Independent project (degree project), 15 credits, for the Degree of Bachelor of Science in Engineering in Computer Science and Engineering, specialization in Internet of Things

Spring Term 2026

Faculty of Natural Sciences

Autonomous Drone Navigation in Forest Environment Using Proximal Policy Optimization Reinforcement Learning

Elina Rosato and Danielis Maizelis

May 11, 2026

Authors
Danielis Maizelis, Elina Rosato

Title
Autonomous Drone Navigation in Forest Environment Using Proximal Policy Optimization Reinforcement Learning

Supervisor
Ali Hassan Sodhro

Examiner
Åke Arvidsson

Authorship Statement

We hereby confirm that this thesis is our own original work. Both authors share equal responsibility for all aspects of the project including literature review, simulation setup, PPO implementation, hardware integration, real-world testing, data analysis, and thesis writing. AI tools were utilised solely for grammar checking and language refinement, without contributing to the research, analysis, or conceptual development of the content.

Word Count

Abstract	297
Introduction	1508
Related Work	1162
Design Methods	834
Experimental Setup	1820
Results	498
Analysis and Discussions	782
Conclusions	274
Future Work	578
Total	**7753**

Abstract
Autonomous drone navigation in forest environments faces significant challenges due to dense, unpredictable obstacles and the high cost of traditional sensing systems (Light Detection and Ranging (LiDAR): $1,000–10,000+; stereoscopic cameras: $300–800), limiting scalable deployment for applications such as bark beetle monitoring. This thesis investigates whether a PPO-based reinforcement learning agent, trained entirely in simulation using only a single low-cost RGB camera with software depth estimation, can perform effective obstacle avoidance when deployed on a real drone in a forest environment.

The system uses an asymmetric actor–critic PPO architecture in which the actor receives only a stacked depth image produced by Depth Anything V2 from a single Arducam IMX219 camera, while the critic has access to privileged simulator state during training. The policy was trained in Cosys-AirSim with extensive domain randomisation and deployed without fine-tuning on a Holybro S500 V2 quadrotor equipped with an NVIDIA Jetson Orin Nano. Evaluation covered two tree density configurations (sparse: 2 trees, dense: 3 trees) in a 15 m corridor across 100 simulation and 100 real-world episodes.

In simulation, the policy achieved an episode-level success rate of 94.00% (sparse) and 84.00% (dense), with a combined rate of 89.11% and a per-obstacle success rate of 95.60%. In real-world trials, episode-level success was 74.00% (sparse) and 70.00% (dense), yielding an overall sim-to-real transfer efficiency of 81.11% at the episode level and 84.02% per obstacle.

The results confirm that a single $20 RGB camera, combined with a software depth model and a domain-randomised PPO policy, is sufficient for autonomous forest obstacle avoidance at a transfer efficiency competitive with sensor-heavy alternatives. The findings provide a reproducible benchmark for low-cost sim-to-real drone navigation in unstructured environments.

Keywords
Proximal Policy Optimization, Autonomous Drone Navigation, Forest Environment Simulation, Reinforcement Learning, Sim-to-Real Transfer, Domain Randomization

Acknowledgements
We would like to express our sincere gratitude to our supervisor, Ali Hassan Sodhro, for providing invaluable guidance, constructive feedback, and continuous support throughout this thesis. Their expertise in reinforcement learning and autonomous systems significantly shaped the direction and quality of this work.

We are grateful to Åke Arvidsson for his thorough review and insightful suggestions that helped strengthen the thesis structure and experimental design.

We would like to thank Kristianstad University and the Department of Computer Science for providing access to computational resources and research facilities that made this thesis possible. Special thanks to the technical staff for their assistance with hardware setup and testing equipment.

We acknowledge the open-source communities behind PyTorch, Cosys-Airsim, and PX4 Autopilot and OpenAI Gymnasium, whose tools and documentation formed the technical foundation of our implementation.

Finally, we thank our families and friends for their patience, encouragement, and understanding during the intensive periods of this thesis. We would like to give a special mention to Erik Karlsson, whose invaluable insights on 3D printing were crucial to the hardware development of this project.

