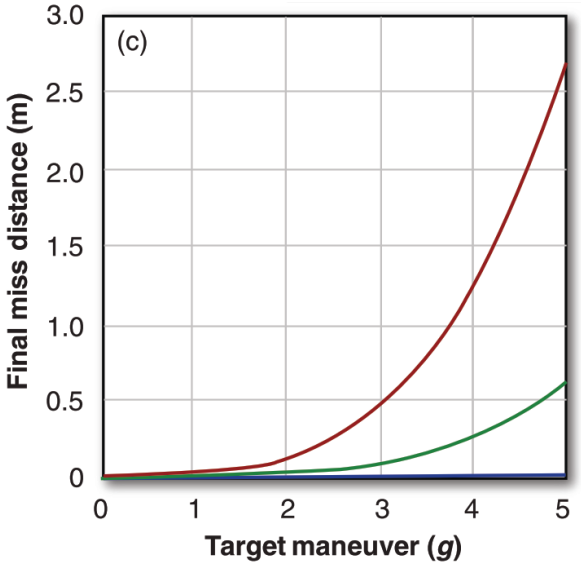


Figure 5.3: Final miss distance of an LPN-guided interceptor as a function of target maneuver acceleration, expressed in multiples of gravitational acceleration. The red, green, and blue curves correspond to interceptor actuator delay time constants of $\tau = 0.5\,\text{s}$, $\tau = 0.3\,\text{s}$, and $\tau = 0.1\,\text{s}$, respectively. The blue curve lies close to the x-axis, indicating minimal miss distance for near-ideal actuator response. Image retrieved from [5].

A more equitable assessment of noise would involve applying state estimation techniques to filter higher-order kinematic signals and thereby mitigate some of the effects of measurement noise. Nonetheless, our preliminary analysis highlights important trends in guidance law design under noisy conditions. In particular, linearized control laws exhibit greater robustness because they operate independently of noisy angular rate measurements. Unlike classical PN methods that rely on line-of-sight rate, linearized laws determine control actions directly from relative position and velocity vectors without requiring explicit angular information. This decoupling from directional noise explains their superior performance under high-noise scenarios. Furthermore, our results suggest that guidance strategies which primarily depend on the magnitude of the relative velocity, rather than its direction, tend to degrade less in the presence of noise. Thus, minimizing reliance on noisy directional measurements appears critical for designing robust interception algorithms.

5.3. Conclusion

This chapter set out to determine which PN guidance law is most effective for aerial-to-aerial interception of highly maneuverable targets, particularly under the unique dynamics and operational constraints faced by MAVs. Based on the preliminary simulation results, FRPN demonstrates the best overall performance across all evaluation metrics. It consistently achieves successful interception even under adverse initial conditions, and it shows strong robustness to both actuator delay and sensor noise. Unlike classical PN variants, FRPN does not rely explicitly on noisy angular measurements and benefits from a linearized formulation. As such, FRPN emerges as the most promising candidate for implementation within the MAV-pest interception system architecture.

Among the globally modified guidance laws, GRTPN stands out as the best performer. GRTPN maintains high interception rates across different noise levels and initial conditions, leveraging a velocity-convergence mechanism that enables consistent re-engagement after missed intercepts. While its operational behavior differs from FRPN, particularly by imposing a persistent closing velocity constraint, it presents a complementary and valuable approach for robust interception guidance. In summary, FRPN is recommended as the primary guidance strategy for MAV aerial interception tasks, with GRTPN identified as an interesting secondary candidate due to its inherently different control behavior. Future work will focus on the practical implementation of these methods and their validation in real-world MAV interception experiments.

Table 5.2: Comparison of Guidance Laws Across Interception Scenarios. Each method is evaluated based on its acceleration formulation and response to critical engagement conditions.

Method	**Working Principle**	**Acceleration Law** $\mathbf{a}_c$	**Re-attempt Intercept**	$V_c(0) < 0$	$v_e = 2v_{p0}$	**Tail chase &** $V_c(0) < 0$	**Pursuer Stationary**
TPN [9]	Acceleration command generated perpendicular to the LOS, scaled by the initial closing velocity and the rate of change of the LOS angle	$\lambda V_c \cdot (\dot{\boldsymbol{\phi}} \times \mathbf{I}_r)$	✗	✗	✗	✗	✗
RTPN [8]	Variation of TPN using the current closing velocity for better responsiveness	$\lambda V_c(t) \cdot (\dot{\boldsymbol{\phi}} \times \mathbf{I}_r)$	✗	✗	✗	✗	✗
PPN [14]	Acceleration commanded normal to pursuer velocity, scaled by magnitude of velocity	$\lambda(\dot{\boldsymbol{\phi}} \times \mathbf{v}_p)$	✗	✓	✗	✗	✗
IPN [11]	Acceleration command normal to the closing velocity, proportional to LOS rate	$\lambda(\dot{\mathbf{r}} \times \dot{\boldsymbol{\phi}})$	✗	✓	✓	✗	✓
AIPN [12]	Same as IPN but with added feedforward evader acceleration	$\lambda(\dot{\mathbf{r}} \times \dot{\boldsymbol{\phi}}) + \mathbf{a}_e$	✗	✓	✓	✗	✓
LPN [5]	Linearized formulation of TPN via optimal control, minimizing miss and effort	$\lambda\left(\frac{\Delta\mathbf{p}+\Delta\mathbf{v}\cdot t_{\text{go}}}{t_{\text{go}}^2}\right)$	✓	✓	✓	✓	✓
FRPN [7]	Hybrid guidance with weighted position term to increase responsiveness	$\lambda\left((1-W)\cdot\frac{\Delta\mathbf{p}+\Delta\mathbf{v}\cdot t_{\text{go}}}{t_{\text{go}}^2} + W\cdot\Delta\mathbf{p}\right)$	✓	✓	✓	✓	✓
GTPN	Classical PN laws enhanced with a global convergence term to guarantee non-zero closing velocity and improve robustness for re-attempts at interception [13]	PN-law $+\left[k_2(\lVert\dot{\mathbf{r}}\rVert - v_r) + \dot{\mathbf{r}}\cdot\dot{\boldsymbol{\phi}}\right]\mathbf{I}_r$	✓	✓	✓	✓	✓
GRTPN			✓	✓	✓	✓	✓
GPPN			✓	✓	✓	✓	✓
GIPN [13]			✓	✓	✓	✓	✓
GAIPN			✓	✓	✓	✓	✓

Table 5.3: Interception and proximity metrics across all control laws for both the Straight-Line and Figure-8 evader paths. Each scenario is evaluated over 1000 trials. The evader's velocity is constant at 1 m/s, and the pursuer's at 1.25 m/s. The pursuer's initial position is randomized on a sphere of radius 2 m centered on the evader's initial position, with initial velocity directed toward the evader. If data is skewed, it is reported as mean | Q1 | Q3 instead of mean ± standard deviation.

Control Law	1st Intercept Time (s)	Intercept Rate (%)	Near-Miss Time (%)	Mean Distance (m)	Distinct Interceptions
Straight-Line Scenario: ($\sigma = 0.01$ m, $\tau = 0.01$ s)					
FRPN	0.66 ± 0.07	100.0	81.72 ± 0.38	0.15 ± 0.01	**
LPN	0.85 ± 0.12	100.0	83.90 ± 0.74	0.14 ± 0.01	**
GTPN	0.65 ± 0.10	100.0	5.35 ± 0.67	1.17 ± 0.11	5.98 ± 0.60
GPPN	0.73 ± 0.13	100.0	2.95 ± 0.66	1.62 ± 0.18	3.01 ± 0.81
GIPN	0.67 ± 0.07	100.0	2.89 ± 0.87	1.64 ± 0.15	3.21 ± 0.75
IPN	0.71 ± 0.08	100.0	–	–	–
GAIPN	0.82 ± 0.16	99.9	2.97 ± 0.88	1.64 ± 0.30	2.92 ± 1.22
AIPN	0.85 ± 0.18	100.0	–	–	–
GRTPN	0.74 ± 0.11	100.0	5.89 ± 0.90	1.07 ± 0.13	5.12 ± 0.32
PPN	1.74 | 0.97 | 2.50	100.0	–	–	
RTPN	2.11 | 0.97 | 2.48	89.9	–	–	
TPN	2.15 | 0.97 | 2.62	86.1	–	–	
Figure-8 Scenario: ($\sigma = 0.025$ m, $\tau = 0.1$ s)					
FRPN	3.51 | 2.32 | 4.29	97.2	0.61 ± 0.05	0.61 ± 0.05	3.75 | 2.0 | 5.0
LPN	3.28 | 2.23 | 4.21	95.2	0.57 ± 0.05	0.57 ± 0.05	3.59 | 2.0 | 5.0
GTPN	3.62 | 1.36 | 5.58	77.4	1.36 ± 0.48	1.37 ± 0.48	1.49 | 1.0 | 2.0
GPPN	1.37 | 1.20 | 1.49	30.5	4.17 ± 0.59	4.17 ± 0.59	0.33 | 0.0 | 1.0
GIPN	0.76 | 0.45 | 0.92	4.2	–	–	
IPN	0.74 | 0.45 | 0.96	4.6	–	–	
GAIPN	0.64 | 0.47 | 0.49	3.6	–	–	
AIPN	0.65 | 0.47 | 0.49	4.3	–	–	
GRTPN	3.78 | 1.41 | 5.95	80.6	1.26 ± 0.49	1.26 ± 0.49	1.51 | 1.0 | 2.0
PPN	4.53 | 2.03 | 5.57	27.0	–	–	
RTPN	3.72 | 1.09 | 5.97	34.5	–	–	
TPN	3.78 | 1.13 | 5.93	32.0	–	–	

[a] ** The FRPN and LPN laws perfectly track the evader after some time leading to infinite interceptions

6

Optimization

Following the discussion of PN guidance in chapter 5, this chapter explores RL-based approaches for MAV interception of agile aerial pests. As outlined in the preliminary system architecture in Figure 3.1, the RL guidance and control layer is responsible for converting observations, whether raw sensor data or processed estimates, into control commands that may operate at different levels of abstraction. Several characteristics of agile pest interception challenge PN's effectiveness and motivate the exploration of more adaptive, learned strategies.

Firstly, PN applies the same control logic uniformly throughout the entire engagement. However, different phases of interception may demand different objectives. For example, during the early stage of pursuit, when the MAV is still far from the target, nuanced lateral adjustments may not be optimal. PN laws primarily induce acceleration commands perpendicular to the LOS, when the MAV is distant this is not the most effective strategy.

Secondly, PN guidance assumes that the pursuer has equal or greater acceleration capability along the LOS compared to the evader. This assumption breaks down in the context of pest interception. Insects can exhibit extremely high agility, exceeding the MAV's acceleration limits. However, pests typically have lower top speeds than MAVs. This creates an asymmetry in capabilities, the MAV can catch up in the long run but struggles to react as quickly in the short term. PN does not account for this asymmetry and thus fails to exploit the MAV's advantage in maximum speed or the pest's limitations in sustained evasion.

Moreover, PN offers no mechanism for exploiting known patterns in adversary behavior. Learned approaches, by contrast, allows agents to implicitly learn observed pest dynamics. For instance, Vlahov et al. [15] showed that an evader trained against a predictable pursuer developed emergent behaviors specifically tuned to exploit the pursuer's weaknesses. Learned controllers opens up the possibility of training MAVs to recognize and counteract such evasive maneuvers. An additional limitation of methods like PN is the implicit assumption that the target is static or non-reactive. Their is no mechanism that may account for the pests often responding actively to the presence of a pursuing MAV.

Finally, while many PN laws are optimal with resect to minimal terminal miss distance, a valid objective given the small size of pests, other terminal conditions may also or be more relevant. For example, the suction effect generated by the MAV's propellers could be exploited to pull the pest into the blades, effectively increasing the interception zone. Alternatively, high-velocity impacts may be desirable to minimize the pest's chance of last-moment evasion. Such context-specific objectives are not easily optimized using fixed-form control laws like PN.

In light of these limitations, the merit of using a learned approach is clear. For the purpose of aerial-to-aerial interception, similar to the PN laws, most research effort of learned controllers has been dedicated to missile guidance. Nevertheless, application of learned controllers to the interception problem has shown to outperform classical PN laws such as TPN [16] [17], PPN [18] and APN [19]. Despite this promise, the literature offers a spectrum of candidate algorithms with no consensus on which ones

deliver the best interception accuracy and training stability for a single MAV or for an adaptive pursuer–evader pair. Clarifying that gap motivates the first research question,

Algorithm Effectiveness Research Question

Which reinforcement learning algorithms are effective for enabling MAV-based aerial-to-aerial interception against agile evaders?

Beyond the choice of algorithm, practical performance hinges on system-level decisions such as how deeply RL is integrated into the control stack, whether prior knowledge is used, and what sensory information the network to be learned receives. Understanding the impact of these design levers motivates the second research question,

Design Considerations Research Question

How do design choices in control abstraction, observation structure, and training methodology influence the performance of learned MAV interception policies?

Together, these questions structure the discussion in this chapter. Section 6.1 first presents the working principles of RL and how MAV-pest interception fits within this framework. Next, the first research question is addressed in section 6.2, which briefly evaluates both single-agent and multi-agent RL (MARL) algorithms to determine their effectiveness for agile MAV-based interception. The second question is explored in section 6.3, with subsections examining the role of prior knowledge, control abstraction depth, and observation structure in shaping policy performance.

6.1. Reinforcement Learning

RL is particularly well-suited for MAV-based pest interception tasks, where success depends on adaptive, long-horizon decision-making in dynamic and uncertain environments. This suitability stems from RL's ability to learn directly from interaction, allowing agents to optimize behavior based on cumulative reward without relying on predefined training data [20]. Its strength in handling sequential decisions, real-time feedback, and long-term planning further supports its application to agile aerial interception [21].

