Beyond Proportional Navigation

Deep Reinforcement Learning for Robust Drone Interception

Maxime Capelle

TU Delft

Beyond Proportional Navigation

Deep Reinforcement Learning for Robust Drone Interception

MSc Thesis report

by

Maxime Capelle

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on November 5, 2025 at 15:00

Thesis committee:	
Chair:	Dr. L. Ferranti
Supervisors:	Dr. L. Ferranti
	Dr. E. Smeur
External examiners:	Prof.dr. J. Alonso-Mora
	Dr. R. Ferrari
Place:	Faculty of Mechanical Engineering, Delft
Project Duration:	Feb, 2025 - November, 2025
Student number:	5261457

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Mechanical Engineering · Delft University of Technology

Preface

The completion of this MSc thesis marks the end of my transformative student journey at Delft University of Technology. Starting at the Faculty of Aerospace Engineering, I could not have asked for a better introduction to the world of science and technology. Being surrounded by bright minds and challenging material ignited a lasting passion for problem-solving that will undoubtedly drive my future career. This naturally led me to the Robotics track at the Faculty of Mechanical Engineering, an environment that embodies innovation and intelligent systems.

A defining moment in this journey was joining the TU Delft MAVLab, a place that truly sparked my fascination with drones and autonomous flight. Their passionate community and cutting-edge projects made me realise the immense potential of aerial robotics in solving complex, real-world problems. This ultimately led to my thesis research, which focuses on developing intelligent drones capable of protecting airspaces safely and autonomously. With this topic, I sought to challenge myself by engaging with the most advanced and rapidly evolving approaches in UAV research, particularly those driven by learning-based controllers. With my thesis, I hope to share my excitement for this new generation of controllers and inspire others to push the boundaries of what autonomous aerial systems can achieve.

None of this work would have been possible without the guidance, support, and encouragement of the people around me throughout this journey. I would first like to express my sincere gratitude to my supervisors, Dr. Laura Ferranti and Dr. Ewoud Smeur, for their constant support, insightful discussions, and enthusiasm for my project. I am truly thankful for the freedom they gave me to explore my ideas while always providing the direction needed to stay on course. I would also like to thank the TU Delft MAVLab community for both their direct and indirect contributions through their experience, support, and the opportunities they provided. Their contagious passion for aerial robotics makes the lab an inspiring place that will continue to motivate future students. Finally, I would like to thank my family and friends for their unwavering support, encouragement, and belief in me. They have been my role models and a constant source of strength, making this journey as rewarding and memorable as it was.

As I look toward the future, I close this chapter with the words of Steve Jobs, which truly resonate with me:

“Stay Hungry. Stay Foolish.”

– Maxime Capelle
Delft, November 2025

Abstract

The rapid increase in drone threats has created an urgent need for practical counter-drone systems for safety and defence purposes. Interceptor drones represent a promising and cost-effective solution for removing unwanted aerial vehicles. However, classical guidance laws such as Proportional Navigation (PN) are not well adapted to highly agile targets under realistic perception constraints. In parallel, deep reinforcement learning (DRL) has demonstrated exceptional performance in high-speed quadcopter control tasks, motivating its consideration for interception applications. A major challenge for such learning-based controllers lies in sim-to-real transfer. This work addresses this challenge through extensive domain randomization and explicit modelling of realistic perception constraints, including sensor noise, limited field of view, sensing ranges, and sensor update rates. A framework is proposed that trains Single Rotor Thrust (SRT) DRL policies for quadcopters and evaluates them against a classical PN implementation within a physics-based simulator. The resulting controllers generalize across platform variations and outperform classical guidance under modelled perception limitations, demonstrating improved robustness and transferability.

Beyond Proportional Navigation: Deep Reinforcement Learning for Robust Drone Interception

Maxime Capelle

***Abstract*—The rapid increase in drone threats has created an urgent need for practical counter-drone systems for safety and defence purposes. Interceptor drones represent a promising and cost-effective solution for removing unwanted aerial vehicles. However, classical guidance laws such as Proportional Navigation (PN) are not well adapted to highly agile targets under realistic perception constraints. In parallel, deep reinforcement learning (DRL) has demonstrated exceptional performance in high-speed quadcopter control tasks, motivating its consideration for interception applications. A major challenge for such learning-based controllers lies in sim-to-real transfer. This work addresses this challenge through extensive domain randomization and explicit modelling of realistic perception constraints, including sensor noise, limited field of view, sensing ranges, and sensor update rates. A framework is proposed that trains Single Rotor Thrust (SRT) DRL policies for quadcopters and evaluates them against a classical PN implementation within a physics-based simulator. The resulting controllers generalise across platform variations and outperform classical guidance under modelled perception limitations, demonstrating improved robustness and transferability.**

I. Introduction

In recent years, the use of drones has risen in both civilian and military domains[1]. Although these systems have enabled numerous beneficial applications, their widespread usage has also introduced new challenges related to defence and civilian safety. Whether used as weaponised loitering munitions[2] or operated carelessly near restricted areas such as airports[3], drones can pose significant risks. Consequently, there is a growing need for counter-drone technologies that can remove aerial vehicles from the airspace efficiently and cost-effectively.

Interceptor drones have emerged as a promising solution for neutralising such airborne threats autonomously[4]. However, as targets become increasingly small and agile, traditional interception guidance techniques such as Proportional Navigation (PN) struggle to adapt. Several extensions have been proposed to improve PN performance, but its success against highly manoeuvrable targets remains limited[5]. Recent developments in quadcopter guidance and control have highlighted the potential of learning-based controllers[6], [7]. These approaches enable sophisticated decision-making and precise manoeuvring in highly dynamic flight and under uncertain conditions[8]. Deep reinforcement learning (DRL) has emerged as a promising solution for training such controllers, as demonstrated by recent successes in autonomous drone racing[9]–[11]. The methods used there can be extended to address the challenges of interception using quadcopters, but concerns about robustness and transferability from simulation to reality remain.

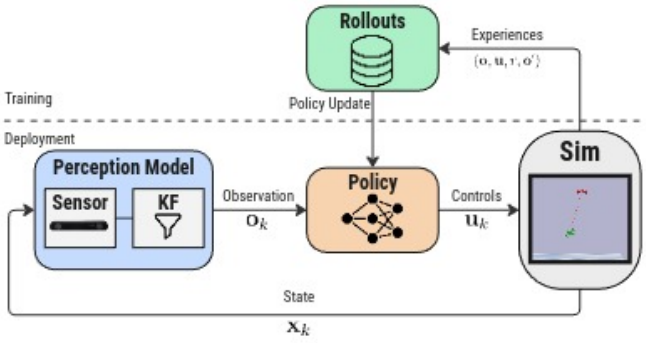

Fig. 1. Overview of the training and deployment pipeline. During training (top), experiences collected in simulation are used to update the DRL policy. During deployment (bottom), the policy receives observations from the perception model and outputs learned control actions.

In this work, a framework for training robust Single Rotor Thrust (SRT) DRL controllers is presented and applied to a modelled quadcopter interception scenario. The SRT architecture replaces the conventional guidance and control modules with neural networks that directly map observations to individual motor thrusts. This low-level control allows the full exploitation of the platform's actuation capabilities. The scenario focuses on the final phase of interception, where guidance and control are the most challenging[12]. The proposed approach incorporates extensive domain randomization[13] and explicitly models key real-world challenges, including sensor noise, limited update rates, finite sensing ranges, and restricted fields of view. By combining physically grounded modelling with large-scale domain randomization, the framework aims to achieve generalisable policies that remain effective across varying drones and perception conditions. Figure 1 illustrates the training and deployment loops with the integrated perception model. The resulting DRL controllers are evaluated against a classical PN baseline to assess their robustness, adaptability, and learned interception strategies under increasingly realistic conditions. The trained controllers show a high degree of generalisation across different platforms. Their superior performance under modelled perception constraints highlights their robustness and potential for real-world transfer.

II. Related Work

With recent advances in quadcopter guidance and control, attention has increasingly shifted towards learning-based planning and control methods [6]. This trend is particularly prominent in drone racing applications, where recent DRL controllers have not only surpassed classical model-based techniques, such as Model Predictive Control (MPC), but have also outperformed world champion human pilots for the first time [9], [10]. Classical model-based approaches such as optimal control and non-linear MPC rely on accurate models and are computationally demanding, limiting their application in agile, real-time flight scenarios [14]. Replacing one or more components of the perception, guidance, and control pipeline with neural networks has shown clear advantages in computational efficiency, handling of high-dimensional inputs, and robustness in dynamic and uncertain environments [8].

With these advances, recent works have explored different architectures for DRL-based controllers. The most prominent are those presented in [7]: Linear Commands (LC), Collective Thrust and Body Rates (CTBR), and Single Rotor Thrusts (SRT). These approaches differ in the level of control authority delegated to the neural network. The LC architecture replaces only the guidance module, with the network outputting motion vectors such as desired velocity or acceleration. While conceptually simple, this approach is less common as it is limited to slow, higher-level planning and still relies on a full control stack to execute the desired motion. The CTBR architecture replaces both the guidance module and part of the controller, operating directly within the inner attitude control loop. This enables more aggressive manoeuvres and has been successfully demonstrated in drone racing, where learned policies outperformed human world champions [9], [10]. The SRT architecture fully replaces both the guidance and control modules as it outputs individual thrust commands for the motors. This design maximises performance potential by utilising the platform's full actuation authority and eliminating intermediate control bottlenecks. Moreover, by reducing inference latency and increasing module cohesion, it enables the development of a new generation of effective and computationally efficient controllers. Although earlier work highlighted challenges in sim-to-real transfer, utilisation of Domain Randomization (DR) [13] in drone racing and low-level control has demonstrated successful direct transfer from simulation to reality [15]–[17], making SRT a promising direction for future research.