Abbreviations

PPO	Proximal Policy Optimization
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
RGB	Red-Green-Blue
LiDAR	Light Detection and Ranging
CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
LSTM	Long Short-Term Memory
CPN	Collision Prediction Network
MAVLink	Micro Air Vehicle Link
MAVROS	MAVLink-to-ROS Bridge
UAV	Unmanned Aerial Vehicle
GPS	Global Positioning System
IMU	Inertial Measurement Unit
ROS	Robot Operating System
ROS2	Robot Operating System 2
FOV	Field of View
GAE	Generalized Advantage Estimation
DQN	Deep Q-Network
DDPG	Deep Deterministic Policy Gradient
TD3	Twin Delayed DDPG
SAC	Soft Actor-Critic
TensorRT	NVIDIA TensorRT (deep learning inference optimiser)
PX4	PX4 Autopilot firmware
MAVSDK	MAVLink Software Development Kit

Table of Contents

1. Introduction

1.1. Background

Unmanned aerial vehicles (UAVs) have become a practical tool for tasks that are dangerous, repetitive, or too large in scale for humans to carry out efficiently. Beyond military and delivery applications, drones are increasingly used for environmental monitoring, search and rescue, infrastructure inspection, and precision agriculture [1], [2]. What makes this possible is the combination of increasingly capable onboard computers, low-cost sensors, and advances in autonomous navigation software.

Despite this progress, most commercial drone deployments still require a human pilot. Operating autonomously inside a forest, where the environment is unstructured and obstacles appear unpredictably, remains an open research problem [1], [3], [4].

Reinforcement learning (RL) has emerged as one of the most promising approaches to this problem. Rather than programming explicit rules for every possible scenario, an RL agent learns a navigation policy through trial and error in a simulated environment, accumulating experience across thousands of episodes [2], [5]. Proximal Policy Optimization (PPO) [5] is among the most widely used RL algorithms for drone control, valued for its training stability and its ability to handle continuous, high-dimensional observation spaces.

A key practical challenge is bridging the gap between simulation and the real world. A policy trained entirely in simulation will encounter visual and physical conditions in deployment that differ from what it was trained on. Domain randomisation [6], [7] is the standard strategy for closing this gap: by varying lighting, obstacles, sensor noise, and other environmental parameters during training, the agent is forced to learn a policy that generalises rather than memorises.

A second and equally important challenge is cost. Most autonomous navigation systems rely on LiDAR sensors ($1,000–$10,000) or stereo cameras ($300–$800) [1], [8]. These costs make fleet-scale deployment impractical and limit autonomous drones to well-funded research or industrial settings. This thesis takes a different position: if a single standard RGB camera, costing under $80, can provide sufficient perceptual information for obstacle avoidance when paired with a learned policy and a software depth model, then the barrier to deploying autonomous drones at scale collapses. Smaller, lighter, and cheaper drones become viable. The same platform can be replicated and deployed in large numbers without sensor budgets that dwarf the cost of the airframe itself. This democratisation of autonomous navigation is a central motivation of this work.

1.2. Motivation

Drone-based surveillance and monitoring is increasingly relevant across a wide range of high-impact applications. UAVs are being deployed for wildfire early detection, where aerial coverage over large terrain is critical and ground access is dangerous. They are used in avalanche rescue operations to locate survivors in areas that are inaccessible on foot. They assist in power-line and pipeline inspection, flood mapping, and coastal erosion monitoring [1], [2]. What these applications share is a need for autonomous flight in environments that are unstructured, dynamic, and difficult to map in advance.

Forest environments present a particularly challenging case. The obstacles are irregular and densely packed, the lighting changes constantly, and the visual appearance varies significantly with species, season, and weather. Yet forest monitoring is an area of growing urgency. One specific example is the spread of bark beetle infestations across European forests. The European spruce bark beetle (*Ips typographus*) has been responsible for the loss of over 150 million cubic metres of timber in recent decades [9], [10], with infestations accelerating across Sweden and Central Europe. Early detection is critical: the green-attack stage, when intervention is still effective, is invisible to the naked eye and requires repeated aerial surveys at weekly intervals [10], [11]. UAVs equipped with multispectral sensors can detect infestations with accuracies of 88–96% [9], [10], [12], but the required flight frequency makes manual piloting impractical at scale.