In recent years, deep RL (DRL), which integrates neural networks, has achieved impressive results across domains, outperforming state-of-the-art controllers and even human experts in complex environments [22]. Within drone control, RL has surpassed state-of-the-art controllers in both simulation and when deployed on hardware [23, 24, 25]. This success motivates its application for MAV-based pest interception.

6.1.1. Problem Formulation

In the context of MAV-based pest interception, RL provides a framework in which an autonomous agent, in this case the MAV, learns to generate interception behaviors through trial and error, guided by the goal of maximizing a cumulative reward. The agent learns by interacting with a simulated environment that mimics real-world aerial pursuit scenarios, receiving feedback based on the success or failure of its interception attempts.

Key components of this RL include the policy, reward, value function, and optionally an environment model. The policy maps from observations to actions, effectively defining the MAV's decision-making strategy. The reward quantifies the desirability of a particular state-action outcome. In the case of pest-interception this may be proximity to the pest, successful interception, or minimizing time-to-capture. The reward may be further shaped to encourage desirable emergent behaviors, such as sustained tracking, predictive maneuvering, or energy-efficient flight. The value function estimates the long-term benefit of a given state, enabling the agent to reason about future outcomes rather than just immediate gains. When used, a predictive model of the environment can further aid learning by simulating future dynamics [21], though model-free approaches are more applicable due to the complexity involved in modeling the aerial-to-aerial engagement.

Due to limited onboard sensing, noisy measurements, and the agile and often unpredictable behavior of real pests, the MAV does not have full access to the true system state. This motivates the use of a Partially Observable Markov Decision Process (POMDP) framework, which formalizes decision-making under uncertainty [20]. A POMDP is defined by the tuple given in Equation 6.1,

$$\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, Z, \gamma) \tag{6.1}$$

where, in the context of MAV-based pest interception, the state space $\mathcal{S}$ captures the full underlying configuration of the environment, including the positions, velocities, and orientations of both the MAV and the pest—though this state is not directly observable. The action space $\mathcal{A}$ defines the set of possible control inputs available to the MAV, which may vary depending on the level of control abstraction. The observation space $\mathcal{O}$ consists of measurable features derived from onboard sensors.

The transition function $P(s' \mid s, a)$ models the system dynamics, specifying how the state evolves in response to a given action, capturing both the MAV's motion and the behavior of the pest. The reward function $R(s, a)$ quantifies the immediate benefit of taking action a in state s, and may be shaped to encourage different emergent behaviors. The observation function $Z(o \mid s', a)$ accounts for sensor noise and partial observability, mapping the hidden state to a stochastic observation. Finally, the discount factor $\gamma \in [0, 1)$ governs the agent's temporal horizon, trading off immediate interception success against long-term tracking effectiveness.

6.2. Algorithmic Evaluation

The evaluation is divided into two parts. Subsection 6.2.1 examines single-agent RL, focusing on Proximal Policy Optimization (PPO). Its theoretical foundations, practical advantages, and suitability relative to alternatives like Deep Deterministic Policy Gradient (DDPG) are discussed.

Subsection 6.2.2 explores MARL, outlining key challenges and comparing decentralized and centralized training approaches. It evaluates Independent PPO (IPPO), Multi-Agent PPO (MAPPO), and contrasts MARL with classical differential game theory.

6.2.1. Single-Agent Reinforcement Learning

This section establishes the suitability of proximal policy optimization (PPO) for the task of aerial-to-aerial interception and introduces the working principles. A variety of RL algorithms have been applied to aerial-to-aerial interception problems. In particular, the Deep Deterministic Policy Gradient (DDPG) [26, 17, 18] and PPO [27, 28, 16] methods are frequently referenced in aerial interception research. A detailed discussion of the nuances of these RL algorithms is beyond the scope of this review. For an evaluation of the strengths and limitations of popular algorithms specifically in the context of aerial-to-aerial interception, the reader is referred to the review by Reinier Vos [29].

To summarize Vos's findings, DDPG offers higher sample efficiency, which allows for faster learning but at the cost of increased memory usage and training complexity. In contrast, PPO converges more slowly due to a conservative update mechanism. However, as noted by Vos [29], the simulated environments used in aerial interception controller training are computationally inexpensive, making the sampling efficiency advantage of DDPG less relevant. Furthermore, in the seminal work by Schulman et al. [30], PPO is shown to exhibit stable learning behavior with minimal hyperparameter tuning, making it an accessible choice. For these reasons, PPO is deemed a suitable and practical method for aerial-to-aerial interception tasks. The remainder of this section will present the working principles of PPO in more detail.

Proximal Policy Optimization

The formulation and explanation of PPO presented below follows from the seminal work of Schulman et al. [30]. PPO is an actor-critic algorithm within the broader class of policy gradient methods. Meaning, the algorithm directly optimizes a parameterized policy using data collected from agent-environment interactions, while leveraging a learned value function to reduce gradient variance and improve learning stability [20]. The objective to maximize for policy gradient methods is given in Equation 6.2,

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right] \tag{6.2}$$

where, $\pi_\theta(a_t \mid s_t)$ denotes the probability of taking action a_t in state s_t under the policy parameterized by θ, and $\hat{A}_t$ is an estimate of the advantage function, which indicates how favorable an action is compared to the expectation of the reward based on the state the agent is currently in [30]. This objective function represents the expected advantage-weighted log-probability of actions under the current policy. By maximizing this objective, the algorithm increases the likelihood of actions that lead to higher-than-expected returns. Note that this objective, encourages increasing the probability of high-advantage actions, nothing inherently limits how drastically the policy can change from one update to the next, resulting in large an unstable policy updates.

Schulman et al. [30] investigate different objective functions with constraints to address the limitations of the traditional policy gradient objective. Only the best performing objective function from the investigation is presented here. First, a probability ratio between the currently policy and old policy is defined as Equation 6.3,

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{6.3}$$

where θ_{old} refers to the weights of the parameterized policy before the last update was done. The varaibles $\pi_\theta(a_t \mid s_t)$, θ, and $\hat{A}_t$ are as previously defined in Equation 6.2. This ratio between the two probabilities quantifies how much more or less likely a given action in a given state has become under the updated policy. Using this probability ratio Schulman et al. [30] propose the following objective given in Equation 6.4,

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{6.4}$$

where ϵ is a hyperparameter that controls how much the new policy is allowed to deviate from the old policy during each update. The other parameters $\pi_\theta(a_t \mid s_t)$, θ, and $\hat{A}_t$ are as defined previously in Equation 6.2. Schulman et al. [30] explain that this objective is designed to balance effective policy improvement with stability. The first term inside the minimum encourages improving the policy by increasing the probability of advantageous actions. The second term applies a clipping function to the probability ratio $r_t(\theta)$, restricting it within the range $[1-\epsilon, 1+\epsilon]$. This prevents the policy from changing too drastically in a single update. By taking the minimum of the clipped and unclipped objectives, PPO ensures that the final loss is a pessimistic estimate, it limits the magnitude of beneficial changes and always accounts for harmful ones.

The behavior of the PPO objective with respect to the advantage function is illustrated in Figure 6.1. In the left plot, we see that the objective is clipped when a beneficial action becomes more probable under the new policy compared to the old one, resulting in more conservative updates. In the right plot, an action with a negative advantage that becomes more probable is heavily penalized. Conversely, a poor action that becomes significantly less probable under the new policy is clipped to further reduce its contribution, amplifying the penalty.

Other surrogate objectives were also explored in the work of Schulman et al. [30]. These different methodologies are generally grouped under the umbrella of PPO variants. While the clipped objective described here is widely adopted and represents the core idea of PPO, it's important to note that many implementations in common RL libraries differ slightly from the original formulation [31, 32, 33]. Shengyi et al. [33] explore the different PPO implementations available and note how their performance differs on benchmark tasks. Implementation differences identified can include variations in how the objective is applied, how advantage estimation is handled, or how clipping is implemented. Therefore, users must carefully inspect the source code of a given library to understand the exact PPO variant being used and how to assign the hyperparameters to ensure desired training behavior.

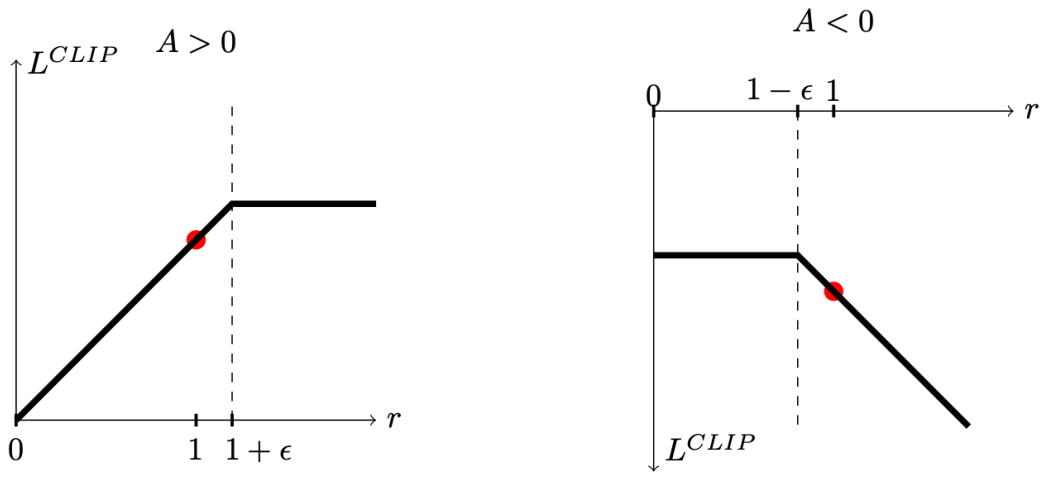


Figure 6.1: Plots showing one term, a single timestep of the surrogate objective L^{CLIP} as a function of the probability ratio $r_t(\theta)$, for positive advantages left and negative advantages right. Note that L^{CLIP} sums many of these terms. Image retrieved from [30].

6.2.2. Multi-agent Reinforcement Learning

Single-agent RL methods are capable of learning either the pursuer's [34, 19, 18, 17, 16, 35] or the evader's guidance [27]. A shortcoming of single-agent methods is that the policy learned does not take into consideration the interactive dynamics inherent in aerial-to-aerial interception scenarios. In the case of a single agent RL policy, the assumption of a static non-reactionary opponent during training will lead to overfitting and potentially fail to generalize to real world interception scenarios. Therefore, there exists a need for MARL which is able to explicitly model and learn the interactive behavior of the pursuer and the evader.

An in-depth evaluation of multi-agent algorithms is beyond the scope of this investigation; for a comprehensive analysis of MARL methods in the context of aerial pursuit and evasion, the reader is referred to the review by Vos [29]. This subsection introduces the challenges specific to MARL, presents common algorithmic approaches, and evaluates their applicability to MAV-based interception tasks.

Challenges of Multi-agent Reinforcement Learning

As synthesized in the review by Vos [29], which draws on the survey by Hernandez et al. [36], MARL introduces several challenges that are either absent or significantly amplified compared to the single-agent setting. A key issue is non-stationarity. In MARL, the transition and reward functions depend on the joint actions of all agents. As a result, the environment from the perspective of any single agent is constantly changing, since other agents are simultaneously learning and adapting. This violates the stationarity assumption underpinning many RL algorithms and complicates stable learning [36].

Another major difficulty is the curse of dimensionality. As the number of agents increases, the joint state and action spaces grow exponentially. This makes efficient learning more challenging, despite the use of powerful function approximators like deep neural networks [36, 37].

The credit assignment problem is also exacerbated in MARL. When multiple agents contribute to a shared outcome, it becomes difficult to determine which agent's actions were responsible for specific rewards. This ambiguity hinders effective policy learning, especially in cooperative or competitive settings where the influence of one agent's actions is entangled with others' [36].

Despite these challenges, MARL has demonstrated promising results in aerial interception tasks. Several studies have applied multi-agent learning to pursuit-evasion scenarios involving autonomous aerial vehicles, with success in developing robust interception strategies under dynamic, adversarial conditions [38, 39, 40].

Multi-agent Reinforcement Learning Alogirthms

The different MARL methods, may be broadly organized into a taxonomy based on how learning and decision-making are done. Firstly, decentralized methods ignore the non-stationary of multi-agent environments and train a policy for each agent independently. On the other hand, centralized methods learn a joint policy for all agents. The last approach combines the two with centralized training with decentralized execution (CTDE). This method combines global information during training with independent execution. Importantly, CTDE addresses the non-stationarity issue by stabilizing training with global views, while still allowing agents to act independently when transferring policies for implementation [41]. The differences between these methods is highlighted in Figure 6.2. For the purpose of learning an interception guidance controller, a centralized execution scheme is not viable, as the pursuer and evader must operate independently. Therefore, this review will exclusively present decentralized learning and CTDE-based methods.

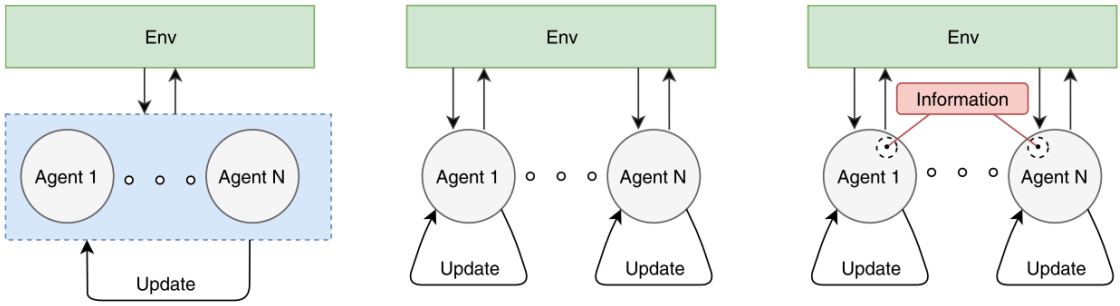


Figure 6.2: Training schemes in the multi-agent setting. (Left) centralized training and centralized execution holds a joint policy for all agents. (Middle) Each agent updates its own individual policy in decentralized training. (Right) CTDE enables agents to exchange additional information during training which is then discarded at test time. Image retrieved from [37]

Most single-agent RL algorithms can be naturally extended to the multi-agent setting through independent learning, where each agent learns its own policy independently of the others. A straightforward extension of PPO to the MARL case is Independent Proximal Policy Optimization (IPPO). This framework is fully decentralized: each agent learns its own PPO policy based solely on local observations and rewards.