More related to interception applications, recent literature has begun to explore using DRL-based controllers for interception and penetration strategies with missiles [18], [19], quadcopters [20], [21] and fixed-wing UAVs [22], [23]. While these studies demonstrate the potential of DRL for interception tasks, their primary focus is not on simulation-to-real transferability. They are generally built on simplified dynamics or assume full observability, with limited consideration of sensing, actuation, and modelling uncertainties. This motivates the present work, which aims to bridge this gap by leveraging an established physically grounded quadcopter model, integrating a realistic perception pipeline, and extending robustness training strategies from drone racing to the interception domain. The main contributions of this work are as follows:

1) Development of a modular training framework that produces SRT DRL policies for high-speed interception guidance on quadcopters, emphasising robustness and sim-to-real transferability.
2) Application of the proposed framework to a physically and perceptually grounded interception scenario, demonstrating the effectiveness and robustness of the trained DRL controller under uncertain conditions.
3) Implementation of a baseline quadcopter Proportional Navigation controller for comparative evaluation.

The final controllers are evaluated in simulation and assessed in terms of interception rate, interception time, and transferability when integrated with a more realistic perception pipeline. This research ultimately aims to advance the practical deployment of DRL-based controllers for autonomous interceptor drones. The code for the implementation is openly available on GitHub[1].

III. Preliminaries

This section provides a brief overview of the quadcopter model used to simulate the drones (Subsec. III-A) and outlines the problem formulation for the interception task (Subsec. III-B).

A. Quadcopter Model

To develop and test the guidance algorithms in simulation, a parametric model identical to that presented in [15] was used. This model was validated on both 3-inch and 5-inch racing drones. It enables direct transfer from simulation to reality for high-speed quadcopter guidance. For completeness, the drone model and dynamics are described below. The quadcopter state vector x and control inputs u can be defined as:

$$x = [\mathbf{p}, \mathbf{v}, \boldsymbol{\eta}, \boldsymbol{\Omega}, \boldsymbol{\omega}]^T, \qquad u = [u_1, u_2, u_3, u_4]^T \tag{1}$$

where $\mathbf{p}, \mathbf{v} \in \mathbb{R}^3$ are the position and velocity in the inertial frame, $\boldsymbol{\eta} \in \mathbb{R}^3$ is the intrinsic *ZYX* Euler angle vector, $\boldsymbol{\Omega} \in \mathbb{R}^3$ is the body angular rate vector, and $\boldsymbol{\omega} \in \mathbb{R}^4$ is the propeller angular velocity vector. The normalised control inputs are given by $u_i \in [0, 1]$. The equations of motion are defined as follows:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} & \dot{\mathbf{v}} &= g\mathbf{e}_3 + R(\eta)\mathbf{F} \\ \dot{\eta} &= Q(\eta)\boldsymbol{\Omega} & \dot{\Omega} &= \mathbf{M} \\ \dot{\omega}_i &= \tfrac{(\omega_{ci} - \omega_i)}{\tau} \end{aligned} \tag{2}$$

Here, R denotes the rotation matrix mapping body-frame vectors to the inertial frame, and Q represents the transformation from body angular rates to Euler angle derivatives. The collective thrust F, the moments M, and the steady-state motor response ω_{ci} are defined as follows:

[1] https://github.com/maximecapelle/quadintercept_drl

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

$$\mathbf{M} = \begin{bmatrix} -k_{p1}\omega_1^2 - k_{p2}\omega_2^2 + k_{p3}\omega_3^2 + k_{p4}\omega_4^2 \\ -k_{q1}\omega_1^2 + k_{q2}\omega_2^2 - k_{q3}\omega_3^2 + k_{q4}\omega_4^2 \\ -k_{r1}\omega_1 + k_{r2}\omega_2 + k_{r3}\omega_3 - k_{r4}\omega_4 \cdots \\ -k_{r5}\dot{\omega}_1 + k_{r6}\dot{\omega}_2 + k_{r7}\dot{\omega}_3 - k_{r8}\dot{\omega}_4 \end{bmatrix} \tag{4}$$

$$\omega_{ci} = (\omega_{\max} - \omega_{\min})\sqrt{k_l u_i^2 + (1 - k_l)u_i} + \omega_{\min} \tag{5}$$

The authors of [15] obtained the set of k-coefficients, along with τ, $\omega_{\min}$, and $\omega_{\max}$, by performing system identification on manual flight test data for the 3-inch and 5-inch racing drones. Additionally, to enable improved transferability between different platforms, the authors of [15] normalised the parameters such that they are in more consistent ranges (see Tab. VIII in App. B).

B. Problem Formulation

Consider an empty 3D environment $\mathcal{W} \subset \mathbb{R}^3$ containing an uncooperative target drone T and an interceptor drone I. The interceptor's objective is to reach a position within the interception sphere of radius r_{int} centred on the target, that is:

$$\|\mathbf{p}_I - \mathbf{p}_T\|_2 \leq r_{\text{int}}, \tag{6}$$

To account for discrete-time dynamics, the interception condition is expressed in terms of the zero-effort miss (ZEM) distance, representing the predicted closest approach between the interceptor and target assuming zero future control input (see App. A). A successful interception occurs if

$$ZEM < r_{\text{int}} \quad \text{and} \quad t_{\text{go}} < \Delta t_{\text{ctrl}}. \tag{7}$$

where t_{go} denotes the time to closest approach and Δt_{ctrl} the controller timestep. This formulation provides a consistent criterion for identifying interception events in discrete control settings.

An interception mission typically consists of three sequential phases: boost, mid-course, and homing, as described in [12]. The boost phase accelerates the interceptor to operating speed, the mid-course phase directs it toward the target's estimated location, and the homing phase employs onboard sensors for the final approach. Since the homing stage is the most critical and demanding in terms of guidance and control, this work focuses specifically on this final segment. The interceptor is assumed to have already completed the mid-course phase using PN guidance with external target tracking and now operates solely based on onboard sensing for the terminal approach. In this work, a stereo camera serves as the primary sensing modality, providing the target's relative direction and depth for state estimation.

Given these sensing conditions, the interception task inherently follows a Partially Observable Markov Decision Process (POMDP)[24], in which the interceptor does not have direct access to the full underlying system state. Instead, it selects actions based on observations obtained from onboard sensors, capturing the partial observability and uncertainty present in real-world interception scenarios.

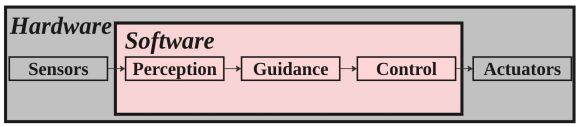

Fig. 2. Traditional autonomous system pipeline.

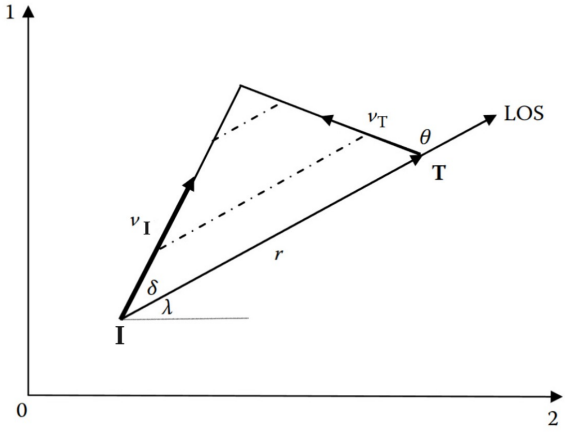

Fig. 3. Geometry of planar engagement[12].

IV. Baseline Guidance & Control

Traditionally, autonomous systems follow a pipeline, as illustrated in Fig. 2. Sensors provide raw data to the software stack, which is typically divided into three modules: Perception, Guidance, and Control. The Perception module interprets sensor data and extracts information for Guidance. Guidance generates a high-level plan for the system's desired actions, while Control translates this plan into actuator commands.

To establish a baseline for guidance and control, a classical approach was implemented in this work. For interception tasks, a widely used strategy in the literature is Proportional Navigation (PN), a simple yet effective guidance law for homing in on a moving target. PN is primarily applied in missile systems, whose dynamics, actuation capabilities, and high-accuracy sensors complement the guidance law. However, the law can also be applied to quadcopters for intercepting moving targets. In this implementation, the PN guidance module first computes the desired linear accelerations for the interceptor. These acceleration commands are then passed to a controller, which converts them into actuator inputs suitable for the quadcopter platform.

In the following, Subsec. IV-A describes Proportional Navigation and how it is implemented in the guidance module, followed by Subsec. IV-B which provides an overview of the implemented cascaded PID controller.

A. Proportional Navigation Guidance

Proportional Navigation (PN) is a widely used guidance principle in modern missile systems[12]. A common PN engagement is illustrated in Fig. 3 to define the relevant geometry. The LOS vector is denoted by r, with LOS angle λ. The interceptor and target velocities are v_I and v_T, respectively.

The angles between the LOS and the corresponding velocity vectors are δ (interceptor) and θ (target). To achieve interception, PN generates acceleration commands that maintain a zero angular rate of the LOS vector. For PN convergence, the initial velocities must satisfy the following closing condition:

$$\dot{r} < 0 \quad \text{with} \quad \dot{r} = v_T \cos\theta - v_I \cos\delta \tag{8}$$

Equation 8 ensures that the interceptor is initially approaching the target. Furthermore, the initial interceptor velocity should also lie within the collision triangle defined by the target velocity and LOS (see Fig. 3). Within this region, the PN guidance law produces lateral accelerations that steer the interceptor toward collision with the target. The general PN guidance law produces linear accelerations a_{PN} as follows:

$$a_{PN} = N \cdot v_c \cdot \dot{\lambda}(t) \tag{9}$$

In Eq. 9[12], N is the navigation constant that scales the proportionality of the LOS rate to the corrective acceleration and v_c is the closing velocity that can be defined as the velocity along r (i.e., $v_c = -||\dot{r}||$). While PN guidance generates the lateral acceleration commands required to achieve collision, it does not impose constraints on the time to intercept. As a result, PN may produce trajectories with insufficient closing velocity or with unnecessarily long interception times. To address this limitation, a complementary closing acceleration *P*-controller was implemented. It utilises the remaining acceleration budget, after the PN command, to generate accelerations along the LOS vector. This maintains a minimum desired closing velocity, thereby encouraging faster and more effective interceptions.