This thesis originated from a collaboration with BeetleSense, a forest monitoring company whose co-founder Christo Van Zyl proposed investigating autonomous drone navigation in forest environments. The bark beetle use case provided a concrete starting point, but the broader question is whether a drone can navigate a forest using only a standard, low-cost RGB camera with no LiDAR or stereo rig. If so, autonomous navigation stops being gated by hardware cost, making it possible to deploy fleets of small, cheap drones where today only expensive, sensor-heavy platforms can operate.

As we began reviewing the research landscape, it became clear that reinforcement learning was the most appropriate framework for this problem. Forest navigation requires making accurate, real-time decisions in an environment that is unpredictable and impossible to fully specify in advance. Classical rule-based approaches cannot generalise to the variability of a real forest. Supervised learning requires labelled trajectory data that does not exist at scale. RL, by contrast, allows the agent to develop navigation behaviour through experience, adapting to obstacle configurations it has never seen before [2], [5]. This combination of requirements and literature evidence pointed clearly toward a PPO-based RL approach trained in simulation and transferred to a real platform.

1.3. Objectives

The objectives of this thesis are:

- To investigate whether a single low-cost RGB camera, combined with a software depth estimation model, provides sufficient perceptual information for autonomous obstacle avoidance in a forest environment.
- To design and implement a PPO-based reinforcement learning agent capable of navigating through a simulated forest corridor and avoiding tree trunk obstacles.
- To study sim-to-real transfer methods and evaluate how well a policy trained entirely in simulation transfers to a physical drone without fine-tuning.
- To gain practical experience with real drone hardware, from flight controller integration and onboard compute to real-world testing in a forest environment.
- To quantitatively evaluate the system's performance across sparse and dense obstacle configurations in both simulation and real-world trials.

1.4. Problem Statement

Recent advances in deep RL, and in PPO in particular [5], have demonstrated strong results for vision-based drone navigation in indoor and structured environments [13], [14], [15], [16]. Monocular RGB cameras have been shown to provide sufficient visual information for obstacle avoidance without depth sensors [16], [17], [18]. Sim-to-real transfer via domain randomisation has been validated for drone control tasks [6], [7], [19], [20]. However, to the best of the authors' knowledge, no existing work combines all of these elements in a forest-specific context: a PPO agent, a single low-cost RGB camera with software depth estimation, training in a forest simulator, and deployment on a real quadrotor. This leads to the following research question:

> **How effectively can a PPO-based reinforcement learning agent, trained entirely in simulation using only a single RGB camera with software depth estimation, perform autonomous obstacle avoidance when deployed on a real drone in a forest environment, across sparse and dense tree configurations?**

1.5. Contributions

The main contributions of this thesis are:

- **A PPO-CNN navigation policy for forest obstacle avoidance.** We design, train, and evaluate a dual-stream PPO agent that processes depth-converted camera images through a CNN alongside a scalar state vector, supported by a reward function combining dense progress shaping, time penalties, lateral offset penalties, and asymmetric terminal signals that enables stable PPO convergence across a two-stage curriculum. The resulting

policy achieves over 89% episode-level success in simulation and 72% in real-world forest trials.

- **Depth estimation from a single RGB camera using an open-source model.** We demonstrate that replacing a hardware depth sensor with Depth Anything V2 [21], an open-source monocular depth estimation model, is a viable strategy for low-cost autonomous drone navigation. This finding lowers the hardware barrier for forest drone deployment significantly.
- **A quantitative sim-to-real evaluation in a real forest.** We conduct a controlled evaluation of 100 real-world flights across two density configurations, providing a sim-to-real transfer efficiency of 81% as a reproducible benchmark for future work in this area.

1.6. Scope and Limitations

1.6.1. Scope

This thesis focuses on the obstacle avoidance component of autonomous drone navigation in forest environments. The system is trained in simulation and evaluated in both simulation and real-world trials. The physical platform is the Holybro S500 V2 quadrotor equipped with an Arducam IMX219 RGB camera and an NVIDIA Jetson Orin Nano companion computer. Trials are conducted under controlled weather conditions: wind below 3 m/s, daylight, and no precipitation. The thesis does not address bark beetle detection, path planning, or multi-drone coordination; only the navigation capability that makes autonomous forest flight possible.

1.6.2. Limitations

The evaluation environment uses naked tree trunks with diameters of 0.3–0.8 m and heights of at least 3–5 m, without branches. These conditions are more controlled than a real operational forest, and results may not generalise directly to environments with branching trees, ground vegetation, mixed species, or more complex obstacle geometry [1], [3], [4].