The study by de Witt et al. [42] investigates and benchmarks the performance of IPPO compared to other independent and CTDE-based MARL methods. A key feature identified as contributing to IPPO's effectiveness is policy clipping, which is hypothesized to reduce the impact of environment non-stationarity, making some of the theoretical limitations of independent learning less severe in practice. Furthermore, Yu et al. compare the performance of IPPO with Multi-Agent PPO (MAPPO), which features a centralized critic. Their results show that even in continuous or communication-heavy domains, IPPO can perform on par with—or even better than—more specialized multi-agent algorithms [43].

Similarly, CTDE methods can be viewed as straightforward extensions of single-agent algorithms, where a shared centralized critic is introduced to incorporate global information during training. For instance, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) extends DDPG by providing each agent with a centralized critic that observes the global state and the actions of all agents, while the actor still relies only on local observations for execution [41]. CTDE-based methods are the dominant approach referenced in multi-agent tasks, including aerial pursuit and evasion [44, 39, 40]. Zhang et al. [45] show that centralized critics significantly boost performance in coordinated interception scenarios, where shared information is critical. Recent work by Chen et al. [38] demonstrated that MAPPO can effectively bridge the simulation-to-reality gap in multi-UAV pursuit tasks. However, as elucidated in the survey of Hernandez et al. [36], CTDE methods introduce several challenges. Centralized critics expand the state-action space, increase memory requirements, training instability and exacerbate the credit assignment problem.

Given the specific requirements of the interception guidance task—namely, a single pursuer operating independently against a reactive target—the added complexity of CTDE methods is not justified. The advantages of CTDE approaches are primarily observed in multi-agent scenarios where coordination or

information sharing is critical [45, 38], which is not the case here. In contrast, IPPO offers a significantly simpler and more scalable alternative, and has been shown to achieve competitive performance relative to more complex centralized methods, even in communication-heavy domains [43]. Furthermore, practical constraints such as limited support in existing MARL libraries [31, 32, 33] and the engineering burden associated with implementing centralized critic architectures reinforce the suitability of decentralized approaches. As such, IPPO is recommended as the most appropriate algorithm for training an autonomous MAV to perform one-on-one aerial interception, balancing performance, scalability, and implementation feasibility.

Differential Games of Pursuit and Evasion

An alternative approach to solving aerial-to-aerial interception problems where the interactive dynamics between the agents are taken into account are differential games of pursuit and evasion. While different variations of these games exist, the fundamental principle unifying them is the conflict between two agents seeking opposing objectives. This interaction is a zero-sum game, where the cost function of one player is the negative of the other. In the context of aerial engagements, the payoff is often measured in terms of time-to-intercept, where the pursuer tries to minimize this time, and the evader tries to delay capture as long as possible [46].

Differential games offer a principled and well-established way to model such interactions, and have historically been used to derive analytical solutions for interception scenarios [47, 48, 49, 50]. These solutions define optimal control strategies, where the pursuer nor evader can improve their outcome by deviating from their current strategy, given the strategy of the opponent [46]. However, deriving closed-form solutions is often challenging and they typically rely on simplified objective functions as well as numerous assumptions. As a result, solving games of pursuit and evasion becomes infeasible in realistic settings.

To address the limitations of differential games in realistic settings, MARL, offers a data-driven alternative that enables the approximation of optimal strategies without requiring analytical tractability. While MARL does not provide the same formal guarantees of optimality, such as provable convergence where neither the pursuer nor the evader can improve their outcome, it is capable of learning strategies that approximate these game-theoretic solutions. Crucially, MARL can incorporate the full complexity of real-world dynamics, that typically make differential game formulations analytically intractable. As a result, MARL can provide useful estimates of optimal behavior under realistic conditions, offering a viable solution where classical approaches are too restrictive to be applied [46].

6.2.3. Algorithm Choice

In response to the first research question, *Which RL algorithms are effective for enabling MAV-based aerial-to-aerial interception against agile evaders?*, this section summarizes the findings from the preceding evaluations. For the single-agent setting, PPO emerges as the most suitable algorithm. Despite its relatively slower convergence compared to DDPG, PPO's stability, robustness to hyperparameter tuning, and wide adoption in both academic literature and RL libraries make it a practical and effective choice. Moreover, the low computational cost of simulation environments for aerial interception diminishes DDPG's advantage in sample efficiency, further supporting the preference for PPO [30, 29].

In the multi-agent setting, IPPO is identified as the most appropriate algorithm. It combines the core advantages of PPO with a fully decentralized training approach, allowing each agent to learn independently while still achieving competitive performance relative to more complex CTDE-based methods [42, 43]. Given the one-on-one nature of MAV-pest interactions and the lack of coordination requirements, the simplicity, scalability, and ease of implementation make IPPO the most viable option.

6.3. Design Considerations

While algorithm selection is critical, the effectiveness of RL-based interception is equally shaped by architectural and methodological decisions. These include how prior knowledge is incorporated, the level of abstraction at which control is applied, and the structure of the agent's input observations. Each of these choices affects training efficiency, generalization, and real-world deployment.

To explore these dimensions, the section is organized as follows: subsection 6.3.1 examines the use of transfer learning, imitation learning, and curriculum learning to accelerate convergence and improve

sample efficiency. subsection 6.3.2 considers the level of control abstraction—from high-level kinematics to low-level motor commands—and its implications for policy expressiveness and deployment. In subsection 6.3.3, the design of input observations is analyzed. Finally, subsection 6.3.4 synthesizes these insights and outlines promising directions for future work.

6.3.1. Leveraging Prior Knowledge

Training RL agents entirely from scratch is often prohibitively expensive in terms of both sample efficiency and training stability. To address this, a variety of methodologies exist for leveraging prior knowledge to accelerate learning. These include transfer learning, imitation learning, and curriculum learning, each offering complementary benefits depending on the task formulation and available data.

Transfer learning refers to the process of reusing knowledge gained from one task to improve learning performance on a related task. This methodology is particularly well-suited to MAV control, where policies must often first acquire low-level flight skills before more advanced behaviors such as interception can be learned. A frequently encountered challenge in this context is the so-called “learning to fly” problem, wherein the agent must implicitly learn basic flight stabilization before it can begin to address higher-level objectives. This problem has been explicitly discussed in multiple studies [51, 52, 53], and is commonly addressed by first training a stable controller in simulation and then reusing this policy as the initialization for downstream tasks. A breakthrough is provided by Eschmann et al. [51], who present a novel framework to overcome the ”learning to fly” problem with only a few seconds of training time required. In the context of MAV-pest interception, this approach can be employed by first developing a base navigation or stabilization controller, which can then be fine-tuned for interception maneuvers. This significantly reduces the exploration burden and lowers the likelihood of catastrophic failure during early training phases.

Another promising methodology is imitation learning, which bypasses the need for extensive exploration by allowing the agent to learn directly from expert demonstrations. As reviewed by Osa et al. [54], imitation learning algorithms infer control policies by mimicking observed expert behavior. The most widely used technique, known as behavior cloning, formulates this problem as supervised learning, where the policy is trained to reproduce expert actions given specific states. This provides a strong initialization that can be further refined through RL, combining the stability of supervised learning with the adaptability of policy optimization. In the context of aerial interception, expert demonstrations could be obtained through classical guidance laws, such as PN, or from human-piloted drone trajectories. By starting with such a policy, the agent is likely to converge more quickly and avoid suboptimal local minima.

Curriculum learning presents a third strategy for improving training efficiency by structuring the learning process through a progressive sequence of tasks. Originally proposed by Bengio et al. [55], curriculum learning draws inspiration from the way humans and animals learn—beginning with simple concepts and gradually progressing to more complex ones. The authors show that this structured approach can accelerate convergence and help optimization avoid poor local minima. In the case of MAV-pest interception, a curriculum could be designed to incrementally increase task difficulty. Training might begin with interception of stationary targets, followed by slow-moving or non-reactive evaders, before ultimately addressing the full pursuit-evasion problem involving agile, adversarial behavior. This gradual increase in complexity allows the policy to build upon simpler skills and adapt them to increasingly dynamic and uncertain environments.

Despite their advantages, these techniques also introduce important considerations. By learning auxiliary tasks such as hovering or imitation of suboptimal expert strategies, the policy may become biased toward behaviors that are not fully aligned with the final objective. This can limit the expressiveness or optimality of the learned policy in the target task. Furthermore, implementing these techniques typically requires additional infrastructure, such as expert data pipelines, staged training environments, or fine-tuning procedures. In some cases, this added complexity may offset the gains in learning efficiency, particularly when the base task is already well-posed for direct policy optimization. Therefore, while transfer learning, imitation learning, and curriculum learning offer valuable tools for accelerating and stabilizing training, their use must be carefully balanced against their potential to constrain final policy performance or complicate implementation.

6.3.2. Depth of Reinforcement Learning Control

The depth at which RL is applied in UAV systems significantly impacts the performance and the complexity. Eschmann et al. [51] present a detailed taxonomy of control abstraction levels for multirotor UAVs, which organizes control inputs by their level of abstraction and associated system complexity. This taxonomy is presented in Figure 6.3. This hierarchy spans from high-level kinematic inputs such as position, velocity, and acceleration, down to increasingly fine-grained and physically grounded levels including collective thrust and body rates (CTBR), body torques, rotor thrusts, RPMs, and ultimately, motor commands. As one descends the hierarchy, each level introduces more non-linearities, integration steps, and domain parameters, leading to increased uncertainty and modeling difficulty.

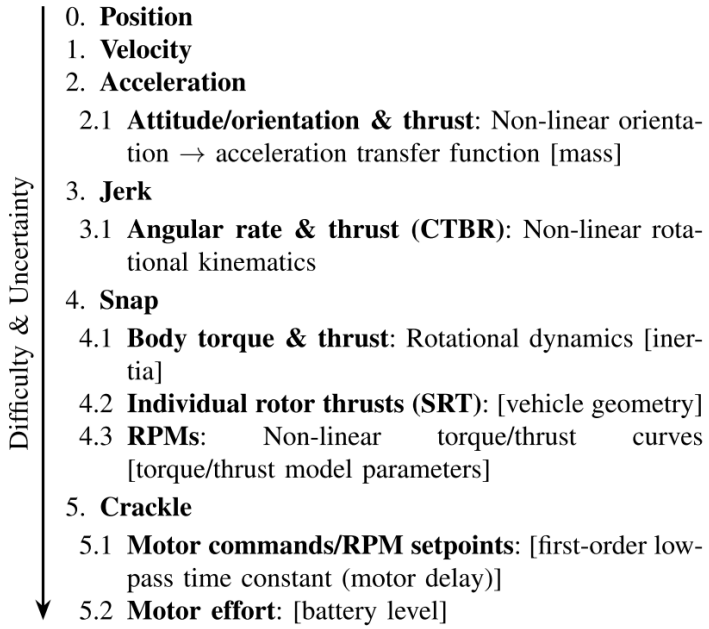


Figure 6.3: Taxonomy of multirotor control abstractions. Additional system/domain parameters introduced at an abstraction level are given in square brackets. Image retrieved from [51]

Despite the rising complexity, Eschmann et al. [51] argue that RL is particularly well-suited for learning policies at these lower levels of control. According to Bellman's Principle of Optimality, the optimal policy for a control problem can be expressed as a function mapping from observations to actions. In theory, a neural network can represent this mapping even in the presence of complex, non-linear dynamics. By contrast, classical control stacks, which are built upon layered abstractions, have limited expressivity and may struggle to represent globally optimal behaviors. While optimization based controllers can approximate optimality, they are often too computationally intensive for real-time execution on low-power embedded hardware. Therefore, Eschmann et al. [51] advocate for lower RL as a promising approach to learn efficient, end-to-end control policies that directly map the system state to low-level motor commands, maximizing expressivity while maintaining real-time viability.

An early systematic investigation into this control abstraction is presented by Kaufmann et al. [56], who bench-marked the performance of different action abstractions in simulation. Their results show a major leap in policy performance when shifting from higher-level abstractions such as linear velocity commands to lower-level control inputs like CTBR. The authors explain that this representation better preserves the drone's dynamic capabilities while maintaining sufficient abstraction to ease sim-to-real transfer.

At the time, however, methods for fully end-to-end RL control, networks outputting direct motor commands, were not yet robust for deployment on physical systems. Since then, advances in sim-to-real strategies have enabled successful real-world deployment of end-to-end RL controllers in challenging domains like drone racing. In recent work, Ferede et al. [22] demonstrate that an end-to-end policy for time-optimal quadcopter racing can outperform a CTBR policy with an incremental nonlinear dy-

namic inversion (INDI) lower level controller in simulation by 1.39 seconds and by 0.17 seconds in real-world testing. Notably, the performance gap observed between simulation and real-world deployment in end-to-end systems highlights the importance of addressing sim-to-real transfer. The problem of discrepancy between simulation and real life for RL policies is discussed later in detail in chapter 7.

More recently, Ferede et al. [57] further validated the power of end-to-end control by demonstrating a single neural network policy capable of generalizing across different racing drones. Although this work highlights the high performance and generalization potential of end-to-end methods, it does not compare results directly to a CTBR baseline, leaving open the question of how much performance is gained over these established architectures. In summary, while guidance-level RL is widely used for aerial interception due to its modularity and ease of integration, recent advances suggest that lower RL control, particularly end-to-end motor command policies, can unlock higher performance albeit at increased implementation complexity.