B. Cascaded PID Control

To convert the accelerations provided by the guidance module into motor commands, a cascaded PID controller is employed. The overall structure of this controller is illustrated in Fig. 4. The controller is setup as a cascade, with an inner and outer loop. The outer loop runs at 100 Hz and maps the commanded accelerations $\boldsymbol{a}_{PN}$ into a desired thrust T_d and attitude $\boldsymbol{\eta}_d$. The inner loop, running at 500 Hz, first computes the desired body rates $\boldsymbol{\Omega}_d$ using a proportional controller. A PID controller then determines the desired angular accelerations $\boldsymbol{\alpha}_d$ from the rates. Finally, T_d and $\boldsymbol{\alpha}_d$ are combined in the motor mixing algorithm to generate the desired propeller angular velocities $\boldsymbol{\omega}_d$. For this controller, a total of 12 gains were manually tuned: three proportional gains for the rate controller, and three sets of PID gains for each rotational axis. The presented cascaded structure is standard in multirotor control[25]–[27].

V. Methodology

This section outlines the overall architecture of the autonomous interceptor pipeline depicted in Fig. 1, and describes the training strategies adopted to ensure controller robustness. The goal of this work is to provide a comprehensive and reproducible framework for applying DRL to an autonomous interception scenario. Subsection V-A introduces the simulated onboard perception module and the modelling assumptions associated with it. Subsection V-B defines the reinforcement learning formulation, detailing the action space, observation space, and reward function. Subsection V-C presents the training methods used to enhance robustness to modelling uncertainties and to improve simulation-to-reality transferability. Finally, Subsec. V-D describes the selected DRL algorithm and the key hyperparameter choices.

A. Perception

Although perception is not the main focus of this work, establishing a realistic interception scenario requires defining the expected format and quality of the state information. This definition ensures that the perception process can be accurately represented in simulation. In this work, the interceptor is equipped with an upward-facing depth camera aligned with the vehicle's vertical axis. This follows standard setups for modern interceptor drones that are expected to perform high pitch and roll manoeuvres. Furthermore, detecting the target in the camera images typically involves computer vision or machine learning techniques. This aspect is beyond the scope of this work, and it is therefore assumed that the target can always be detected within the image at the same frame rate as the depth information, provided it remains within the field of view (FOV). The following subsections describe the depth-camera modelling assumptions and the filtering methods used to obtain stable target state estimates.

Depth Camera Model

To establish a representative performance profile of the depth camera, this work relies on the technical specifications of the commercially available *StereoLabs ZED2i*[28]. It also incorporates the empirical depth error models proposed in [29]. The *ZED2i* is offered with two lens configurations, a 2.1mm wide-angle lens optimised for short-range operation, and a 4.0mm lens optimised for long-range operation at the cost of a reduced FOV. The key technical specifications for the different focal lengths are summarised in Tab. I.

TABLE I
TECHNICAL SPECIFICATIONS OF THE STEREOLABS ZED2I CAMERA. LENS SPECS AND RESOLUTION/FPS ARE PRESENTED SEPARATELY.

Lens	Min Depth [m]	Max Depth [m]	FOV [H × V × D]°
2.1 mm	0.3	20	110 × 70 × 120
4.0 mm	1.5	35	72 × 44 × 81

Resolution [px]	Max FPS
2208 × 1242	15
1920 × 1080	30
1280 × 720	60
672 × 376	100

The tables highlight how lens choice affects sensing range and FOV, and how camera resolution influences the maximum frames per second (FPS). Here, H, V, and D denote the horizontal, vertical, and diagonal opening angles of the FOV, respectively. To simplify the camera model and reduce computation, five modelling assumptions have been made,

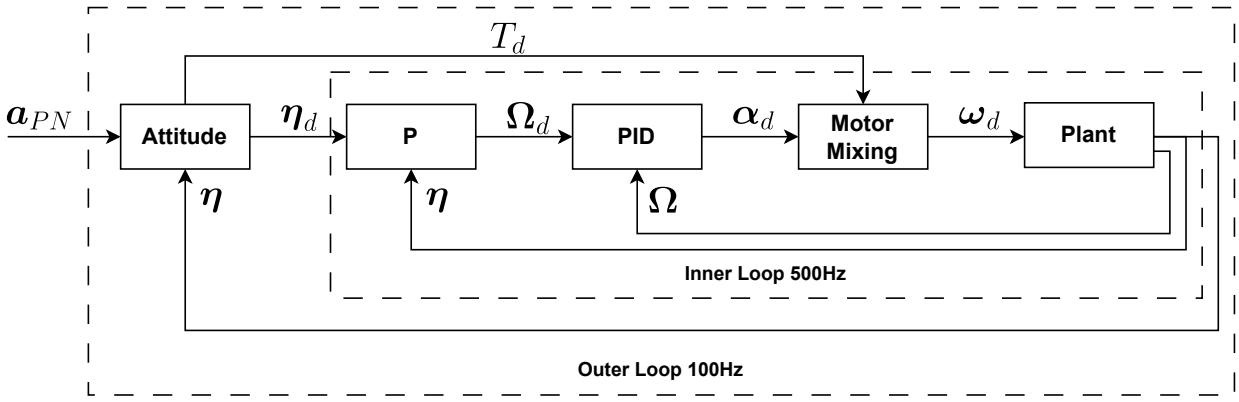

Fig. 4. Cascaded PID controller converting commanded linear accelerations $\boldsymbol{a}_{PN}$ into desired propeller angular velocities $\boldsymbol{\omega}_d$.

denoted by the identifier **D**.

Assumption D1. *The parametric depth error model obtained for one focal length can be scaled by a factor s_f to generate a corresponding model for a different focal length.*

The parametric depth error model for the $2.1\,\text{mm}$ lens, reported in [29], is given in Eq. 10, with the coefficients a and b for each resolution listed in Tab. II.

$$f(Z) = a \cdot e^{bZ} \tag{10}$$

Here, $f(Z)$ denotes the root mean square (RMS) error at depth Z. Since the study in [29] provides coefficients only for the short-range lens, equivalent values for the long-range lens were extrapolated. This was done by applying a focal length scaling factor s_f, based on the inverse proportionality between depth error σ_z and focal length[30].

$$s_f = \frac{f_s}{f_l} \qquad \sigma_z \propto \frac{1}{f} \tag{11}$$

Here, f_s and f_l denote the focal lengths of short- and long-range lenses, respectively. The coefficients are adjusted to preserve the RMS error magnitude at the respective minimum and maximum depths of the lens, resulting in the long-range lens coefficients shown in Tab. II.

TABLE II
THE STEREO CAMERA ERROR COEFFICIENTS FOR THE DIFFERENT RESOLUTIONS. FOCAL LENGTH 2.1MM VALUES OBTAINED FROM [29]. FOCAL LENGTH 4.0MM EXTRAPOLATED FROM 2.1MM COEFFICIENTS.

Resolution [px]	**Coeff. f_s**	**Coeff. f_l**
2208×1242	a = 0.01805 b = 0.1746	a = 0.009476 b = 0.1746
1920×1080	a = 0.01060 b = 0.2215	a = 0.00557 b = 0.2215
1280×720	a = 0.01840 b = 0.2106	a = 0.00966 b = 0.2106
672×376	a = 0.01150 b = 0.2986	a = 0.00604 b = 0.2986

Assumption D2. *The measurement noise of the depth sensor is assumed to be Gaussian with zero mean and a standard deviation given by the RMS error predicted by the parametric model:*

$$\tilde{Z} = Z + \epsilon, \quad \epsilon \sim \mathcal{N}\big(0, \sigma(Z)^2\big), \tag{12}$$

where $\tilde{Z}$ is the observed depth and $\sigma(Z)$ is depth-dependent.

Works such as [31] note that depth camera noise is not strictly Gaussian and may be better modelled using a Student's t-distribution[32]. For simplicity, this paper still adopts the Gaussian assumption. However, if a more accurate representation is required, the framework allows for modular adaptation of the sensor model.

Assumption D3. *The minimum measurable depth of both lenses is ignored and treated as zero.*

This simplification reduces the model complexity despite introducing a deviation from the true camera behaviour.

Assumption D4. *The effective depth sensing range, beyond which invalid (NaN) values are generated, is assumed to scale proportionally with the focal length in the same manner as the parametric error model.*

The authors of [29] reported that, for the short-range lens at all resolutions, *NaN* values began to appear at ranges exceeding $15\,\text{m}$. This distance is therefore defined as the effective depth sensing range in this work. To remain consistent with Assumption D1, the corresponding range for the long-range lens is obtained by scaling with the factor s_f defined in Eq. 11, giving an approximate range of $28.5\,\text{m}$. Since this value does not represent a hard limit of the camera, the effective range is modelled as a normal distribution centred at $28.5\,\text{m}$ with a standard deviation of $1\,\text{m}$.

Assumption D5. *The FOV is approximated as circular with a uniform opening angle equal to the diagonal (D) FOV, instead of the true rectangular shape defined by horizontal and vertical angles.*

This assumption reduces geometric complexity at the cost of overestimating FOV coverage.

To summarise, with Assumptions D1–D5, the depth camera is modelled in a simplified yet physically grounded manner, with

parameters derived from real specifications. Integrating this model into the interception pipeline captures four key real-world perception effects: sensor noise, sensing update rates, finite sensing range, and restricted field of view.

Kalman Filter – Constant Acceleration

To complete the perception pipeline, a filter is applied to smooth the noisy sensor data. Since the filter must perform uncooperative tracking without access to the target's control inputs, a target manoeuvre model must be assumed. Any sensor noise or deviation from the expected target motion is then treated as process noise. For this purpose, a standard Constant Acceleration (CA) Kalman Filter (KF) is employed to model and estimate the target trajectories. The CA model can represent stationary, constant velocity, and constant acceleration trajectories, making it suitable for general uncooperative motion estimation.