The study evaluates two density configurations (sparse and dense) across a fixed 15 m corridor. Performance across longer corridors, higher densities, or variable flight altitudes is not assessed. Sim-to-real transfer is evaluated on a single physical platform, limiting the generalisability of the transfer efficiency findings [7], [19].

1.7. Thesis Structure

This thesis is organised as follows. Section 1 introduces the background, motivation, objectives, problem statement, contributions, and scope. Section 2 surveys the related work. Section 3 describes the design methods, covering the rationale for the chosen algorithm stack and reward design. Section 4 details the experimental setup, covering the simulation environment, PPO training pipeline, hardware platform, and evaluation protocol. Section 5 presents the

experimental results. Section 6 analyses and discusses the findings in relation to the research question. Section 7 states the conclusions, and Section 8 outlines directions for future work.

2. Related Work

A structured search across Google Scholar, IEEE Xplore, and Semantic Scholar over five topics – bark-beetle detection, obstacle avoidance, drone sensors, reinforcement learning for UAVs, and sim-to-real transfer – limited to 2019–2025, narrowed about 450 results to 25 after duplicate, title, and abstract screening; three foundational papers [5], [8], [22] were added by hand, giving 28 references. Studies are grouped into classical single-camera navigation, deep reinforcement learning (DRL) with vision in simulation, sim-to-real with real flights, and sim-to-real methodology.

The thesis goal is a low-cost autonomous drone flying through a forest with one RGB camera (62° FOV), no LiDAR or stereo, depth recovered in software with Depth Anything. The platform is a full-size Holybro S500 V2, the algorithm is Proximal Policy Optimization (PPO), and training is in AirSim (indoor + outdoor scenes) before real tests in a Swedish forest.

2.1. Overview and Research Gap

Table 1 compares the closest studies along sensor, algorithm, test environment, and whether the drone actually flew.

Table 1. Comparison of closely related studies along the axes used in this thesis.

Study	Sensor	Algorithm	Test environment	Real flight
Kumar et al. [17]	Single RGB camera	Classical CV	Indoor	Yes
Kabas [16]	RGB or depth (AirSim)	PPO	AirSim corridor with openings	No
Loquercio et al. [6]	Monocular RGB (300×200)	CNN + imitation + DR	Racing track	Yes
Jang et al. [20]	Mixed	DRL + sim-to-real methods	Mixed	Partial
Del Col et al. [23]	RealSense D435i + T265	Depth autoencoder + LSTM CPN	Boreal Finnish forest	Yes
This thesis	**Single RGB, 62°, + Depth Anything**	**PPO**	**AirSim + real Swedish forest**	**Yes**

Three patterns stand out: deep-RL avoidance is usually only tested in simulation [16]; real-world studies use either racing drones on clean tracks [6] or platforms with stereo and tracking cameras [23]; and richer sensor stacks work, but LiDAR ($1,000–$10,000) and stereo ($300–$800) are too expensive, heavy, and fiddly to calibrate for routine forest use [1], [8]. No earlier study combines PPO, a single narrow-FOV RGB camera with software depth from Depth Anything, a full-size quadrotor, AirSim training, and real forest flights – the gap this thesis closes to remove the cost barrier to routine bark-beetle monitoring.

2.2. Classical Single-Camera Navigation

Software depth from a single RGB camera is now possible thanks to recent monocular depth models. Ranftl et al. [24] showed with MiDaS that training across many depth datasets with a scale-invariant loss generalises to unseen scenes. Yang et al. [21] pushed this further with Depth Anything V2, trained on synthetic depth labels and pseudo-labelled real images, giving sharper depth and running 10× faster than diffusion-based alternatives – making software-only depth a real alternative to hardware sensors when cost is tight.

Kumar et al. [17] – single-camera obstacle avoidance on a small onboard computer. Kumar et al. built a navigation pipeline using image-processing rules on a small onboard computer with a single RGB camera and MiDaS for depth – direct support for the idea that one cheap camera can be enough. Weaknesses are hand-written rules rather than a learned policy and tidy indoor-only tests. This thesis replaces the rule-based pipeline with a CNN policy trained by PPO that learns directly from camera images, swaps MiDaS for Depth Anything V2, and tests outdoors against real tree trunks.