6.3.3. Input Observations

The choice of information made observable to the agent has a critical role in shaping the learned behavior, interpretability and the performance [58]. Across recent studies involving aerial-to-aerial interception, a wide spectrum of input designs can be observed. Some approaches expose the agent to low-level, raw measurements [34] [27] [19]. In these cases, the RL policy may implicitly learn the relevant control features. In contrast, some approaches rely on preprocessed or filtered inputs, where features like relative velocity, acceleration, LOS rate, or target motion are derived and directly provided to the RL agent [18] [19]. This reduces the burden on the policy to infer latent variables, but it also risks overly constraining the agent.

Dionigi et al. [58] conducted a benchmark study on how different input configurations affect the performance of RL control policies for quadrotor flight. Several RL agents with varying observation inputs were trained in simulation and then evaluated their zero-shot sim-to-real transfer capabilities on a real drone to determine which configurations lead to robust and efficient control. The authors concluded that the observation space containing the minimum necessary information led to the best overall performance. Providing additional input data did not improve results and sometimes even reduced performance. While the findings of Dionigi et al. offer valuable insight into observation space design, their conclusions are drawn from a relatively simple stationary point tracking task. This setup does not capture the dynamic, nature of aerial-to-aerial interception, where richer or more predictive input features may be critical for performance. Extending the analysis to more complex tasks presents an interesting direction for future research.

A common practice when designing the observation space is to represent the quadcopter's attitude and rotational rates using a rotation matrix [58, 59, 56, 51]. This choice is primarily motivated by findings such as those of Zhou et al. [60], which highlight that commonly used representations like Euler angles or quaternions are discontinuous and can hinder the learning process in neural networks. While using a rotation matrix may therefore facilitate more stable or efficient training, its effect on final task performance has not been established. Notably, the current state-of-the-art in drone racing [57] relies on Euler angles, suggesting that are no performance gains from using rotation matrices.

Another best practice identified by Dionigi et al. [58] is the inclusion of historical data in the observation space. Specifically, using a sequence of the last 10 observations was shown to significantly improve training convergence and performance, with results indicating that shorter histories failed to achieve similar effectiveness. However, it is worth noting, based on my own interpretation, that the effectiveness of using a 10-step history is inherently linked to the temporal resolution of the control loop. If the control loop runs at a higher or lower frequency, the time span covered by these 10 steps would differ accordingly, potentially impacting the relevance and utility of the historical data.

Supporting the inclusion of historical data, a study by Eschmann et al. [51] highlights that actuator dynamics, particularly motor lag, introduce a substantial delay between when an action is issued and when it affects the system state. This delay renders the environment partially observable, as the immediate state no longer fully reflects the impact of recent control inputs. To mitigate this, they include a history of control actions in the observation space, serving as proprioceptive memory to improve credit assignment and learning stability.

A contrasting result regarding the observation history is presented in the study by Ferede et al. [57]. Historical observations were briefly explored as potential input augmentations to the neural control policy. However, they found that including up to the last 3 time steps did not yield any significant performance improvement and therefore excluded them from the final architecture. This stands in contrast to the findings of Dionigi et al. [58] and Gronauer et al. [61], who demonstrate that including a longer temporal history, specifically more than the last 5 observations, is necessary to achieve meaningful performance gains. This discrepancy suggests that the benefit of historical data may depend on the task dynamics and observation structure, and that further investigation into the role of temporal context is warranted.

An emerging trend is the integration of auxiliary networks and recurrent architectures to enhance input observations. Li et al. [16] propose an assisted RL approach that incorporates an auxiliary network trained through supervised learning to estimate the target's acceleration. This estimated acceleration is then used as an additional observation input to the RL agent . Similarly, Chen et al. [38] use a LSTM network to predict the evaders future trajectory based on historical observations. This leverages the ability to forecast the evaders behavior to generate better policies for interception.

Ultimately, the decision of what state information to expose—whether raw, derived, or learned—reflects a trade-off between generalization, interpretability, and learning complexity. It is a fundamental design choice that must align with the system architecture, sensing capabilities, and deployment goals of the interception system.

6.3.4. Design Recommendations

In response to the second research question, *How do design choices in control abstraction, observation structure, and training methodology influence the performance and robustness of learned MAV interception policies?*, this section summarizes the most salient insights and identifies promising directions for further research.

Across all three design dimensions examined, leveraging prior knowledge, control abstraction, and input structure, evidence suggests that thoughtful architectural choices can significantly improve learning efficiency and performance. In particular, the use of transfer learning and imitation learning has been shown to accelerate convergence and overcome early instabilities associated with low-level control, such as the "learning to fly" problem [52, 53, 51]. Similarly, curriculum learning offers a principled way to scaffold task complexity, allowing agents to progressively master interception in increasingly dynamic settings [55]. While these methods reduce sample complexity, their reliance on auxiliary objectives or expert data introduces the risk of suboptimal policy bias and increased implementation burden.

With respect to control abstraction, recent work indicates that lower RL control—especially end-to-end policies that directly output motor commands—can offer superior performance and generalization in high-speed, dynamic tasks [22, 57]. However, these benefits come at the cost of higher implementation complexity and more difficult sim-to-real transfer.

Observation design remains an equally influential factor. Minimal yet informative state representations have been shown to enhance robustness and sim-to-real transfer in basic control tasks [58], while in interception scenarios, additional structure—such as history windows or auxiliary prediction modules—can improve credit assignment and temporal reasoning [51, 38]. However, findings remain mixed, with some high-performance policies succeeding without temporal history [57].

In summary, effective MAV interception policies benefit from modular use of prior knowledge, increased control expressiveness, and carefully structured observations. Nonetheless, many of these design practices remain underexplored in the context of interception, and more empirical work is needed to evaluate their interaction effects. Future investigations should prioritize comparative studies across abstraction levels and input encodings, particularly in dynamic and adversarial environments, to better understand how design decisions translate to generalization and real-world reliability.

7

Reality Gap

In the context of MAV-based moth interception, simulators play a critical role in the development and verification of autonomous control strategies. They enable safe, cost-effective testing of both classical control laws and learned controllers, all within a controlled and repeatable environment. However, simulators inevitably rely on simplified or idealized models of MAV dynamics, pest behavior, and environmental interactions. These simplifications result in what is known as the reality gap (RG), a discrepancy between the behavior and performance in simulation and their actual performance in real-world conditions [62]. In pest interception tasks, where high agility of the target and precise maneuvering are required due to the pests' small size, even minor discrepancies in flight dynamics, sensor noise, or environmental variability can cause strategies that perform well in simulation to degrade or fail when deployed in physical greenhouse settings.

While the reality gap poses challenges for all control strategies, its impact is particularly pronounced for RL controllers. Learned policies are tightly coupled to the simulator used during training. The controller's behavior is effectively shaped by the fidelity of the simulated environment. This dependency makes sim-to-real transfer a critical bottleneck in RL-controlled MAV-pest interception. The most consequential failure, an unsuccessful sim-to-real transfer, occurs when a policy that appears successful in simulation cannot generalize to real-world conditions [62]. Therefore, this section addresses the following research question,

Reality Gap Research Question

What methods can be used to reduce the reality gap and increase the likelihood of successful sim-to-real transfer for learned controllers in MAV-pest interception tasks?

To provide an answer to this question, first a formal definition of the RG is given. Next the current state-of-the-art method to solving the RG problem are discussed, where present also relating them to applications in interception tasks. Finally, an evaluation of the methods is given focusing on methods that may be practically implemented and be used to predict the sim-to-real discrepancy to be expected.

7.1. Reality Gap Formalization

The formalization of the RG problem developed by Salvato et al. [62] is presented here. The concept of an environment mapping operator, denoted as ϕ, is introduced. This operator is assumed to transform the real environment E into an approximate, simulated counterpart $E' = \phi(E) = (X', A', O', f', g', h')$, where X', A', and O' represent the simulated state, action, and observation spaces; f', g', and h' denote the dynamics, observation model, and reward function, respectively. Although ϕ is typically unknown in practice, it serves as a conceptual tool to understand the modeling process in sim-to-real transfer. Under this framework, the real-world system is modeled as a digital approximation, with the expectation that a control policy trained in simulation (E') can be deployed on the real system (E), this transfer is visualized in Figure 7.1.

This modeling paradigm naturally splits the controller development process into two stages. Firstly, an agent L interacts with the simulated environment E' to learn a policy π^*; and secondly, the learned policy π^* is then deployed and evaluated on the real environment E. The RG emerges from the discrepancies between E and E', which may cause π^* to under perform or fail when transferred to the real system [62].

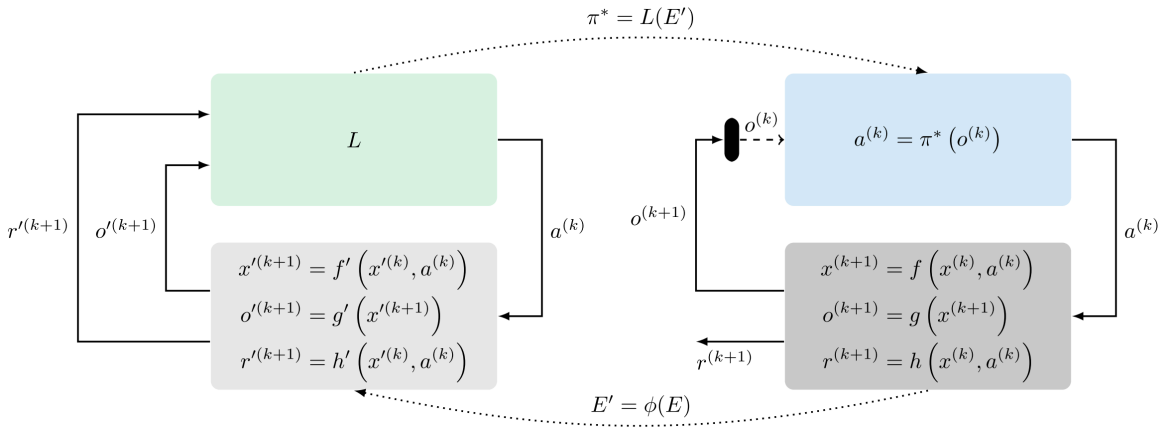


Figure 7.1: Schematic representation of the application of RL to the robot control problem using simulation. First, given the real robot and it environment E (block at bottom right), a simulator E (block at bottom left) is obtained by modeling E using ϕ. Then an agent L (block at top left) learns a policy in simulation. Finally, the learned policy is transferred to the real robot where it can be deployed (block at top right). Image retrieved from [62].

If the performance achieved in simulation is acceptable, but significantly degrades during real-world implementation, this indicates the presence of a RG. Salvato et al. [62] identify that possible causes of the RG can be attributed to two main factors. Firstly, the environment mapping operator ϕ may be unrealistic. The differences between E and E' in terms of their dynamics functions f, f' and/or observation functions g, g' may lead to discrepancies in observations. Secondly, the learning agent L may fail to produce a policy π^* that is sufficiently robust to cope with small but inevitable inaccuracies in the mapping ϕ. Various methods exist to overcome the RG, the remainder of the section shall be dedicated to reviewing the most popular methods.

7.2. Transfer Learning

Building on the principles discussed in subsection 6.3.1, transfer learning may also be used for crossing the reality gap. A controller is first trained in a simulation environment before being trained further in a real-world setting [62]. However, for aerial-to-aerial interception tasks, the conventional two-phase transfer learning approach, wherein the simulation policy is fine-tuned through additional real-world training, proves impractical. Drones are vulnerable to abrupt failures, hardware damage, and limited battery life restricts the feasibility of extended on-hardware training.

For the reason stated above the preferred method for policy transfer is zero-shot transfer. In this case the simulation learned policy is transferred directly to the hardware, without any additional training or measures being taken [59]. Therefore the remainder of the methods that will be presented are focused on enhancing zero-shot transfer of policies.

7.3. System Identification

System identification seeks to refine the simulator dynamics f' by fitting model parameters to real flight data, thereby closing the gap between simulation and reality [63]. In their survey of sim-to-real techniques, Zhao et al. [64] list system identification as a consistently effective practice: the higher the simulator's fidelity, the greater the chance that a policy will transfer zero-shot to hardware.

Achieving such fidelity for quadrotors is, however, notoriously difficult. The vehicle's aerodynamics are dominated by complex rotor and body wake interactions that defy simple analytical treatment. Sun

et al. [65] address this with a graybox model that blends first-principles rotorcraft theory with wind-tunnel data from high-speed flight, cutting force and moment errors to 1–3% NRMS. Yet their models contain more than ten empirical regressors with opaque physical meaning. By contrast, the current drone-racing world record was set with an end-to-end RL policy trained in a simulator that captures only rigid-body dynamics plus coarse aerodynamic damping [57]. This result demonstrates that the importance of system identification may be limited, making a moderately accurate, interpretable model a pragmatic choice.

Practical obstacles further limit how far system identification may be pushed. Greenhouse interception flights face wide swings in temperature, humidity, and airflow, all of which shift vehicle parameters and sensor characteristics. Minor hardware changes such as fresh propellers or IMU calibration can likewise invalidate a finely tuned model, necessitating repeated data-collection cycles. Moreover, high-fidelity simulators are computationally expensive; when millions of episodes are required, a lower-fidelity model that enables faster iterations may yield better overall progress [62].

In summary, system identification is a valuable starting point for sim-to-real transfer, but its benefits must be weighed against modeling complexity, environmental variability, and computational cost. A well-calibrated, lightweight model combined with the reality-gap-bridging strategies presented in the sections that follow offers the best balance of accuracy and training speed.

7.4. Domain Randomization

Domain randomization is a technique based on the idea that, rather than attempting to precisely model all aspects of the real-world environment, one can instead intentionally introduce extensive variability into the simulation. By randomizing physical and sensory parameters during training, the simulation can encompass a broader range of possible real-world scenarios, thereby increasing the likelihood that the trained policy will generalize well to the real environment, despite the inherent modeling bias between simulation and reality [66]. The intuition behind domain randomization is visualized in Figure 7.2.