The CA dynamic model [33] can be described by the following expressions.

$$
\begin{aligned}
x &= [\mathbf{p}, \mathbf{v}, \mathbf{a}]^T, \qquad \mathbf{F} = \begin{bmatrix} \mathbf{I} & \Delta t\,\mathbf{I} & \frac{1}{2}\Delta t^2\,\mathbf{I} \\ \mathbf{0} & \mathbf{I} & \Delta t\,\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \\
\mathbf{H} &= [\mathbf{I} \quad \mathbf{0} \quad \mathbf{0}], \quad \mathbf{R} = \mathbf{L} \begin{bmatrix} \sigma_z^2 & 0 & 0 \\ 0 & \sigma_{lat}^2 & 0 \\ 0 & 0 & \sigma_{lat}^2 \end{bmatrix} \mathbf{L}^T
\end{aligned} \tag{13}
$$

The CA dynamic model is described by the state vector x, where $\mathbf{p}, \mathbf{v}, \mathbf{a} \in \mathbb{R}^3$ represent the estimated position, velocity, and acceleration of the target. The state transition matrix $\mathbf{F}$ captures constant acceleration motion, with $\mathbf{I}$ and $\mathbf{0}$ denoting the three-dimensional identity and null matrices. The observation matrix $\mathbf{H}$ indicates that the KF receives only position measurements. The measurement noise covariance $\mathbf{R}$ is initially defined in the sensor's line-of-sight frame, where σ_z represents depth uncertainty along the viewing axis and σ_{lat} the lateral uncertainties orthogonal to it. The rotation matrix $\mathbf{L}$ transforms $\mathbf{R}$ into the world frame, ensuring the KF correctly interprets the depth and lateral uncertainties relative to the global coordinate system. The process noise covariance matrix $\mathbf{Q}$ is derived under the assumption that deviations of the target's acceleration from the constant-acceleration model can be modelled as continuous-time white noise (see [33] for a full derivation):

$$
\mathbf{Q} = q \begin{bmatrix} \frac{1}{20}\Delta t^5\mathbf{I} & \frac{1}{8}\Delta t^4\mathbf{I} & \frac{1}{6}\Delta t^3\mathbf{I} \\ \frac{1}{8}\Delta t^4\mathbf{I} & \frac{1}{3}\Delta t^3\mathbf{I} & \frac{1}{2}\Delta t^2\mathbf{I} \\ \frac{1}{6}\Delta t^3\mathbf{I} & \frac{1}{2}\Delta t^2\mathbf{I} & \Delta t\mathbf{I} \end{bmatrix} \tag{14}
$$

Here, Δt is the timestep size and q is a scalar tuning parameter.

B. RL Formulation

To train a DRL controller, the inputs, outputs, and objective function must be clearly defined. The following subsections describe these components in detail, defining the action space, observation design, and reward function that together characterise the learning setup.

Action Space

Following the traditional autonomous pipeline shown in Fig. 2, learning-based controllers replace one or more modules with a neural network. In this work, the SRT architecture is proposed, where both the guidance and control modules are replaced by a single network that directly generates motor commands for each rotor. This removes all intermediate control layers, enabling full exploitation of the drone's actuation authority and eliminating intermediate bottlenecks. While sim-to-real transfer has historically been challenging, recent results [15]–[17] show that SRT can achieve robust real-world performance, motivating its adoption in this study. The proposed pipeline is illustrated in Fig. 5.

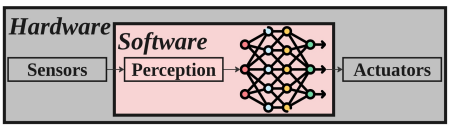

Fig. 5. Proposed interceptor pipeline with Single Rotor Thrust (SRT) neural controller.

Here the neural network outputs four normalised motor commands $u_i \in [-1, 1]$.

Observation Design

Since the interception task is formulated as a partially observable Markov decision process (POMDP), the structure of the observation vector depends on the sensing conditions defined in the scenario. Under ideal conditions, the base observation vector is defined as a 26-dimensional vector $\mathbf{o}_{\text{ideal}}$. All body-frame quantities are expressed in the interceptor's body frame, indicated by a superscript b.

$$
\mathbf{o}_{\text{ideal}} = [\,\hat{\mathbf{r}}^b,\ d,\ \mathbf{v}_I^b,\ \mathbf{v}_T^b,\ \mathbf{q}_I,\ \boldsymbol{\Omega}_I^b,\ \boldsymbol{\omega}_I^b,\ \mathbf{u}_{t-1},\ t_k\,] \tag{15}
$$

where:

- $\hat{\mathbf{r}}^b \in \mathbb{R}^3$ – unit LOS vector from interceptor to target in the body frame
- $d \in \mathbb{R}$ – distance to the target
- $\mathbf{v}_I^b, \mathbf{v}_T^b \in \mathbb{R}^3$ – velocities of the interceptor and target in the body frame
- $\mathbf{q}_I \in \mathbb{R}^4$ – attitude quaternion of the interceptor
- $\boldsymbol{\Omega}_I^b \in \mathbb{R}^3$ – body angular rates of the interceptor
- $\boldsymbol{\omega}_I^b \in \mathbb{R}^4$ – propeller angular velocities
- $\mathbf{u}_{k-1} \in \mathbb{R}^4$ – control inputs from previous timestep
- $t_k \in \mathbb{R}$ – time elapsed since the start of the engagement

Alongside the essential state information of the interceptor and target, the previous actions $\mathbf{u}_{k-1}$ and the elapsed time t_k are appended to the observation vector. Including the previous actions provides direct access to the recent motor commands, which enables smoother control behaviour and complements the reward penalties on high action variation. Similarly, appending the elapsed time places a timestamp on the received input, enabling the network to infer the time-based relationships from the observations.

Ideal conditions assume perfect state estimation and that target information is externally provided with no error at

each control timestep. As the modelled perception module is introduced in training for more realistic target tracking, the observation vector is extended to a 34-dimensional vector, $\mathbf{o}_{real}$.

$$\mathbf{o}_{\text{real}} = [\, \mathbf{o}_{\text{ideal}}, \boldsymbol{\sigma}^2_{\text{pos}}, \boldsymbol{\sigma}^2_{\text{vel}}, t_{\text{KF}}, d_f \,] \tag{16}$$

where $\boldsymbol{\sigma}^2_{\text{pos}}, \boldsymbol{\sigma}^2_{\text{vel}} \in \mathbb{R}^3$ denote the position and velocity variance estimates from the KF, $t_{\text{KF}} \in \mathbb{R}$ is the time elapsed since the last KF update, and $d_f \in \{0, 1\}$ is an integer flag indicating whether the depth measurements are available. Adding these additional variables to the observation vector enables the drone to understand the KF confidence and interpret the lack of depth information when out of range. Finally, to incorporate temporal information, frame stacking of size n_{stack} is applied. This means that the current observation vector is provided together with the vectors from the previous $n_{\text{stack}}-1$ timesteps, resulting in a flattened observation of size $o_{\text{size}} \times n_{\text{stack}}$. In this work, a stack size of $n_{\text{stack}} = 3$ was used to provide temporal context without overwhelming the network with inputs.

Reward Design

The network was trained using the following reward function:

$$r_k = \begin{cases} +\,(t_{\max} - t_k), & \text{if intercepted,} \\ -\,\frac{d_{k_{\text{clip}}}}{r_{\text{int}}}(t_{\max} - t_k), & \text{if failed,} \\ +\,c_d r_{\text{dist}} - c_\Omega p_\Omega - c_u p_u, & \text{otherwise,} \end{cases} \tag{17}$$

Here, $t_{\max}$ denotes the maximum number of timesteps per episode, t_k is the current timestep, $d_{k_{\text{clip}}}$ is the distance at that timestep clipped to the maximum miss value, and r_{int} is the interception radius. The clipped distance $d_{k_{\text{clip}}}$ ensures that the failure penalty scales with the miss distance whilst remaining bounded. The progressive distance reward r_{dist} is defined as follows:

$$r_{\text{dist}} = \tanh\left(\frac{d_{k-1} - d_k}{v_{cl_{safe}} \cdot \Delta t_{ctrl}}\right), \qquad d_k = ||\mathbf{p}_T - \mathbf{p}_I|| \tag{18}$$

where $v_{cl_{safe}}$ is a parameter used to set a "safe" closing velocity that is used to normalise the change in distance. The hyperbolic tangent ensures smooth normalisation past $v_{cl_{safe}}$ but gives decreasing marginal gains for exceeding it. This distance reward is invariant to range, as it rewards the change in distance and enables user tuning of the pursuit intensity. The rate penalty p_Ω and action smoothness penalty p_u are given by the following expressions:

$$\begin{aligned} p_\Omega &= \sum_{i=1}^{3} \left(\min\left(\frac{|\Omega_i|}{\Omega_{r_{\max}}}, 1\right)\right)^2 \\ p_u &= \frac{1}{16}\sum_{i=1}^{4} (\Delta u_i)^2. \end{aligned} \tag{19}$$

where $\Omega_{r_{\max}}$ defines the point at which the maximum penalty for rotational rates is applied, which helps penalise high rates earlier. The scaling coefficients c_d, c_Ω and c_u weight the respective reward components.

In Eq. 17, the interception and failure terms provide large, sparse signals defining the episode outcome, with any non-interception termination classified as failure. The dense terms are deliberately simple to avoid biasing the interception strategy, while the penalties discourage valid but undesirable behaviours. Excessive shaping of the reward function is avoided, since overlapping objectives can lead to unclear learning signals and reduce interpretability. Additionally, the reward function is scaled such that the maximum achievable return in an episode always equals $t_{\max}$. Dense rewards are bounded to 1, and a successful interception grants a constant reward of 1 for all remaining timesteps, effectively encouraging earlier interception with maximum return. This structure enables more interpretable reward shaping and allows more intuitive tuning of the training objective.