2.3. DRL with Vision in Simulation

Kabas [16] – PPO with a single RGB or depth camera in AirSim corridors. Kabas trained a PPO agent end-to-end in an Unreal Engine + AirSim corridor with round openings. RGB alone reached 96 % success using a stack of grayscale frames through a Convolutional Neural Network (CNN) – direct evidence that PPO + single camera suffices for non-trivial avoidance. Weaknesses are the regular corridor, minimal domain randomisation, and the policy never running outside simulation. This thesis swaps the corridor for irregular outdoor trees, borrows the software-depth idea from [17], adds heavier domain randomisation in AirSim, keeps a CNN over a frame stack, and closes the loop with real flights.

2.4. Sim-to-Real Studies with Real Flights

Loquercio et al. [6] – Deep Drone Racing. A CNN reads a single 300×200 RGB image and outputs a local target direction and speed; a classical planner and controller produce motor commands. The network is trained by imitation learning on simulator-optimal trajectories with the scene randomised across background, lighting, gate shape, and gate texture. Versus a no-randomisation baseline, performance rose by **300 %** in simulation and **36 %** in the real world, showing zero-shot sim-to-real transfer for fast camera-based flight. An ablation ranks background randomisation essential, lighting second, gate-shape smallest – evidence that domain randomisation alone can bridge sim and reality with one camera.

Weaknesses for forest use are a tidy track with bright gates, a small racer, and imitation of a pre-computed expert – a recipe the authors say fails without a defined expert path. This thesis keeps the single-camera, randomised, sim-to-real philosophy but uses PPO instead of imitation learning, since a forest has no gates and no optimal path. Following the ablation, AirSim training randomises three forest analogues: **trunk shape and angle** (gate shape), **time of day** (lighting), and **forest layout with randomised tree positions** (background – the most important factor). The track becomes an unstructured Swedish forest, the racer a Holybro S500 V2, and high-speed racing a safe forest cruise.

Del Col et al. [23] – autonomous drone navigation in Finnish boreal forests. Del Col et al. (UAV-g 2025, Espoo) carry an Intel RealSense D435i stereo depth camera and

an Intel RealSense T265 tracking camera, with onboard computation on an NVIDIA Orin NX and a Holybro Pixhawk 6C Mini. The action space is **discrete**: constant forward speed × 8 vertical-speed × 32 yaw-rate ($8 \times 32 = 256$ trajectories) scored each step. The Collision Prediction Network is trained by *supervised* learning on simulator-generated collision labels (RotorS and Aerial Gym), not RL, and tested in real Finnish forests with autonomous flights around 60 m in cluttered scenes – the most directly comparable forest study. Weaknesses are the cost, weight, and calibration of the stereo and tracking cameras ($300–$800 [1], [8]).

The hardware/middleware base matches this thesis: the same Holybro/PX4 family (Pixhawk 6C vs. 6C Mini), the same NVIDIA Jetson Orin family (Orin Nano vs. Orin NX), and ROS as middleware. This thesis also uses discrete actions, but lets PPO learn which to pick directly from the camera image rather than scoring a fixed motion-primitive grid. The sensor side is deliberately cheaper and lighter: one monocular RGB camera with Depth Anything in software replaces the RealSense D435i + T265 pair.

2.5. Sim-to-Real Methodology Studies

Zhao et al. [7] and Salvato et al. [19] – sim-to-real surveys. Two complementary surveys of sim-to-real transfer in deep RL review the standard toolkit – domain randomisation, dynamics randomisation, system identification, and observation augmentation – and single out sensor-noise injection (Gaussian image noise, brightness/contrast jitter, sensor delay, motor-command noise) as one of the cheapest and most reliable ways to harden a policy; neither offers a recipe specific to outdoor UAV flight. This thesis applies the conclusion directly: the sensor pipeline injects per-frame Gaussian image noise, brightness/contrast jitter, motion blur, randomised inference delay, and small action noise during training, tuned for the Depth Anything inference profile on the onboard Jetson.

Jang et al. [20] – deep-RL policies from simulation to reality. Jang et al. treat sim-to-real systematically across robotic platforms, concluding that careful domain randomisation plus manageable real-world tuning usually closes the gap. The case studies are not on outdoor forest flight – which this thesis re-checks on a full-size drone with a single RGB camera.