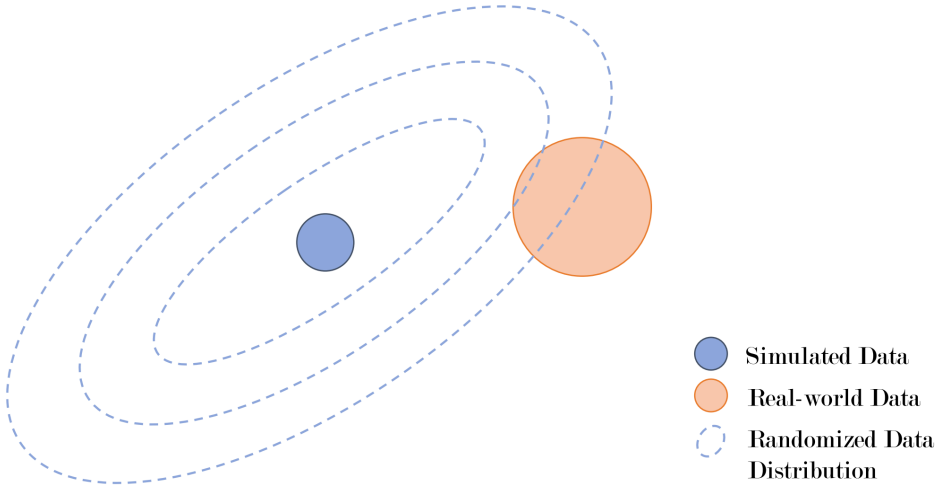


Figure 7.2: Working principle of domain randomization. Image retrieved from [64]

Following the formalism introduced by Salvato et al. [62], the parameters ξ' and ψ' are used to induce controlled variability in the simulation environment, resulting in a corrupted simulator $\widetilde{E}' = \tilde{\phi}(E)$. During training, the learning agent L interacts with multiple variations of $\widetilde{E}'$ by sampling based on ξ' and ψ', and learns a policy π^* that maximizes the expected cumulative reward across these variations, as shown in Equation 7.1,

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{\xi', \psi'} \left[J_{\pi, T} \right] \tag{7.1}$$

where $J\pi, T$ denotes the expected return of policy π over time horizon T. This procedure aims to produce a policy π^* that is robust to the variations introduced by domain randomization. If these variations sufficiently approximate the uncertainties present in the real environment, then π^* is expected to be sim-to-real transferable, effectively mitigating the RG. However, one of the key trade-offs of DR is its impact on computational efficiency. Since the training process must span a wide range of randomized environments, the agent requires significantly more interactions to converge to a robust policy. Each training episode may sample a different configuration of dynamics, sensor noise, or environmental conditions, which increases the variance of the learning signal and slows down convergence. As a result, while domain randomization enhances generalization, it does so at the cost of greater computational demand and longer training times [62].

Different methods for performing DR exist, varying in complexity. A study by Valassakis et al. [67] compares several DR techniques and evaluates their performance after transfer on three different tasks increasing in complexity. The key takeaway from their work is that no single DR method has a higher success rate in completing the tasks. A straightforward approach, such as injecting random disturbances, can achieve performance comparable to more sophisticated strategies like full dynamics parameter randomization or a comparable method of enhancing sim-to-real transfer like improved SysID. Notably, this simpler method requires considerably less time and engineering effort, making it a highly practical choice.

While the purpose of DR is to improve the robustness and generalization of learned policies, several studies show it can be unnecessary or even detrimental. The results of Chen et al. [59] and Molchanov et al. [68], indicate that policies trained on well-calibrated dynamics using SysID, without any domain randomization, consistently outperform those trained with DR. They conclude that this is because DR unnecessarily enlarges the policy search space, making learning more complex and potentially degrading both simulation and real-world performance. However, when certain dynamics parameters cannot be reliably identified or calibrated, DR remains a valuable fallback, introducing the variability needed to train more adaptable and robust policies.

A recent study by Ferede et al. [57] offers a more nuanced view by systematically examining the effect of varying degrees of DR on policy performance and generalization. Their findings are particularly relevant because they explore not only sim-to-real transfer but also cross-platform generalization in drone racing. The study shows that performance degrades as the degree of randomization increases, policies trained with no DR achieved higher rewards and faster flight speeds in simulation than those trained with DR. However, a small amount of DR proved critical for successful transfer, as policies trained without any randomization failed to operate on real hardware, while those trained with just 10% model parameter variation were able to do so effectively. Despite this, a significant sim-to-real performance gap remained: all policies exhibited considerably lower rewards and flight speeds in real-world testing compared to their simulated performance, regardless of the randomization level. Importantly, the study also demonstrates that DR facilitates hardware generalization, a single policy trained with sufficient randomization could successfully control both 3-inch and 5-inch quadrotor with a small amount of performance losses compared to policies fine-tuned to each platform. These results highlight the delicate trade-off involved in applying DR: while it is essential for enabling sim-to-real transfer and broadening policy applicability, excessive randomization can hinder learning efficiency and performance.

While the ability of a single, generalized policy to control multiple drone platforms is impressive, such generalization may not be ideal for highly competitive tasks like drone racing. In contrast, for aerial-to-aerial interception for pests, where robustness and ease of deployment may outweigh the need for absolute peak performance, the trade-off introduced by such generalization could be acceptable.

The study by Ferede et al. unfortunately does not examine the interaction between domain randomization and control abstraction levels. Specifically, it remains unclear how DR affects the performance and transferability of policies operating at the CTBR level with an INDI lower-level controller. This is a notable gap, especially considering that in their earlier work [22], the performance gain of an end-to-end policy over a CTBR with INDI controller in the real world was only 0.17 seconds, raising the question of whether such marginal gains justify the added complexity of end-to-end learning with DR when more abstracted control might suffice.

An important limitation of standard domain randomization is its inability to adequately capture rare but

critical variations in task-relevant scenarios. To address this, Chen et al. [38] introduce an environment adaptation strategy that dynamically perturbs and evolves the initial positions of both the UAV and the evader throughout training. By actively sampling challenging start configurations rather than relying on randomization, this method not only improves sample efficiency but also ensures robust performance on corner-case geometries that naïve DR might miss. Consequently, Chen et al. [38] show that for tasks with strong initial-condition sensitivity, such adaptive scenario generation can outperform both uniform DR by targeting the most informative variations.

In conclusion, for MAV-based pest interception, where robustness and transferability may take precedence over peak performance, lightweight domain randomization emerges as a practical and effective strategy. It offers a good balance between generalization and learning efficiency without incurring the complexity and training instability of more heavily weighted DR. However, current literature lacks systematic evaluation of DR's impact at different control abstraction levels, this provides an interesting avenue for additional research.

7.4.1. Privileged Reinforcement Learning

Using DR comes at the cost of significantly increased training time and sample complexity. To mitigate this computational burden, Pinto et al. [69] introduce an asymmetric actor–critic architecture that exploits privileged simulator information during training without altering the deployment policy.

In this setup, the actor network receives only the partial, noisy observations available at deployment, while the critic has access to the full, noise-free simulator state. By decoupling inputs, the critic learns more accurate value estimates from privileged data, which yields stronger gradient signals to the actor and accelerates convergence. At deployment, only the lightweight actor is used, relying solely on realistic observations, so there is no added inference cost. This simple yet powerful modification retains the robustness benefits of DR while alleviating some of the extra computational overhead.

7.4.2. Adversarial Reinforcement Learning

Adversarial RL (ARL) is similar to the domain randomization in the sense that it aims to induce disturbances during training. However instead of relying on fixed or random perturbations, ARL learns an adversarial policy that actively seeks to challenge the agent, forcing it to develop strategies that are robust under a wide range of disruptions [62].

An implementation of this idea is proposed by Pinto et al. [70] is Robust ARL (RARL). In RARL a protagonist and an adversary are defined. The system is modeled as a two-player zero-sum game, with both the protagonist and adversary observing the shared state and selecting their respective actions at each timestep. The actions of the adversary correspond to disturbances applied to the system. The protagonist maximizes the cumulative reward, while the adversary minimizes it.

Training a policy counteracted by an adversary ensures that the resulting policy is exposed to the worst-case disturbances. A key advantage is that the learned policy has been exposed to disturbances that are actively chosen to be difficult, contrary to domain randomization which inputs disturbances randomly. However, Pinto et al. [70] caution that setting the adversary's influence too high can bias the training distribution toward near-impossible disturbances, leading to unstable learning or outright failure to converge. In extreme cases, an overly strong adversary may consistently prevent the protagonist from making any progress, effectively blocking the policy from ever learning the task. Consequently, careful tuning of the adversary's capabilities is essential to balance robustness gains against training stability.

7.5. Reward Shaping

Reward shaping involves modifying the reward function to guide learning and improve the zero-shot sim-to-real transferability of learned policies [64]. It has become a widely adopted strategy across recent work in quadrotor control [57, 59, 51, 25], yet its implementation varies significantly depending on task and control abstraction.

Two primary forms of reward shaping emerge in the literature. The first is a penalty on the rotational rates of the multirotor. This is used in both CTBR-level controllers [25] and end-to-end policies [57]. However, the justification for this term remains loosely established in the sources reviewed; it is often

included without ablation or theoretical backing. The second and more explicitly motivated form of shaping is a penalty on the difference in action between consecutive time steps, intended to discourage abrupt control changes. This smoothness shaping is relevant because in simulation training policies learn to exploit unrealistically aggressive actions. While such behavior performs well in simulation, it leads to instability or infeasibility when deployed on real hardware [59, 38].

Despite its utility, reward smoothing introduces a notable trade-off. Penalizing abrupt changes in control commands generally enhances real-world stability but can reduce agility. Chen et al. [59] demonstrate this trade-off through a systematic study of CTBR policies, testing various smoothness reward formulations. Their results show that the penalty on successive action differences yields the best real-world performance on a trajectory tracking task.

This trade-off is formalized in the reward function shown in Equation 7.2,

$$r = r_{task} + k_{smooth} \cdot r_{smooth} \tag{7.2}$$

where r_{task} denotes the primary task reward, and r_{smooth} represents the smoothness penalty term. The coefficient k_{smooth} adjusts the influence of the smoothing penalty relative to the task objective. Since both reward components are normalized to the range [0,1], k_{smooth} directly defines the relative importance of smooth control behavior in the learning objective. Choosing an appropriate value for this coefficient is critical: too high, and the policy becomes overly conservative; too low, and it may adopt unrealistic, unstable behaviors that fail in the real world.

Chen et al. [59] also explore non-reward-based alternatives for smoothing, such as action clipping and low-pass filtering. While low-pass filtering enforces smoothness by design, it fails to generalize across different velocity regimes. Action clipping, meanwhile, limits peak actuator demands but sacrifices high-speed responsiveness. What the study of Chen et al. [59] finds is that reward-based smoothing via action-level penalties proves more effective than either alternative, particularly for aggressive trajectories.

Nevertheless, shaping the reward function is not a guaranteed solution for transferability. In some cases, a policy may still choose large action jumps if the gain in task reward outweighs the shaping penalty. Thus, while action penalties may encourage smooth behavior, they cannot enforce it. This opens the door to architectural solutions, such as filtering control outputs during both training and deployment. However, as Chen et al. [59] note, such methods also suffer from generalization limitations and have been shown to impair performance on high-speed tasks.

Despite extensive empirical use, several gaps remain in the literature. First, the necessity and effectiveness of body rate penalties are not well understood and are applied without justification. Second, the reward shaping studies, are limited to CTBR-level control. Therefore, how these strategies apply across different control abstractions remains underexplored. Finally, the interaction between reward shaping and other sim-to-real strategies, is poorly understood. A more systematic evaluation of reward shaping's role in transferability represents an interesting direction for future research.

7.6. Recommendations for Bridging the Reality Gap

The review above highlights that while numerous methods exist to mitigate the reality gap, all come at the cost of increased learning complexity or computational demand, a fundamental trade-off identified by Salvato et al. [62]. Despite this, for RL-based MAV control in moth interception scenarios, certain best practices have emerged that strike a balance between robustness and feasibility.

A practical and effective zero-shot sim-to-real transfer strategy begins with lightweight system identification. Capturing key MAV dynamics and sensor models while avoiding over-complexity. Although perfect modeling is unattainable and often impractical in dynamic greenhouse conditions, even coarse calibration of simulation parameters significantly improves the transferability of learned policies [62].

Once a well-calibrated simulator is established, DR should be applied to capture residual uncertainties and mismatches in the model. Crucially, this randomization should be light and focused solely on small variations in parameters [57]. More aggressive randomization strategies can unnecessarily expand

the policy search space, reducing convergence speed without meaningfully improving robustness, as cautioned by Salvato et al. [62] and supported by Chen et al. [59]. Moreover, as was determined by Valassakis et al. [67] for physical parameters, injecting simple random noise is often as effective as more sophisticated perturbation schemes.

Scenario adaptation emerges as a particularly promising complement to DR. Especially for interception tasks where success depends heavily on initial geometry, actively varying the start conditions during training—rather than relying on uniform randomization—results in greater robustness to the full distribution of encounter configurations [38]. This method has the added benefit of improving sample efficiency by targeting high-variance edge cases rather than wasting training episodes on uninformative states. Moreover, to offset the increased training burden introduced by DR, it is beneficial to employ privileged RL, such as asymmetric actor–critic methods introduced by Pinto et al. [69]. By using full state information for the critic during training while keeping the actor reliant on realistic observations, this approach accelerates learning without adding runtime complexity to the deployed policy.

Lastly, reward shaping through control smoothness penalties may be introduced to prevent unrealistic actuator commands observed in simulation-trained policies. This is especially useful on lower-performance platforms where aggressive actions cause instability or are infeasible in practice [59, 38]. However, this comes at the potential cost of reduced agility—highlighting that the degree of shaping should be tailored to the capabilities of the target hardware.

8

Conclusion

This literature review set out to answer the research question:

Literature Review Research Question

What integrated guidance and control strategies are effective for MAV interception of agile pests, and what strategies are used to aid transition from simulation to deployment in real-world environments?