C. Training for Robustness

Domain Randomization (DR)[13] is a widely used technique to improve robustness and generalisation during training. By exposing the agent to variability in the environment and system parameters, DR encourages the learned policy to achieve the task objective whilst still generalising to a range of conditions. It can and should be applied to every aspect of the environment where uncertainty or variability is expected. Introducing such randomization is crucial for transferring policies from simulation to reality, as it accounts for inevitable discrepancies between the two domains. The following subsection details the randomization strategies applied to the drone platform, depth camera, initial conditions, and control frequency.

Drone Platform

The randomization strategy for the drone platform utilises the concepts and parameters presented in [15]. Based on real-world manual flight test data, the model parameters of 3-inch and 5-inch racing drones were obtained and normalised using the maximum propeller speeds $\omega_{\max}$. From these normalised parameters, the following randomization policies were defined:

- **General Policy**: A policy with a wide range of randomization that spans both the 3-inch and 5-inch domains. This produces a robust solution that generalises well to differences in scale and dynamics. The normalised parameter ranges are reported in Tab. IX.
- **Finetuned Policy**: A policy that utilises the true platform parameters with controlled randomization around the nominal values. The randomization levels, discussed in [15], are fixed at $[0, 10, 20, 30]\%$ of the nominal values, enabling analysis of the robustness–performance tradeoff.

While [15] introduced the concepts of General and Finetuned policies primarily for comparative evaluation, in this work they are applied sequentially. The *General Policy* is first trained to provide a broad, robust understanding of drone dynamics and control across platforms. The *Finetuned Policy* is then initialised from this base and specialised to the true platform parameters. This ensures that all policies are grounded in the same generalised representation of the drones before specialising to their platform-specific dynamics.

Depth Camera

To account for uncertainty in the depth camera, particularly in the true error parameters and the effective sensing range, a $\pm 10\%$ randomization is applied to the a coefficient in Tab. II, as well as the sensing range discussed in assumption **D4**. During training, the KF continues to use the nominal estimated coefficients, thereby modelling the expected discrepancy between the assumed sensor characteristics and the true errors. This approach effectively incorporates sensor variability into the framework, promoting policy robustness to various levels of measurement noise and sensing ranges.

Initial Conditions

Variability in initial conditions is crucial for policy generalisation and for ensuring robustness to both nominal and edge cases. Assuming the mid-course of the interception is executed using PN, the initial relative positions and velocities are randomly sampled from feasible PN conditions. Matching the initial conditions to valid PN states ensures a seamless transition to the homing phase when the DRL policy takes control in real applications. Defining and sampling valid PN initial conditions is rarely discussed in the literature, as most treatments focus on the guidance law itself rather than on the feasible set of interceptor velocities. For clarity and reproducibility, a detailed description of the approach used to sample valid PN conditions is provided in App. C.

To define realistic approaches, the interceptor position is uniformly sampled from a hemispherical shell beneath the target, with inner and outer radii of 25m and 35m, respectively. Sampling is performed using the standard technique for uniform sampling in a spherical volume, ensuring uniform density[34]. The above starting range is chosen to cover the transition from distances where depth measurements are unavailable, past the depth sensing range of 28.5m, to distances where depth outputs are available. By training the DRL policy across this initial distance range, the agent learns to handle the complex transition in sensing conditions. This further demonstrates the potential of DRL under changing sensory inputs, where traditional methods may struggle. The initial interceptor attitude is randomly interpolated between the upward direction and its velocity vector, introducing variability in the initial flight orientation. The initial body rates are sampled as $\boldsymbol{\Omega_I} \sim \mathcal{U}(-0.2, 0.2)$ and the normalised propeller speeds as $\boldsymbol{\omega} \sim \mathcal{U}(-0.5, 0.5)$, adding small random variations in both rotation and thrust.

Control Frequency

To improve robustness to timing mismatches between simulation and real hardware, the control frequency was randomized within $\pm 10\%$ of the nominal value during training. This prevents the policy from exploiting fixed timing assumptions and justifies the inclusion of the elapsed time t_k in the observation vector.

D. RL Algorithm

For training the network, *Proximal Policy Optimization* (PPO)[35] was chosen due to its rapid learning and efficient parallelism. The specific choice of algorithm is not the focus of this work and PPO is employed primarily to demonstrate the proposed DRL framework and its ability to surpass classical approaches. To emphasise this, the implementation uses the Stable-Baselines3[36] library in Python with mostly default arguments[2], illustrating that minimal hyperparameter tuning is required but can be done to tailor performance. The only algorithm parameters changed were the network architecture, the PPO clip range ϵ and the discount factor γ.

- **Network Architecture**: Prior work in drone racing and low-level control commonly employs networks of 2–3 hidden layers with 64 neurons each[15], [16]. In contrast, the presented interception task involves a larger observation space and higher task complexity. The agent must reason over both pursuit dynamics and uncertainty in the environment. To provide sufficient representational capacity for these requirements, a larger network of 3 layers of 512 neurons was used.
- **Clip Range** ϵ: Since PPO can be susceptible to instability during training, the default clip range of 0.2 was reduced to 0.05, constraining policy updates to at most 5% change per step. This adjustment significantly improved training stability without noticeably slowing convergence.
- **Discount Factor** γ: The discount factor determines how future rewards are valued relative to immediate ones. A value close to 0 yields short-sighted, greedy policies, while values approaching 1 encourage long-term planning[37]. The default PPO setting of $\gamma = 0.99$ implies that a reward received 100 steps in the future is weighted as 37% of its original value. To increase the agent's planning horizon, γ is raised to 0.999, for which the 100 step weighting remains 90%. This enables a longer planning horizon and should lead to more effective interceptions.

VI. Simulation Setup

This section describes the implemented simulation framework (Subsec. VI-A), the defined interception episode (Subsec. VI-B), and the training curriculum (Subsec. VI-C) employed to enable effective learning under the final interception conditions. The presented implementations are openly available on GitHub[1].

A. Simulation Framework

A custom vectorised environment was developed based on the quadcopter model and dynamics described in Sec. III, enabling the parallel simulation of over 1000 drones. All environments and their corresponding states are batched into large arrays, allowing efficient vectorised operations using *NumPy*[38]. Computationally intensive components, including physics propagation, KF updates, and reward computations, were further accelerated by compiling them with *JAX*[39] and *Numba*[40], significantly reducing the overall training time. The simulation control frequency was set to vary around 100 Hz, with the physics frequency defined as twice the control

[2]https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

frequency (two physics steps per control step), balancing computational load and simulator fidelity. This configuration can easily be adjusted depending on available computational resources.

The framework was inspired by the *Pybullet Drones*[41] repository and the methods utilised in [15]. Combining these approaches created a unique, fully vectorised drone simulation platform tailored for large-scale DRL training for interception tasks.

B. Interception Episode

Each training episode involves one interceptor drone and one target drone. The interceptor begins with the initial conditions described in Subsec. V-C and has a maximum of 10 seconds to achieve a valid interception, as defined in Subsec. III-B. Episodes are terminated early if any of the following conditions are met: an interception is achieved ($r_{\text{int}} = 0.3\text{m}$), the drone exceeds the environment boundaries, the body-rate limits are exceeded (30rad/s), or a miss occurs. A miss is defined as entering a sphere of radius 2m around the target while having an increasing distance. This forces commitment from the drone in the final 2m. The radius of the miss sphere can be tuned to force the interceptor to commit earlier or later, controlling how much freedom the agent has before finalising the interception.

For each episode, the interceptor is randomly faced with one of three target motion profiles: stationary, constant velocity (CV), or constant acceleration (CA). The corresponding motion bounds are listed in Tab. III.

TABLE III
TARGET TRAJECTORY PARAMETER BOUNDS.

Trajectory	Horizontal Max	Vertical Max	Total Max
CV (m/s)	19	6	27.5
CA (m/s^2)	3.5	1.5	5.2

The horizontal velocity or acceleration components in the x and y directions are sampled independently within the listed bounds. Consequently, the reported total represents the L2-norm of the horizontal and vertical components combined. For CA trajectories, velocities are sampled using the same initial velocity limits as the CV case and then constrained such that the resulting velocity does not exceed the predefined maximum. Thus, although the profile is labelled "constant acceleration," it must be bounded to allow meaningful acceleration without permitting unrealistic speeds.

C. Curriculum Learning

The interception task is inherently complex, and exposing an untrained neural network directly to the full difficulty overwhelms the learning process. This makes it nearly impossible for the agent to acquire meaningful behaviour. To address this, *Curriculum Learning*[42] is used, where the agent is first exposed to simpler versions of the task before gradually progressing to harder ones. By building on experience from earlier stages, the agent can learn more effectively in the subsequent ones. The proposed curriculum consists of 11 consecutive stages, which can be grouped into three main phases:

- **LearnToFly:** The first four stages feature a slow-moving target while the interceptor's initial attitude, body rates, and propeller speeds are highly randomized. This encourages the agent to first master platform dynamics and low-speed control. Across levels, the initial distance to the target is increased and then misses are introduced. This phase is trained using the General Policy introduced in Subsec. V-C.
- **TargetSpeedUp:** The following six stages gradually increase the target speed from an initial limit of $7\,\text{m/s}$ to $27.5\,\text{m/s}$. Dividing this range over multiple stages enables smoother policy learning. Greater weight is assigned to sampling CV and CA trajectories, as these require additional training focus. Training in this phase is performed using the Finetuned Policy, transferring from what was learned from the previous phase.
- **Finetuning:** The final stage exposes the agent to the full task complexity. Most of the total training occurs here. The Finetuned Policy is used throughout this phase.