In pursuit of this goal, chapter 4 established the foundational kinematic framework for MAV-to-pest interception by deriving equations of motion in a line-of-sight coordinate frame, enabling precise modeling of relative motion and interception geometry. It introduced a taxonomy of pursuit strategies along with diagnostic cost functions to evaluate controller adherence, especially for complex controllers or learned policies. Additionally, the chapter defined performance metrics such as time to interception, interception rate, near-miss time, and proximity tracking to assess real-world interception effectiveness. These tools collectively enable rigorous evaluation of both traditional and learning-based guidance and control.

Next, chapter 5 investigated the suitability of various PN guidance laws for MAV interception of agile aerial pests. Classical PN strategies, while computationally efficient and rooted in proven missile guidance paradigms, were shown to be limited in their responsiveness, re-engagement capability, and robustness to sensor noise and actuator delay. In contrast, modern adaptations—LPN and FRPN—offered significant advantages due to their use of relative Cartesian information and independence from noisy angular measurements. FRPN emerged as the top-performing strategy across diverse scenarios, exhibiting low time to interception, and resilience to degraded initial conditions. Global modifications like GRTPN also showed promise by enabling consistent re-engagement via a velocity-convergence term. Future research should prioritize implementing FRPN and GRTPN in real-world systems, exploring hybrid strategies, and integrating these laws with robust state estimation to mitigate sensing noise in practical deployments.

Chapter 6 examined the potential of RL as an adaptive alternative to proportional navigation for MAV-based pest interception. The chapter identified PPO as the most suitable algorithm for single-agent scenarios, and IPPO for multi-agent setups, balancing performance with implementation simplicity. Beyond algorithm choice, the chapter emphasized the critical role of design decisions, showing that leveraging prior knowledge accelerates convergence, while lower control abstraction can unlock higher performance. Observation design, especially minimal yet structured inputs, was shown to influence generalization and sim-to-real transfer. These findings highlight RL's capacity to overcome limitations of traditional guidance and offer highly adaptable, high-performance interception strategies.

Finally, chapter 7 explored the reality gap, the discrepancy between simulated and real-world performance, as a central obstacle to deploying RL-based MAV controllers for pest interception. While traditional controllers are relatively robust, RL policies are tightly coupled to the fidelity of the training

environment. To address this, the chapter evaluated a range of sim-to-real transfer strategies, including lightweight system identification, domain randomization, privileged learning, and reward shaping. It was found that modest system identification combined with light domain randomization yields robust zero-shot transfer without incurring excessive training complexity. Additionally, strategies like environment adaptation and asymmetric actor–critic architectures further improve sample efficiency and policy generalization. Reward smoothing, specifically penalties on abrupt control changes, was shown to improve real-world stability, though often at the cost of reduced agility.

8.1. Research Objective and Questions

The literature survey shows encouraging advances in both classical PN variants and newer RL schemes for aerial interception, yet three intertwined weaknesses remain. First, no study has yet produced a rigorously controlled benchmark that pits state-of-the-art classical guidance directly against modern RL policies on the same set of agile-evader scenarios, leaving their relative merits unclear. Second, while many RL papers feature different control-abstraction depth or observation encoding, few isolate how these design choices actually govern performance. Third, the community still lacks consensus on which sim-to-real techniques, particularly lightweight domain randomization and action-smoothing penalties, most effectively close the reality gap for zero-shot deployment on MAVs. Addressing these gaps motivates an integrated research program that simultaneously benchmarks classical and learning-based controllers, disentangles key RL design factors, and tests targeted transfer strategies. The ultimate aim of this thesis is framed by the following research objective,

Research Objective

To enable reliable, pesticide-free elimination of highly maneuverable airborne pests in greenhouse environments by developing, benchmarking, and validating both classical and reinforcement learning-based guidance-and-control frameworks for MAVs.

To reach this objective, three targeted research questions are posed. Research Question 1 compares RL controllers with the current best classical guidance laws on identical, high-agility evader scenarios. Research Question 2 probes how two key RL design levers, control-abstraction depth and observation encoding, shape interception performance. Research Question 3 investigates whether domain randomization and action-smoothing penalties can deliver zero-shot sim-to-real transfer for RL policies. Together, these questions close the literature gaps in benchmarking, RL design understanding, and reality-gap mitigation, thereby laying the groundwork for a robust benchmark suite and practical design guidelines for future MAV interception systems.

Research Question 1

How do reinforcement learning-based guidance and control policies for MAV pest interception compare to state-of-the-art classical controllers in terms of performance across representative evasion scenarios?

Research Question 2

How do control abstraction level and observation design influence the performance of reinforcement learning-based MAV pest interception policies?

Research Question 3

How do domain randomization and action smoothing influence the zero-shot sim-to-real transferability of reinforcement learning-based control policies for MAV pest interception?

Part III

Conclusion

9

Conclusion

This thesis contributes to the development of the PATS-X system by studying guidance and control for aerial to aerial interception. An extensive literature review established the necessary background, distilled applicable insights, and identified gaps where a meaningful contribution could be made. Classical guidance laws for interception were evaluated to provide a strong reference. The potential of reinforcement learning was then assessed, and the principal design levers that influence performance were identified. On this basis, guidance and control methods were developed, tested, and validated across those design levers and compared against the best classical approach from the literature.

Using the findings from this investigation the research questions are now answered. The research questions are restated and subsequently answered. Throughout the limitations of the research are identified and to finalize the report, recommendations for future research are presented.

Research Question 1

How do reinforcement learning-based guidance and control policies for MAV pest interception compare to state-of-the-art classical controllers in terms of performance across representative evasion scenarios?

Across replays of recorded insect flights made with the PATS-X system, RL policies outperform the state-of-the-art controller FRPN in both speed and reliability. The best RL policy achieved a median first-interception time of $0.85\,[\text{IQR } 0.76\text{-}1.07]$ s with a 99.1% interception rate, whereas FRPN showed $1.90\,[\text{IQR } 1.04\text{-}2.80]$ s at 95.6% success. RL also maintained closer proximity during engagements, indicating more persistent control once on target. On PATS-X the reinforcement learning controller, deployed through the available acceleration interface with soft frustum constraints, attained a 95.6% interception rate versus 80.0% for the existing PATS controller and showed a trend toward faster interceptions.

Research Question 2

How do control abstraction level and observation design influence the performance of reinforcement learning-based MAV pest interception policies?

To answer the second research question, lower abstraction consistently improves RL performance: median first-interception time drops from 1.44 s (acceleration) to 1.31 s (CTBR) and to 0.85 s (motor), a $\approx 41\%$ reduction from acceleration to motor control. Observation design is decisive; expressing target relative position and velocity in the body frame $[\Delta p_b, \Delta v_b]$ yields the best performance. An interesting observation is that adding LOS-rate $\dot{\phi}$, the crucial feedback signal for all PN interception laws degraded performance significantly. Minimal inputs enabling interception are relative position plus relative velocity; supplying absolute states alone does not suffice. Action history does not help, and short observation histories act mainly as a noise filter.

Research Question 3

How do domain randomization and action smoothing influence the zero-shot sim-to-real transferability of reinforcement learning-based control policies for MAV pest interception?

Motor-level policies trained with 0% DR perform best in simulation but fail to transfer; modest DR 10% enables zero-shot transfer, while heavier DR progressively degrades performance. Action-difference penalties materially aid motor-level transfer and safety; reducing command jitter, avoiding motor overheating, and yielded the best deployed performance. However, too much smoothing destabilized flight. CTBR policies transfer across all tested configuration and show no clear benefit from DR nor reward smoothing.

The main limitations concern evaluation realism, hardware suitability, and experimental repeatability. Evaluation realism is limited by non-reactive targets. Real pests respond to pursuit; therefore, open-loop replays can overstate performance. Future work should incorporate reactive evaders through multi-agent RL, like was done in [29]. A consistent limitation faced throughout the investigation was due to hardware and implementation constraints. The Bebop 2 platform prevented reliable acceleration-level deployment and hindered motor-level fidelity; this can be addressed by moving to a rigid, agile airframe with a capable acceleration controller. Finally, to better evaluate methods for crossing the reality gap, testing on a task with higher repeatability would allow for better identification of trends. Specifically, the interplay of abstraction, reward smoothing, and domain randomization should be evaluated on an easily repeatable task with agile flight, such as drone racing.

References

[1] Yi Yang et al. “Climate change exacerbates the environmental impacts of agriculture”. In: *Science* 385.6713 (2024).

[2] E-C Oerke. “Crop losses to pests”. In: *The Journal of agricultural science* 144.1 (2006), pp. 31–43.

[3] PATS Indoor Drone Solutions. *PATS-X*. https://www.pats-drones.com/patsx. Accessed: 2025-04-05. 2025.

[4] Ridha Guebsi, Sonia Mami, and Karem Chokmani. “Drones in precision agriculture: A comprehensive review of applications, technologies, and challenges”. In: *Drones* 8.11 (2024), p. 686.

[5] Neil F Palumbo, Ross A Blauwkamp, and Justin M Lloyd. “Modern homing missile guidance theory and techniques”. In: *Johns Hopkins APL technical digest* 29.1 (2010), pp. 42–59.

[6] Ermin Wei et al. “Pursuit and an evolutionary game”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2105 (2009), pp. 1539–1559. DOI: 10.1098/rspa.2008.0459.

[7] Michal Pliska et al. “Towards Safe Mid-Air Drone Interception: Strategies for Tracking and Capture”. In: (May 2024). URL: http://arxiv.org/abs/2405.13542.

[8] Ciann-Dong Yang and Chi-Ching Yang. “A unified approach to proportional navigation”. In: *IEEE Transactions on Aerospace and Electronic Systems* 33.2 (1997), pp. 557–567. DOI: 10.1109/7.575895.

[9] Debasish Ghose. “True proportional navigation with maneuvering target”. In: *IEEE Transactions on Aerospace and Electronic Systems* 30.1 (1994), pp. 229–237. DOI: 10.1109/7.250423.

[10] U.S. Shukla and P.R. Mahapatra. “The proportional navigation dilemma-pure or true?” In: *IEEE Transactions on Aerospace and Electronic Systems* 26.2 (1990), pp. 382–392. DOI: 10.1109/7.53445.

[11] M. Mehrandezh et al. “Robotic interception of moving objects using ideal proportional navigation guidance technique”. In: *Robotics and Autonomous Systems* 28.4 (1999), pp. 295–310. DOI: 10.1016/S0921-8890(99)00044-5.

[12] M. Mehrandezh et al. “Robotic interception of moving objects using an augmented ideal proportional navigation guidance technique”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 30.3 (2000), pp. 238–250. DOI: 10.1109/3468.844354.

[13] Bing Zhu, Abdul Hanif Bin Zaini, and Lihua Xie. “Distributed Guidance for Interception by Using Multiple Rotary-Wing Unmanned Aerial Vehicles”. In: *IEEE Transactions on Industrial Electronics*. Vol. 64. Institute of Electrical and Electronics Engineers Inc., July 2017, pp. 5648–5656. DOI: 10.1109/TIE.2017.2677313.

[14] Chin-Der Yang and Cheng-Chi Yang. “Optimal pure proportional navigation for maneuvering targets”. In: *IEEE Transactions on Aerospace and Electronic Systems* 33.3 (1997), pp. 949–957. DOI: 10.1109/7.599315.

[15] Vlahov et al. “On Developing a UAV Pursuit-Evasion Policy Using RL”. In: *AIAA Guidance, Navigation, and Control* (2018).

[16] Weifan Li, Yuanheng Zhu, and Dongbin Zhao. “Missile guidance with assisted deep reinforcement learning for head-on interception of maneuvering target”. In: *Complex and Intelligent Systems* 8 (2 Apr. 2022), pp. 1205–1216. ISSN: 21986053. DOI: 10.1007/s40747-021-00577-6.

[17] He et al. “Computational Missile Guidance via Deep Reinforcement Learning”. In: *Aerospace Science and Technology* (2021).

[18] Wang et al. “Integrated Guidance-and-Control Design for 3D Interception Based on DRL”. In: *Aerospace* (2023).

[19] Qiu et al. “Recorded Recurrent TD3 for Missile Guidance”. In: *Aerospace Science and Technology* (2024).

[20] David Silver. *Lectures on Reinforcement Learning*. url: https://www.davidsilver.uk/teaching/. 2015.

[21] Ahmad Taher Azar et al. *Drone deep reinforcement learning: A review*. May 2021. doi: 10.3390/electronics10090999.

[22] Robin Ferede et al. “End-to-end reinforcement learning for time-optimal quadcopter flight”. In: *arXiv preprint arXiv:2311.16948* (2023). url: https://arxiv.org/abs/2311.16948.

[23] William Koch et al. “Reinforcement learning for UAV attitude control”. In: *ACM Transactions on Cyber-Physical Systems* 3.2 (2019), pp. 1–21.

[24] Nathan O. Lambert et al. “Low-level control of a quadrotor with deep model-based reinforcement learning”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4224–4230.

[25] Yunlong Song et al. “Reaching the limit in autonomous racing: Optimal control versus reinforcement learning”. In: *Science Robotics* 8.82 (2023), eadg1462.

[26] Fu et al. “A UAV Pursuit-Evasion Strategy Based on DDPG and Imitation Learning”. In: *International Journal of Aerospace Engineering* (2022).

[27] Yan et al. “Ballistic Missile Midcourse Intelligent Maneuver Strategy Based on PPO Algorithm”. In: *ICGNC 2022*. 2023.

[28] Brian Gaudet and Roberto Furfaro. “Integrated and Adaptive Guidance and Control for Endoatmospheric Missiles via Reinforcement Meta-Learning”. In: *AIAA Scitech 2022 Forum*. American Institute of Aeronautics and Astronautics. 2022. doi: 10.2514/6.2022-1244. url: https://arc.aiaa.org/doi/10.2514/6.2022-1244.

[29] Reinier Vos. “Hunt like a Dragonfly and Strike like a Drone: Simulating Games of Pursuit and Evasion to Optimize Quadcopter Control for Insect Pest Interception in Greenhouses”. Master’s thesis, Faculty of Aerospace Engineering. MA thesis. Delft, The Netherlands: Delft University of Technology, Oct. 2024. url: http://repository.tudelft.nl/.