Overall, training is limited to 1 billion timesteps, approximately 3000 simulation hours, and progression to the next stage only occurs once the predefined interception success rate is met. The final stage continues until the timestep budget is exhausted. While extending training beyond 1 billion timesteps yields further improvements, the limit was imposed to ensure computational feasibility and fair comparison.

VII. RESULTS

In the following section, the quantitative results obtained from training using the presented methodology are summarised. First, a baseline comparison between the two controller types is presented under ideal conditions to isolate any effects of environmental uncertainty. Next, under similar conditions, the controllers' transferability is evaluated when trained with different levels of platform randomization. Finally, the performance of both controllers with the integrated perception module is assessed.

Note that, due to the stochastic nature of reinforcement learning training, minor deviations between results are expected. The obtained results should be interpreted as statistical tendencies rather than deterministic outcomes. All results are generated on the same set of 1000 randomized environments for fair comparison. As each episode is limited to a maximum of 10 seconds, the highest achievable reward in the episode is 1000. Lower rewards indicate lower interception rates, longer interception times or higher cumulative penalties due to less desirable behaviour.

A. Baseline Ideal Conditions

To evaluate the capabilities of both the classical PN and learned DRL controllers, the systems are first trained and tested under ideal conditions. This setup assumes perfect state information without sensor noise, perception uncertainty, or FOV limitations, effectively simulating an external motion capture system. Under these conditions, PN guidance operates

near optimally, as it can precisely follow geometrically ideal interception approaches.

The PN controller benefits from a high-frequency 500 Hz inner attitude loop and an exact motor mixing model based on true drone parameters. This enables precise execution of its commanded accelerations. In contrast, the SRT DRL policy directly outputs the learned motor commands at 100 Hz, making it inherently more challenging to achieve the same level of control precision. The results are presented in Tab. IV.

TABLE IV
CONTROLLER PERFORMANCE COMPARISON UNDER IDEAL CONDITIONS.

Model	Rate (%)	Length (s)	Reward (-)	RVC (-)	Angle (°)
PN	99.6	1.53	973	-0.950	57.8
DRL	99.4	2.18	940	-0.898	39.1

Here, RVC denotes the *Range–Vector Correlation*, a metric that quantifies how well the trajectory aligns with the geometric approach utilised in PN guidance:

$$RVC = \frac{r \cdot v_{\text{rel}}}{|r||v_{\text{rel}}|} \tag{20}$$

where r is the relative position vector and v_{rel} the relative velocity vector between interceptor and target. The interception angle represents the relative orientation between the interceptor and target velocity vectors at interception, with $0°$ indicating pursuit motion and $180°$ a head-on approach.

Although both controllers achieve nearly identical interception rates, their average rewards differ due to variations in interception time and the extent to which each trajectory minimises the penalty terms defined in Subsec. V-B. Table IV shows that the DRL controller performs the interception noticeably slower and deviates more from the geometric PN approach. This behaviour is further illustrated in Fig. 6, where the DRL controller adopts a less orthogonal trajectory, approaching the target more from behind, indicative of a more conservative interception strategy.

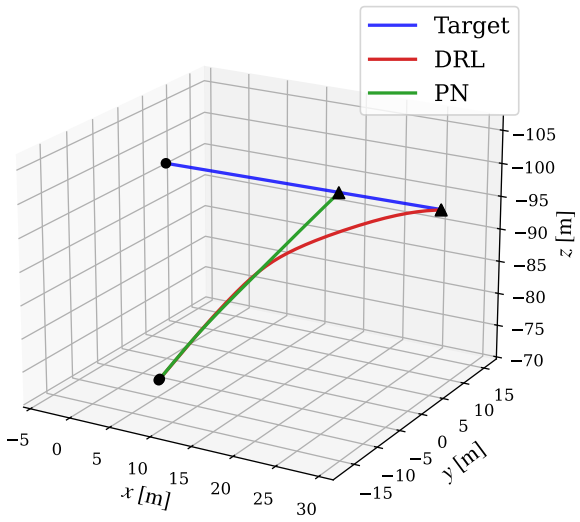

Fig. 6. Interception engagement of PN (green) and DRL (red) controllers on constant velocity Target (blue). The start and end of the trajectories are illustrated with a circle and triangle, respectively.

TABLE V
TRANSFERABILITY PERFORMANCE UNDER IDEAL CONDITIONS FOR 3-INCH (TRAINING) AND 5-INCH (TRANSFER) DRONES.

3-inch (Training Platform)				
Model	**Rate (%)**	**Length (s)**	**Reward (-)**	**Angle (°)**
DR0%	99.4	2.18	940	39.1
DR10%	99.6	2.12	941	37.6
DR20%	99.9	2.33	938	29.9
DR30%	100.0	2.63	918	29.6
5-inch (Transfer Platform)				
DR0%	38.4	1.99	-474	34.7
DR10%	83.7	1.94	599	38.2
DR20%	97.0	2.20	873	30.6
DR30%	99.6	2.58	910	31.0

B. Transferability under Ideal Conditions

To evaluate the effectiveness of different levels of DR in the drone platform, the models are first tested under ideal conditions. This isolates the influence of DR from perception noise and environmental uncertainty, allowing a clear assessment of how increasing variability during training affects both performance and transferability. The levels of DR tested were $\pm[0, 10, 20, 30]\%$ around the nominal parameters of the 3-inch racer. Although the controllers were trained with DR, their final performance was assessed on the nominal 3-inch and 5-inch platforms without randomization to enable a fair and consistent comparison. The 3-inch drone serves as the *Training Platform*, demonstrating the controller's performance under familiar configurations, while the 5-inch drone represents the *Transfer Platform*, highlighting the controller's robustness and generalisation capability to a platform with different dynamics. The corresponding results are presented in Tab. V. The PN controller is omitted in this analysis, as its parameters can be explicitly retuned for each platform, unlike the learning-based controllers whose transferability must emerge from training.

From Tab.V, clear trends emerge between the level of DR and the resulting performance and transferability. On the *Training Platform*, higher DR levels achieve slightly higher interception rates but at the cost of longer and more conservative trajectories, as reflected in the lower average interception times and reward values. This behaviour results from the controller learning to generalise across a broader range of platforms, which limits its ability to fully exploit the actuator capabilities of the drone. The benefit of this generalisation becomes evident on the *Transfer Platform*, where performance improves substantially with higher DR levels, reaching near-equal results across both platforms when $\pm 30\%$ DR is applied during training.

In Fig. 7, the trajectories of the different controllers in an identical interception scenario are shown. A clear trend of

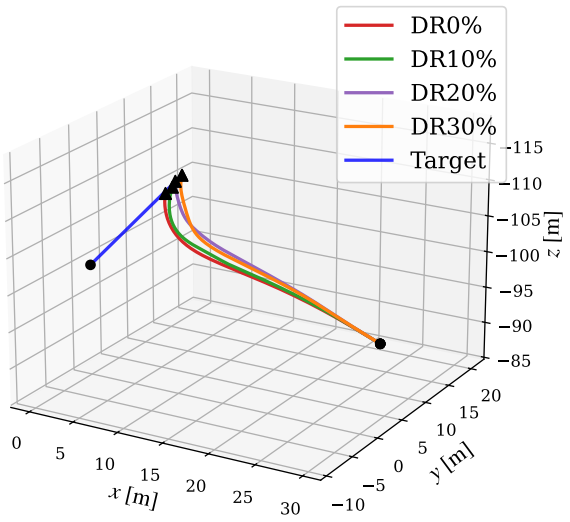

Fig. 7. Interception engagements of controllers trained with different domain randomization (DR) levels on the nominal 3-inch drone platform.

increasingly cautious interception behaviour is evident. Higher levels of DR lead to lower average interception angles and more gradual, controlled approaches. These results validate the effectiveness of DR in enabling robust interception behaviour across platforms with different dynamic characteristics. It confirms that the learned policy captures a generalised control strategy rather than overfitting to a single platform.

C. Integrated Perception Module

In real interception scenarios, the perception pipeline is far from ideal, and controllers must operate effectively under realistic sensing constraints. To assess this, the DRL controller was trained with the modelled perception module in the loop. This enables evaluation of its ability to learn and act under uncertain and imperfect perception. The perception model and its assumptions are defined in Subsec. V-A.

The modelled stereo camera is available in multiple resolutions that trade-off depth accuracy against maximum FPS. To study this trade-off, DRL controllers were trained at different camera resolutions to evaluate the effect of sensing accuracy and frame rate on robustness and interception performance. For this experiment, the long-range lens was used, corresponding to a FOV of 81° and the f_l coefficients in Tab. II. For comparison, the classical PN controller was also evaluated using the same perception models. The results are summarised in Tab. VI.

In Tab. VI, the FOV metric denotes the percentage of the episode in which the target is within the interceptor's field of view. This metric highlights the trade-off each controller performs between keeping sight of the target and effectively approaching the target. The DRL controller maintains interception rates above 95% for all resolutions, while PN drops from 65% to 45% as resolution increases and the frame rate decreases. As illustrated in Fig. 8, the DRL controllers operating with lower-resolution configurations prioritise maintaining closer proximity to the target before initiating the final interception approach. This behaviour compensates for the reduced depth accuracy at longer ranges. In contrast, the PN controller exhibits greater sensitivity to sensing limitations, as reflected in reduced FOV percentage and reward values. As its guidance law does not account for perception constraints, visibility loss frequently occurs during the approach. To investigate this dependency further, the best-performing PN configuration (1280×720) was compared to its DRL counterpart across different engagement geometries and target dynamics, in Tab. VII.

TABLE VI
PERFORMANCE OF PN AND DRL CONTROLLERS UNDER THE FULL PERCEPTION MODEL FOR DIFFERENT STEREO CAMERA RESOLUTIONS.