[30] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. url: http://arxiv.org/abs/1707.06347.

[31] Ashley Hill et al. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[32] Eric Liang et al. “RLlib: Abstractions for Distributed Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2018. url: https://arxiv.org/pdf/1712.09381.

[33] Shengyi Huang et al. “CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms”. In: *Journal of Machine Learning Research* 23.274 (2022), pp. 1–18. url: http://jmlr.org/papers/v23/21-1342.html.

[34] Brian Gaudet, Roberto Furfaro, and Richard Linares. “Reinforcement learning for angle-only intercept guidance of maneuvering targets”. In: *Aerospace Science and Technology* 99 (Apr. 2020). issn: 12709638. doi: 10.1016/j.ast.2020.105746.

[35] Xia et al. “Hybrid SAC-FIS for Multi-Target UAV Interception”. In: *Drones* (2024).

[36] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. “A Survey and Critique of Multiagent Deep Reinforcement Learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.6 (2019), pp. 750–797. doi: 10.1007/s10458-019-09421-1. url: https://doi.org/10.1007/s10458-019-09421-1.

[37] Sven Gronauer and Klaus Diepold. “Multi-agent deep reinforcement learning: a survey”. In: *Artificial Intelligence Review* 55 (2 Feb. 2022), pp. 895–943. issn: 15737462. doi: 10.1007/s10462-021-09996-w.

[38] Jiayu Chen et al. “Multi-UAV Pursuit-Evasion with Online Planning in Unknown Environments by Deep Reinforcement Learning”. In: *arXiv preprint arXiv:2409.15866* (2024).

[39] Kaifang Wan et al. “An improved approach towards multi-agent pursuit–evasion game decision-making using deep reinforcement learning”. In: *Entropy* 23.11 (2021), p. 1433.

[40] Jianfeng Ye et al. “A Pursuit Strategy for Multi-Agent Pursuit-Evasion Game via Multi-Agent Deep Deterministic Policy Gradient Algorithm”. In: *2022 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE. 2022, pp. 418–423. doi: 10.1109/ICUS56180.2022.9979335.

[41] Ryan Lowe et al. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *CoRR* abs/1706.02275 (2017). arXiv: 1706.02275. url: http://arxiv.org/abs/1706.02275.

[42] Christian Schroeder De Witt et al. “Is independent learning all you need in the starcraft multi-agent challenge?” In: *arXiv preprint arXiv:2011.09533* (2020).

[43] Chao Yu et al. “The surprising effectiveness of ppo in cooperative multi-agent games”. In: *Advances in neural information processing systems* 35 (2022), pp. 24611–24624.

[44] Cristino De Souza, Felipe da Silva, and Thiago Oliveira-Santos. “Decentralized multi-agent pursuit using deep reinforcement learning”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4552–4559. doi: 10.1109/LRA.2021.3062194.

[45] Ruilong Zhang et al. “Game of drones: Multi-UAV pursuit-evasion game with online motion planning by deep reinforcement learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* 34.10 (2022), pp. 7900–7909.

[46] Isaac E. Weintraub, Meir Pachter, and Eloy Garcia. *An Introduction to Pursuit-evasion Differential Games*. 2020. arXiv: 2003.05013 [math.OC]. url: https://arxiv.org/abs/2003.05013.

[47] J. Shinar and S. Gutman. “Recent advances in optimal pursuit and evasion”. In: *Conference on Decision and Control including the 17th Symposium on Adaptive Processes*. Vol. 17. San Diego, CA: IEEE, 1978, pp. 960–965.

[48] J. Shinar. *Solution Techniques for Realistic Pursuit Evasion Games*. Technical Report. Haifa, Israel: Technion - Israel Institute of Technology, 1980.

[49] J. Shinar, V. Y. Glizer, and V. Turetsky. “A Pursuit-Evasion Game with Hybrid Pursuer Dynamics”. In: *Proceedings of the European Control Conference*. Kos, Greece, July 2007, pp. 1306–1313.

[50] N. Greenwood. “A Differential Game in Three Dimensions: The Aerial Dogfight Scenario”. In: *Dynamics and Control* 2.2 (1992), pp. 161–200.

[51] Jonas Eschmann, Dario Albani, and Giuseppe Loianno. “Learning to fly in seconds”. In: *IEEE Robotics and Automation Letters* 9.7 (2024), pp. 6336–6343.

[52] Jiayi Xu et al. “Learning to Fly: Computational Controller Design for Hybrid UAVs with Reinforcement Learning”. In: *ACM Transactions on Graphics* 38.4 (2019), pp. 1–12. doi: 10.1145/3306346.3322940.

[53] Julio Panerati et al. “Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-Agent Quadcopter Control”. In: *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2021, pp. 8584–8591.

[54] Takayuki Osa et al. “An Algorithmic Perspective on Imitation Learning”. In: *Foundations and Trends® in Robotics* 7.1-2 (2018), pp. 1–179.

[55] Yoshua Bengio et al. “Curriculum Learning”. In: *Proceedings of the 26th International Conference on Machine Learning*. 2009, pp. 41–48.

[56] Elia Kaufmann, Leonard Bauersfeld, and Davide Scaramuzza. “A benchmark comparison of learned control policies for agile quadrotor flight”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 10504–10510.

[57] Robin Ferede et al. “One Net to Rule Them All: Domain Randomization in Quadcopter Racing Across Different Platforms”. In: *arXiv preprint arXiv:2504.21586* (2025).

[58] Alberto Dionigi, Gabriele Costante, and Giuseppe Loianno. “The Power of Input: Benchmarking Zero-Shot Sim-to-Real Transfer of Reinforcement Learning Control Policies for Quadrotor Control”. In: *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 2024, pp. 11812–11818. isbn: 9798350377705. doi: 10.1109/IROS58592.2024.10802831.

[59] Jiayu Chen et al. “What Matters in Learning A Zero-Shot Sim-to-Real RL Policy for Quadrotor Control? A Comprehensive Study”. In: *arXiv preprint arXiv:2412.11764* (2024).

[60] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[61] Sven Gronauer, Daniel Stümke, and Klaus Diepold. “Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 7508–7514.

[62] Erica Salvato et al. “Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning”. In: *IEEE Access* 9 (2021), pp. 153171–153187. issn: 21693536. doi: 10.1109/ACCESS.2021.3126658.

[63] Lennart Ljung. “System identification”. In: *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[64] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey”. In: (Sept. 2020). doi: 10.1109/SSCI47803.2020.9308468.. url: http://arxiv.org/abs/2009.13303%20http://dx.doi.org/10.1109/SSCI47803.2020.9308468..

[65] Sihao Sun, Coen C de Visser, and Qiping Chu. “Quadrotor gray-box model identification from high-speed flight data”. In: *Journal of Aircraft* 56.2 (2019), pp. 645–661.

[66] Josh Tobin et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada: IEEE, 2017, pp. 23–30. doi: 10.1109/IROS.2017.8202133. url: https://doi.org/10.1109/IROS.2017.8202133.

[67] Eugene Valassakis, Zihan Ding, and Edward Johns. “Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5372–5379.

[68] Artem Molchanov et al. “Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 59–66.

[69] Lerrel Pinto et al. “Asymmetric Actor Critic for Image-Based Robot Learning”. In: (Oct. 2017). url: http://arxiv.org/abs/1710.06542.

[70] Lerrel Pinto et al. “Robust Adversarial Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2817–2826. url: https://proceedings.mlr.press/v70/pinto17a.html.

A

System Identification

The procedure used for system identification for the Bebop 2 is taken from Ferede et al. [57], which draws upon the work of Sun et al. [65]. As input the logs of a flight under CTBR guidance is used. The filtered signals are used to perform a regression to fit the coefficient of each of the respective dynamic models.

A.1. Motor

For identification we use a reduced end-to-end quadrotor model taken from [57] that exposes only the parametric maps between commanded motor inputs and rotor speed, rotor speed and specific force, and rotor speed and body moments.

Motor first–order dynamics and thrust curve. Each rotor speed ω_i follows a first–order response to a steady-state command $\bar{\omega}_{ci}$,

$$\dot{\omega}_i = \frac{\bar{\omega}_{ci} - \omega_i}{\tau_\omega}, \qquad i \in \{1, \dots, 4\}, \tag{A.1}$$

with time constant $\tau_\omega > 0$. The steady-state speed is a shaped function of the normalized input $u_i \in [0, 1]$:

$$\bar{\omega}_{ci} = (\omega_{\max} - \omega_{\min}) \sqrt{k_t u_i^2 + (1 - k_t) u_i} \; + \; \omega_{\min}, \tag{A.2}$$

where $\omega_{\min}, \omega_{\max}$ and $k_t \in [0, 1]$ are identified. See Figure A.1 for parameters fit for the Bebop 2 and the values used throughout experimentation.

Specific force Let (v_x^B, v_y^B) be the body-frame lateral components of the vehicle velocity. The specific force $F = [F_x, F_y, F_z]^T$ is parameterized as

$$F = \begin{bmatrix} -\sum_{i=1}^{4} k_x \, v_x^B \, \omega_i \\ -\sum_{i=1}^{4} k_y \, v_y^B \, \omega_i \\ -\sum_{i=1}^{4} k_\omega \, \omega_i^2 \end{bmatrix}, \tag{A.3}$$

with planar drag gains k_x, k_y and thrust coefficient k_ω to be fitted. See in Figure A.2 the parameters identified and simulated fit. Note, the fit for the lateral specific force (F_x, F_y) is noticeably weaker than for the thrust term F_z. This is expected, during our flights the vehicle spends most of its time accelerating and decelerating along the thrust axis (z_B), while lateral body speeds $|v_x^B|, |v_y^B|$ remain small.

Body moments. The angular acceleration surrogate $M = [M_x, M_y, M_z]^T$ is modeled as

$$M_x = -k_{p1}\omega_1^2 - k_{p2}\omega_2^2 + k_{p3}\omega_3^2 + k_{p4}\omega_4^2, \tag{A.4a}$$
$$M_y = -k_{q1}\omega_1^2 + k_{q2}\omega_2^2 - k_{q3}\omega_3^2 + k_{q4}\omega_4^2, \tag{A.4b}$$
$$M_z = -k_{r1}\omega_1 + k_{r2}\omega_2 + k_{r3}\omega_3 - k_{r4}\omega_4 - k_{r5}\dot{\omega}_1 + k_{r6}\dot{\omega}_2 + k_{r7}\dot{\omega}_3 - k_{r8}\dot{\omega}_4, \tag{A.4c}$$

where the $k_\cdot$ coefficients (including the terms on $\dot{\omega}_i$ that capture propeller inertial effects) are identified. See in Figure A.3 the regression performed.

A.2. CTBR/Acceleration

The CTBR abstraction level exposes first–order tracking between commanded and measured body rates and thrust, and a linear planar drag map in the body frame. Note, the parameters identified here are also used in the acceleration abstraction level

Actuator first–order dynamics (rates & thrust). We model the body rates and specific thrust as first–order trackers of their commands:

$$\dot{y} \;=\; \frac{u - y}{\tau_y}, \qquad y \in \{p, q, r, T\}, \quad u \in \{p_{\mathrm{cmd}}, q_{\mathrm{cmd}}, r_{\mathrm{cmd}}, T_{\mathrm{cmd}}\}, \tag{A.5}$$

with per–channel time constants $\tau_p, \tau_q, \tau_r, \tau_T > 0$ identified from flight logs. Combined fits and simulated overlays are shown in Figure A.4.

Planar drag. We model the in–plane specific force in body coordinates as

$$\begin{bmatrix} a_x^B \\ a_y^B \end{bmatrix} \;=\; \begin{bmatrix} -d_x & 0 \\ 0 & -d_y \end{bmatrix} \begin{bmatrix} v_x^B \\ v_y^B \end{bmatrix}, \tag{A.6}$$

and estimate d_x, d_y by linear regression on filtered signals, discarding samples with $|v_\cdot^B| < v_{\min}$. Identified values and fit quality are summarized in Figure A.5.

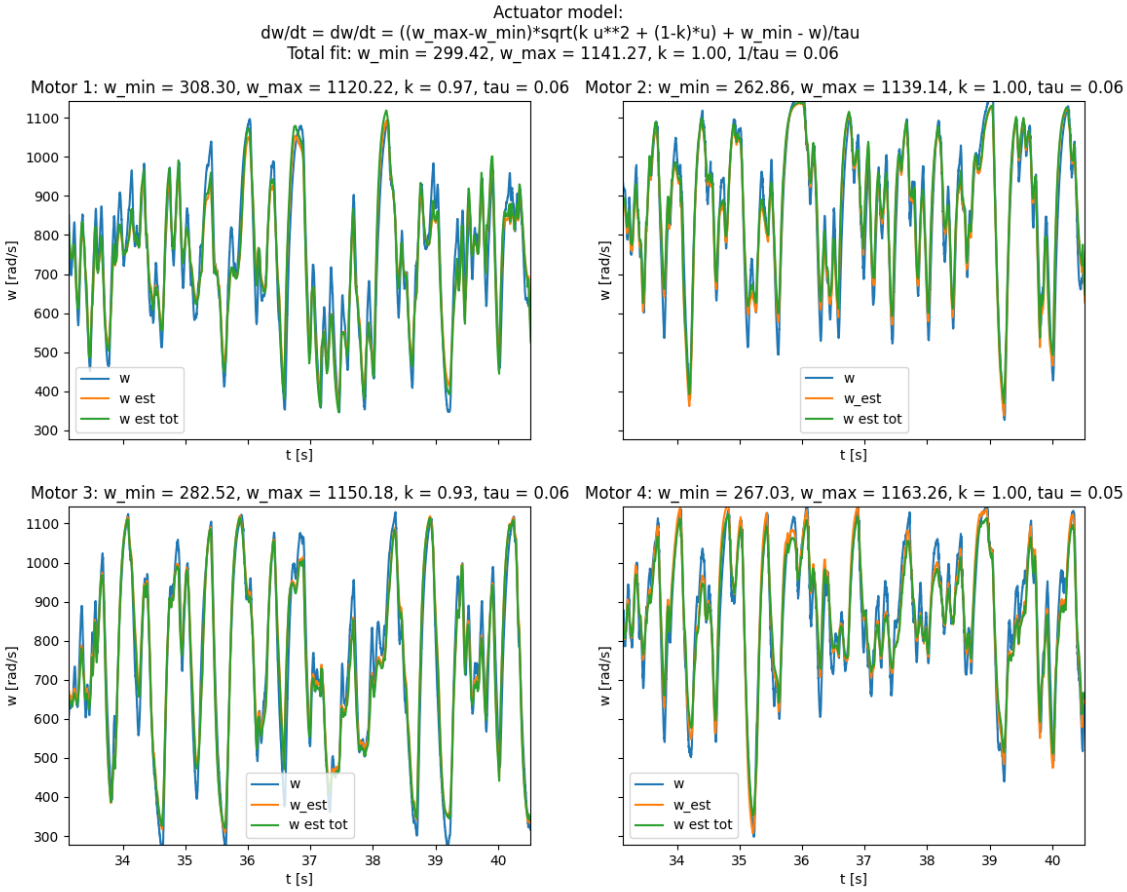


Figure A.1: Motor-speed model identification. A nonlinear static map from input u to steady-state $\bar{\omega}$ combined with a first-order lag $\dot{\omega} = (\bar{\omega} - \omega)/\tau_\omega$. Estimated $\omega_{\min}, \omega_{\max}, k_t, \tau_\omega$ yield simulated speeds that closely track the measurements, as seen through the simulated responses.