PN				
Res. [px]	**Rate (%)**	**Length (s)**	**Reward (-)**	**FOV (%)**
1280×720	65.5	3.79	266	52.5
1920×1080	63.6	3.85	220	51.2
2208×1242	44.9	5.36	-117	45.2
DRL				
1280×720	95.0	2.70	761	71.2
1920×1080	95.6	2.87	757	76.3
2208×1242	96.0	2.66	782	77.2

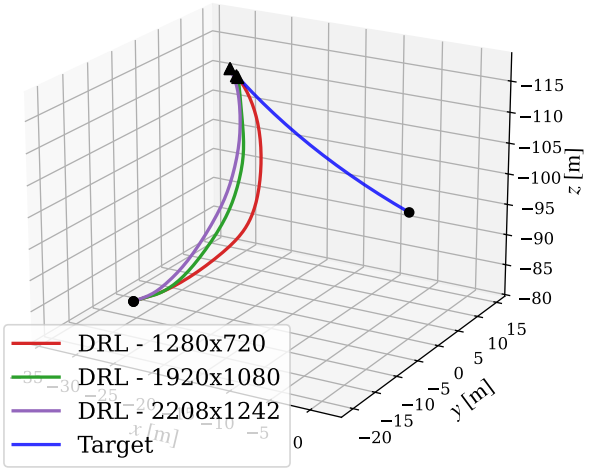

Fig. 8. Interception engagements of controllers trained with different camera resolution configurations on the nominal 3-inch drone platform.

TABLE VII
INTERCEPTION SUCCESS RATE (%) OF PN AND DRL CONTROLLERS WITH THE REALISTIC PERCEPTION MODULE (1280×720) FOR DIFFERENT ENGAGEMENT GEOMETRIES AND TARGET DYNAMICS.

Model	**Pursuit**	**Crossing**	**Head-On**	**High Vel.**	**High Acc.**
DRL	96.8	96.0	91.5	88.9	85.4
PN	55.9	71.5	56.3	37.9	45.5

In Tab. VII, pursuit, crossing, and head-on refer to initial engagement geometries divided into 60° regions: behind, orthogonal to, and in front of the target. The high velocity and high acceleration cases represent the upper bounds of the constant-velocity (21–28.5 m/s) and constant-acceleration (4.2–5.7 m/s^2) target trajectories. The results show that PN's success rate varies with initial geometry and target dynamics. Both pursuit and head-on conditions show a clear drop in performance. Furthermore, at higher velocities and accelerations, PN performance degrades significantly due to the more demanding manoeuvres required, which often cause loss of target visibility. The DRL controller maintained high success rates across all challenging conditions, indicating consistent performance and hence robustness to complex manoeuvres and varying engagement scenarios.

VIII. Discussion

The following section provides a qualitative interpretation of the findings from Sec. VII.

Under ideal conditions, both controllers achieved comparable interception rates (Tab. IV), indicating that the DRL policy can successfully learn to intercept within the presented framework and target trajectories. However, their differing control characteristics directly influenced the efficiency of the interception manoeuvres, as shown in Fig. 6. The PN controller, benefiting from a high-frequency attitude loop and explicit geometric guidance, executed more direct trajectories toward the target. In contrast, the DRL controller's lower control frequency encouraged more risk-averse interception paths. Rather than reflecting inferior performance, this behaviour suggests that the policy implicitly learned to trade interception speed for stability and safety to achieve higher cumulative rewards. This behavioural distinction becomes even more pronounced when the DRL policy is trained with different levels of DR on the interceptor platform. The results on the training platform in Tab. V demonstrate that increasing platform randomization produced progressively more cautious controllers. Requiring the DRL policy to generalise over a wider range of drone configurations reduces the effective control authority that the policy can exploit and thus leads to more risk-averse trajectories.

The ability of the DRL controller to generalise is crucial for ensuring successful simulation-to-reality transfer. The learned policies must remain robust to variations in interceptor dynamics, as discrepancies between the simulated and real physical systems are inevitable. The policies were tested on the 5-inch drone despite being trained on the 3-inch platform (Tab. V). The results demonstrated the effectiveness of using domain randomization to enforce generalisability during training. The controller's ability to perform on the 5-inch drone reflects its potential for bridging the gap between simulation and reality, consistent with the approach and results reported in [15].

To assess the DRL controller's robustness under uncertainty, the proposed perception module was integrated into the training loop, and its performance was compared against the PN controller. Incorporating this module introduced four key real-world perception effects into the simulation: sensor noise, sensing rate limitations, finite sensing range, and restricted field of view. Under these realistic constraints, the DRL controller consistently outperformed the PN controller across all camera resolutions (Tab. VI). The restricted field of view was by far the most disruptive constraint, drastically affecting the performance of the classical PN controller. This is because the classical guidance is not FOV-aware, and explicitly implementing the trade-off between target tracking and efficient approach is highly complex and difficult to tune for acceptable results. In contrast, the DRL policy can implicitly learn to balance this trade-off efficiently and consistently approach the target. This balance is reflected in the FOV percentages achieved by the DRL controller in Tab.VI, which remain between 70–80%, indicating that it is not necessarily optimal to keep the target continuously in view. The learned policy ensures sufficient tracking to intercept, whereas the classical controller's success relies heavily on favourable initial conditions and target dynamics. This sensitivity becomes evident in Tab. VII, where the interception rate varies strongly with the initial geometries and degrades with more challenging target trajectories. Meanwhile, the DRL controller maintains consistently high interception rates even at the upper limits of the target's flight envelope. Clear behavioural trends are evident in Fig. 8 with varying camera resolutions. Controllers trained at lower resolutions, affected by higher depth noise, first close the distance to obtain more reliable measurements before executing the final interception. Conversely, higher-resolution configurations can initiate the final approach earlier thanks to improved measurement accuracy, but must maintain target visibility for a larger portion of the episode to compensate for their lower frame rates. These results further demonstrate the learning-based controller's ability to adapt its strategy to the sensing characteristics of the perception system.

The presented framework successfully demonstrates robust interception performance within the implemented setup under more realistic perception constraints. However, the true strength of the approach lies in its modularity. The framework is not inherently bound to a particular reinforcement learning algorithm, sensing modality, quadcopter platform, or initial engagement condition. Substituting the onboard sensor, modifying the drone parameters, or employing a different learning algorithm would require only limited reconfiguration rather than fundamental changes to the overall architecture. Similarly, extending the framework beyond the mid-course–to–homing transition considered in this work would mainly entail redefining the initial conditions within the same training structure. Likewise, the reward formulation used in this work is intentionally generic for interception and can be extended with additional terms to promote more specialised behaviours from the controller.

Despite this, the approach still has several limitations that need to be acknowledged. The most evident is the lack of real-world flight experiments to validate the sim-to-real transfer of the controllers. This limitation stems from time constraints in the project rather than from deficiencies in the framework, yet it remains an essential next step toward practical deployment. Furthermore, the implementation presented in this work relies on several assumptions introduced to reduce computa-

tional complexity during training, such as simplified stereo-camera modelling, an idealised computer-vision algorithm, and a reduced control frequency for the DRL controllers. These assumptions made training feasible on the available hardware but also contribute to the remaining gap between simulation and real-world conditions. Further bridging the gap would require removing these assumptions and incorporating sensor and computational delays into the pipeline. Finally, uncertainty in the interceptor's own state estimation was not explicitly modelled. Given that the dominant source of noise and variability in interception scenarios typically originates from relative target tracking rather than self-state estimation, this effect was considered secondary within the scope of this work.

IX. Conclusion and Future Work

In this work, a comprehensive training framework was developed to generate Single Rotor Thrust (SRT) DRL controllers and to evaluate them against a classical PN implementation. The obtained policies exhibited strong generalisation across platform variations and surpassed the classical baseline under modelled perception constraints, indicating enhanced robustness and transferability.

Future work could incorporate reactive or evasive target manoeuvres to evaluate the controller's adaptability to dynamically changing target trajectories. Additional sources of uncertainty, such as sensor delays and external disturbances, could also be introduced to further narrow the gap between simulation and reality. Another promising direction lies in coupling the framework with a real computer-vision algorithm, or an accurately modelled equivalent, to form a complete integrated perception pipeline. Ultimately, validation on real quadrotor platforms is essential to confirm the framework's practical effectiveness and enable deployment in real-world interception scenarios.

Appendix A
ZEM Calculation

The zero-effort miss (ZEM) is defined as the miss distance achieved assuming no more control inputs are executed. It is mathematically defined by the following equation [12]:

$$\text{ZEM} = \|\mathbf{r} + t_{go}\mathbf{v}_{\text{rel}}\|,$$

where $\mathbf{r}$ is the line-of-sight vector from interceptor to target, $\mathbf{v}_{\text{rel}}$ is the relative velocity vector between interceptor and target, and t_{go} is the time until the closest approach is reached. Finding t_{go} is derived below:

$$\mathbf{r}_{\text{closest}} \cdot \mathbf{v}_{\text{rel}} = 0.$$

$$(\mathbf{r} + t_{go}\mathbf{v}_{\text{rel}}) \cdot \mathbf{v}_{\text{rel}} = 0,$$

$$\mathbf{r} \cdot \mathbf{v}_{\text{rel}} + t_{go}\|\mathbf{v}_{\text{rel}}\|^2 = 0.$$

$$t_{go} = -\frac{\mathbf{r} \cdot \mathbf{v}_{\text{rel}}}{\|\mathbf{v}_{\text{rel}}\|^2}.$$

With the t_{go} found, the ZEM can be calculated.

Appendix B
Drone Parameters

TABLE VIII
Normalised drone parameters for 3-inch and 5-inch platforms [15].

Parameter	3-inch	5-inch	Parameter	3-inch	5-inch
$\hat{k}_\omega$	14.3	27.1	ω_{min} (rad/s)	305.4	238.5
$\hat{k}_x$	0.16	0.16	ω_{max} (rad/s)	4887.6	3295.5
$\hat{k}_y$	0.18	0.24	k_l	0.84	0.95
			τ (s)	0.04	0.04
$\hat{k}_{p1}$	615.0	711.0	$\hat{k}_{r1}$	47.1	35.2
$\hat{k}_{p2}$	598.0	718.0	$\hat{k}_{r2}$	47.1	35.2
$\hat{k}_{p3}$	650.0	691.0	$\hat{k}_{r3}$	47.1	35.2
$\hat{k}_{p4}$	479.0	724.0	$\hat{k}_{r4}$	47.1	35.2
$\hat{k}_{q1}$	217.0	573.0	$\hat{k}_{r5}$	5.57	6.49
$\hat{k}_{q2}$	238.0	637.0	$\hat{k}_{r6}$	5.57	6.49
$\hat{k}_{q3}$	280.0	548.0	$\hat{k}_{r7}$	5.57	6.49
$\hat{k}_{q4}$	196.0	640.0	$\hat{k}_{r8}$	5.57	6.49

TABLE IX
General policy normalised parameter ranges [15].

Parameter	Distribution	Parameter	Distribution
ω_{min}	$\mathcal{U}(0, 500)$	$\hat{k}_\omega$	$\mathcal{U}(10, 30)$
ω_{max}	$\mathcal{U}(3000, 5000)$	$\hat{k}_x$	$\mathcal{U}(0.1, 0.3)$
k_l	$\mathcal{U}(0, 1)$	$\hat{k}_y$	$\mathcal{U}(0.1, 0.3)$
τ	$\mathcal{U}(0.01, 0.1)$		
$\hat{k}_p$	$\mathcal{U}(200, 800)$	$\hat{k}_{p1}, \hat{k}_{p2}, \hat{k}_{p3}, \hat{k}_{p4}$	$\hat{k}_p \pm \mathcal{U}(50)$
$\hat{k}_q$	$\mathcal{U}(200, 800)$	$\hat{k}_{q1}, \hat{k}_{q2}, \hat{k}_{q3}, \hat{k}_{q4}$	$\hat{k}_q \pm \mathcal{U}(50)$
$\hat{k}_r$	$\mathcal{U}(20, 80)$	$\hat{k}_{r1}, \hat{k}_{r2}, \hat{k}_{r3}, \hat{k}_{r4}$	$\hat{k}_r$
$\hat{k}_{rd}$	$\mathcal{U}(2, 8)$	$\hat{k}_{r5}, \hat{k}_{r6}, \hat{k}_{r7}, \hat{k}_{r8}$	$\hat{k}_{rd}$

Appendix C
Sampling Valid Initial Conditions for Proportional Navigation

Assuming the target velocity v_T and the relative position vector are fixed, valid initial conditions for Proportional Navigation (PN) can be sampled by enforcing the closing condition:

$$v_T \cos\theta - v_I \cos\delta < 0,$$

where v_T is the target speed, v_I is the interceptor speed, θ is the angle between the target velocity and the line-of-sight (LOS), and δ is the angle between the interceptor velocity and the LOS. Rearranging the inequality gives an upper bound on the allowable angle δ:

$$\delta_{\text{lim}} = \arccos\left(\frac{v_T \cos\theta}{v_I}\right).$$

For this expression to be valid, the argument of $\arccos(\cdot)$ must lie in $[-1, 1]$, which imposes a lower bound on the interceptor speed:

$$v_{I,\text{min}} = v_T \cos\theta.$$

The interceptor speed is then sampled from a user-defined range:

$$v_I \sim \mathcal{U}(v_{I,\min}, v_{I,\max}),$$

with $v_{I,\max}$ being the maximum allowable interceptor speed. Once v_I is sampled, the corresponding $\delta_{\lim}$ can be computed. Optionally, $\delta_{\lim}$ can be further constrained to the interceptor's FOV to ensure that the target is visible from the start:

$$\delta_{\lim} = \min(\delta_{\lim}, \tfrac{\theta_{\text{FOV}}}{2}).$$

Finally, the interceptor's initial velocity direction relative to the LOS is sampled uniformly within this admissible range:

$$\delta \sim \mathcal{U}(0, \delta_{\lim}).$$

Together, the sampled interceptor speed v_I and heading angle δ fully define the interceptor's initial state relative to the known target velocity and line-of-sight geometry.

REFERENCES

[1] S. M. R. Islam, "Drones on the rise: Exploring the current and future potential of uavs," 2023. [Online]. Available: https://arxiv.org/abs/2304.13702

[2] InsideFPV, "What is a kamikaze drone and how does it work?" https://insidefpv.com/blogs/blogs/what-is-kamikaze-drone-how-does-it-work, May 2025, accessed: Oct. 7, 2025.

[3] T. Burridge. (2019, 2) 'sustained' drone attack closed gatwick, airport says. Accessed: 2025-03-31. [Online]. Available: https://www.bbc.com/news/business-47302902

[4] S. Sprenger, "Interceptor drones are europe's new hope for downing russian shaheds," https://www.defensenews.com/global/europe/2025/09/15/interceptor-drones-are-europes-new-hope-for-downing-russian-shaheds/, Sep. 2025, accessed: Oct. 7, 2025.

[5] A. Kumar, A. Ojha, S. Yadav, and A. Kumar, "Real-time interception performance evaluation of certain proportional navigation based guidance laws in aerial ground engagement," *Intelligent Service Robotics*, vol. 15, no. 1, pp. 95–114, 2022.

[6] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and G. Casalino, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, Apr. 2021.

[7] 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)*, 2022, pp. 10 504–10 510.

[8] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, vol. 40, pp. 3044–3067, 2024.

[9] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 8 2023.

[10] 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.

[11] C. De Wagter, "Autonomous drone from tu delft defeats human champions in historic racing first," 2025, accessed: Oct. 7, 2025. [Online]. Available: https://mavlab.tudelft.nl/dronerace-defeating-champions/

[12] R. Yanushevsky, *Modern Missile Guidance*, 09 2018.

[13] 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)*. IEEE Press, 2017, p. 23–30. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1109/IROS.2017.8202133

[14] A. Ghotavadekar, F. Nekovář, M. Saska, and J. Faigl, "Variable time-step mpc for agile multi-rotor uav interception of dynamic targets," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1249–1256, 2025.

[15] R. Ferede, T. Blaha, E. Lucassen, C. D. Wagter, and G. C. H. E. de Croon, "One net to rule them all: Domain randomization in quadcopter racing across different platforms," 2025. [Online]. Available: https://arxiv.org/abs/2504.21586

[16] 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," 2019. [Online]. Available: https://arxiv.org/abs/1903.04628

[17] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2021, p. 1205–1212. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1109/IROS51168.2021.9636053

[18] 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, Feb. 2023.

[19] Z. Hu, L. Xiao, J. Guan, W. Yi, and H. Yin, "Intercept guidance of maneuvering targets with deep reinforcement learning," *International Journal of Aerospace Engineering*, vol. 2023, no. 1, p. 7924190, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/7924190

[20] C.-L. Chen, Y.-W. Huang, and T.-J. Shen, "Application of deep reinforcement learning to defense and intrusion strategies using unmanned aerial vehicles in a versus game," *Drones*, vol. 8, no. 8, p. 365, 7 2024.

[21] A. A. Darwish and A. Nakhmani, "Drone navigation and target interception using deep reinforcement learning: A cascade reward approach," *IEEE Journal of Indoor and Seamless Positioning and Navigation*, vol. 1, pp. 130–140, 2023.

[22] X. Zhang, H. Guo, T. Yan, X. Wang, W. Sun, W. Fu, and J. Yan, "Penetration strategy for high-speed unmanned aerial vehicles: A memory-based deep reinforcement learning approach," *Drones*, vol. 8, no. 7, p. 275, 6 2024.

[23] X. Zhuang, D. Li, Y. Wang, X. Liu, and H. Li, "Optimization of high-speed fixed-wing uav penetration strategy based on deep reinforcement learning," *Aerospace Science and Technology*, vol. 148, p. 109089, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963824002220

[24] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000437029800023X

[25] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[26] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, p. 620–626, 2018.

[27] PX4 Developers, "Controller diagrams — px4 flight stack architecture," https://docs.px4.io/main/en/flight_stack/controller_diagrams.html, PX4, 2025, accessed: 2025-09-23.

[28] "Zed 2i stereo camera," Available: https://www.stereolabs.com/en-nl/store/products/zed-2i, Stereolabs Inc., 2023, accessed: Sep. 15, 2025.

[29] L. E. Ortiz, V. E. Cabrera, and L. M. G. Goncalves, "Depth data error modeling of the zed 3d vision sensor from stereolabs," *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, vol. 17, no. 1, pp. 1–15, Jun. 2018. [Online]. Available: https://elcvia.cvc.uab.cat/article/view/v17-n1-ortiz

[30] H. Sahabi and A. Basu, "Analysis of error in depth perception with vergence and spatially varying sensing," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 447–461, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S107731429690034X

[31] A. Abdelsalam, M. Mansour, J. Porras, and A. Happonen, "Depth accuracy analysis of the zed 2i stereo camera in an indoor environment," *Robotics and Autonomous Systems*, vol. 179, p. 104753, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889024001374

[32] F. JONSSON, "On the heavy-tailedness of student's t-statistic," *Bernoulli*, vol. 17, no. 1, pp. 276–289, 2011. [Online]. Available: http://www.jstor.org/stable/20839243

[33] I. Reid, "Estimation ii: Discrete-time kalman filter," https://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf, n.d., accessed: 2025-09-15.

[34] R. Harman and V. Lacko, "Generators of uniformly distributed points in n-dimensional spheres and on their surfaces," *Journal of Multivariate Analysis*, vol. 101, no. 10, pp. 2297–2304, 2010.

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

[36] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, pp. 1–8, 2021. [Online]. Available: https://jmlr.org/papers/volume22/20-1364/20-1364.pdf

[37] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach, Global Edition*. Pearson Deutschland, 2016. [Online]. Available: https://elibrary.pearson.de/book/99.150005/9781292153971

[38] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[39] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[40] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: a llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/2833157.2833162

[41] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7512–7519.

[42] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," 2020. [Online]. Available: https://arxiv.org/abs/2003.04960