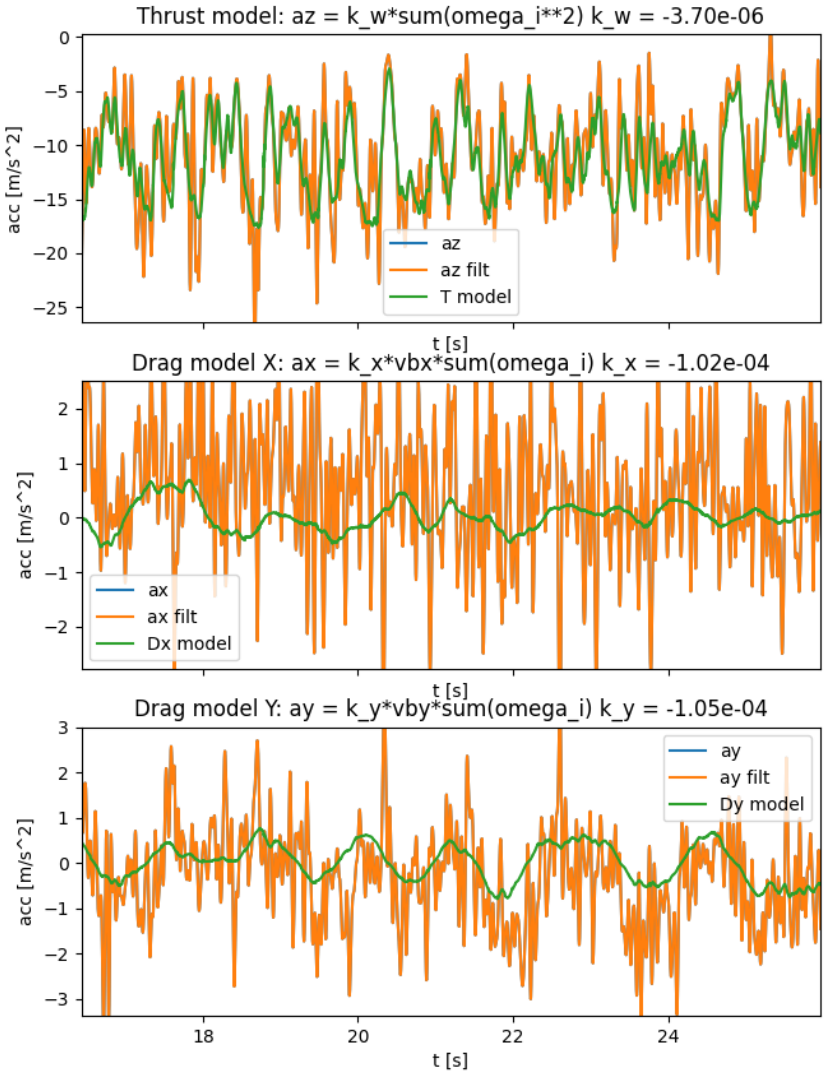


Figure A.2: Specific-force model fit, thrust and planar drag). Top: vertical thrust $a_z \approx k_\omega \sum_i \omega_i^2$. Middle/bottom: in-plane drag $a_x \approx k_x\, v_{bx} \sum_i \omega_i$ and $a_y \approx k_y\, v_{by} \sum_i \omega_i$. Green curves are model predictions; orange curves are filtered measurements.

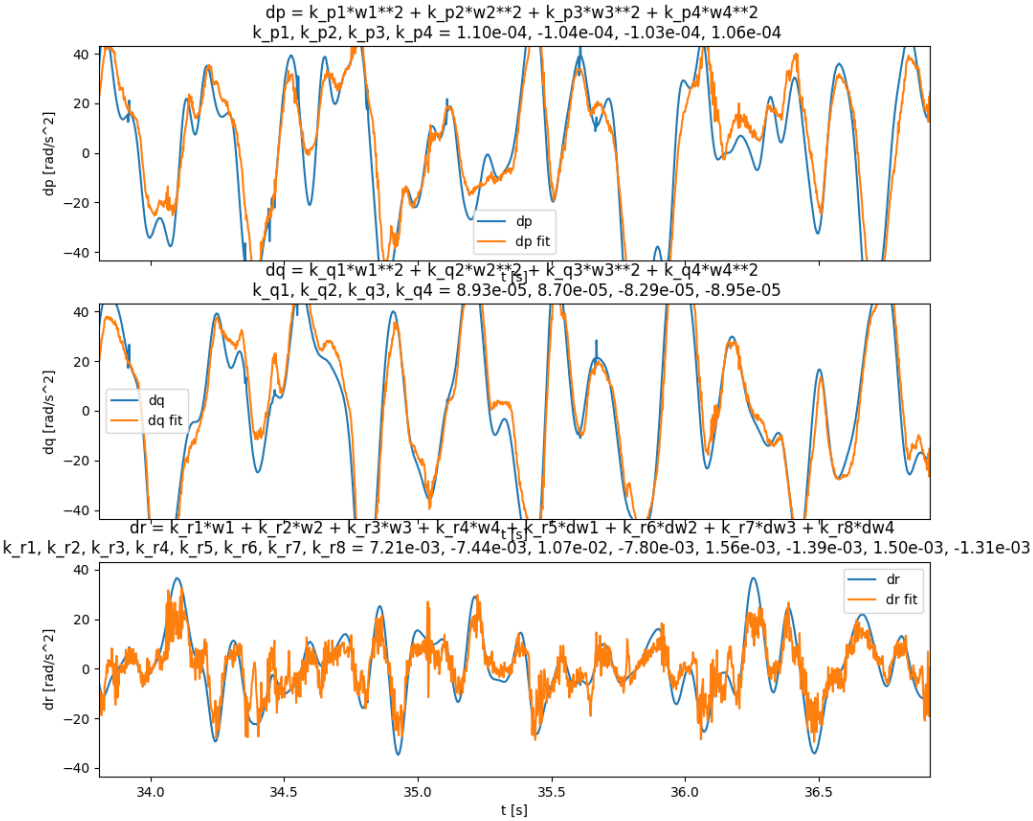


Figure A.3: Angular-moment model fit. Measured derivatives $(\dot{p}, \dot{q}, \dot{r})$ overlaid with a model linear in rotor terms (ω_i^2 and $\dot{\omega}_i$). The identified gains (k_{p*}, k_{q*}, k_{r*}) reproduce the main features of the rate dynamics as seen through the simulated response.

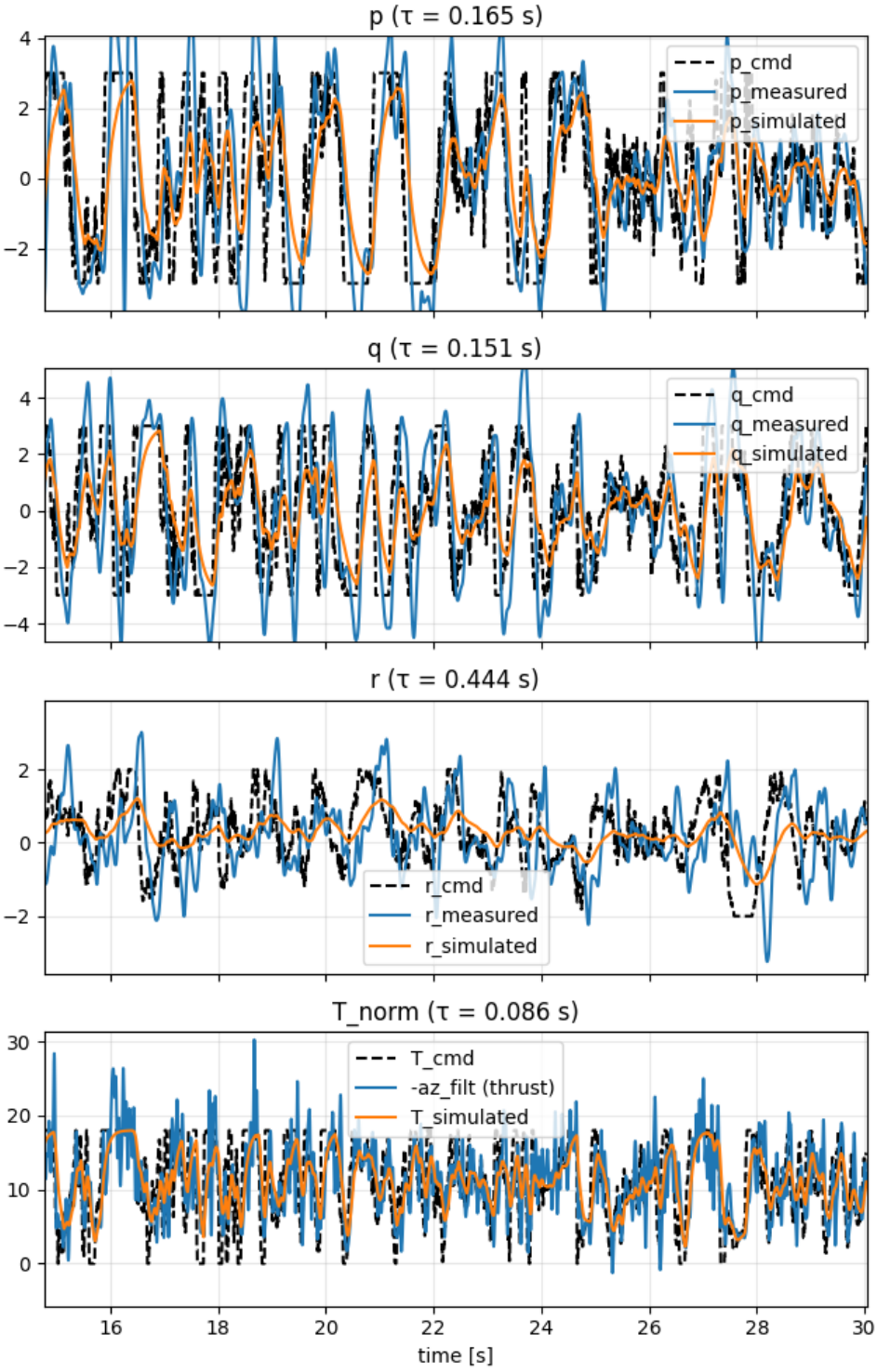


Figure A.4: First–order fits for p, q, r, T: measured (blue) against simulated with identified $\tau_p, \tau_q, \tau_r, \tau_T$ (orange).

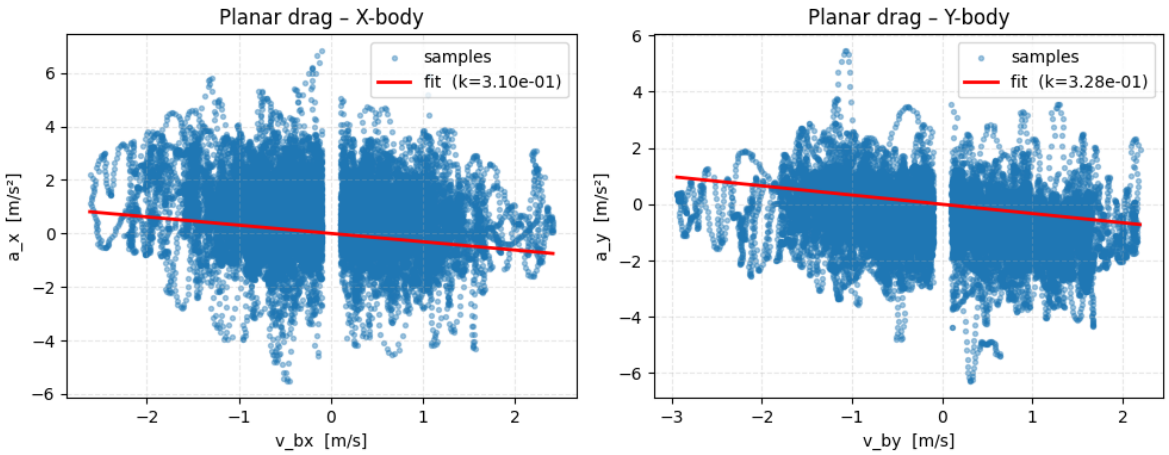


Figure A.5: Planar drag identification in the body frame. Velocities filtered to only be included when above a threshold of 0.01m/s. Red shows fitted response against all samples shown in blue.