No prior work combines PPO, a single narrow-FOV RGB camera with software depth from Depth Anything, asymmetric actor–critic training, and validation in a real forest. The methods chapter describes that combination, and the next subsection sets out the rationale behind each design choice.

3. Design Methods

The method described in this chapter is grounded in five lines of prior work: PPO as the learning algorithm [5], asymmetric actor–critic as the training scheme [22], monocular RGB perception with software depth from Depth Anything V2 [21], the AirSim corridor + frame-stack + CNN recipe of Kabas [16], the domain-randomisation strategy of Loquercio et al. [6], and the discrete-action plus Holybro/PX4 + ROS deployment design of Del Col et al. [23]. The remainder of this section walks through how each of these is used and why.

The starting point is a practical goal: enabling drones to fly forest-monitoring missions, such as bark-beetle surveys, in a way that is repeatable, reliable, and cheap. The cost requirement directly drives the design. The drone carries only one normal RGB camera – the main ingredient of this method – so that the same hardware can be deployed at fleet scale without a LiDAR or stereo budget.

That choice immediately creates a problem. Writing explicit instructions for a drone to fly through unforgiving forest terrain is not realistic: the geometry is irregular, the lighting changes constantly, and no clean hand-written rule maps "what the camera sees" to "what to do" [16]. Reinforcement learning is the natural way out. Kabas [16] showed that a PPO agent trained on a stack of monocular RGB frames in AirSim, processed through a CNN, can avoid obstacles at a success rate comparable to setups that rely on much more expensive depth sensors. The simulated drone in this thesis adopts that same framework: PPO, a stack of grayscale frames, and a CNN.

On top of that, this thesis adds an explicit depth signal. Depth Anything V2 [21] produces a precise, low-latency depth map from a single RGB image; trained on millions of labelled and pseudo-labelled scenes, it generalises to almost any object at almost any distance.

Even so, a model that is robust on its training distribution is not automatically robust on a real drone in a real forest. To close the gap between simulation and reality, this thesis applies domain randomisation in the same spirit as the deep drone-racing work of Loquercio et al. [6]: tree shapes, tree angles, and tree positions are all randomised between training episodes, and the time of day is varied to change the lighting. Training itself is done in a long corridor scene in which the drone is respawned at the start whenever it collides with anything [6], [16], allowing thousands of attempts per session.

For the action side, this thesis follows the design philosophy of Del Col et al. [23]: the policy chooses among discrete actions rather than emitting continuous commands. Keeping the decision space small and clean makes the policy easier to train and easier to reason about. The same paper also informs the on-drone hardware and middleware choices used here: the Holybro/PX4 family of flight controllers, the NVIDIA Jetson Orin family for onboard computation, and ROS as middleware – even though the perception and learning stack built on top of that

base is entirely different.

The reward function itself follows the dense-plus-sparse split that is standard in PPO-based UAV navigation [5], [16]. A dense forward-progress term gives the agent a continuous gradient from the first episode [16], a small per-step time penalty discourages hovering or stalling [16], and a lateral-offset penalty paired with a small centerline bonus keeps the policy on the spawn-to-goal axis – the same trajectory-tracking idea used by Loquercio et al. for drone racing [6]. Terminal signals are asymmetric: a goal bonus larger than the collision penalty plus an efficiency bonus for fast completion [16], with the explicit collision cost mirroring the central role collision probability plays in the Collision Prediction Network of Del Col et al. [23]. All rewards are normalised online with VecNormalize so the signal stays stable across curriculum stages.

The network that consumes those signals is built around an asymmetric actor–critic [22]: the actor sees only what will be available on the real drone – the stacked grayscale frames and a small scalar state vector – while the critic, active only during training, additionally receives privileged simulator-only information, narrowing the sim-to-real gap by construction. The actor is a dual-stream network: a CNN over the 8-frame stack, validated for PPO from monocular RGB in AirSim [16] and for sim-to-real drone racing on a forward-facing RGB camera [6], plus a small MLP over the scalar state. The critic head consists of two 128-unit fully connected layers, larger than the policy head, following standard PPO practice [5], [22].

Put together, the method built from these decisions is a single repeatable pipeline. A low-cost drone with one forward-facing RGB camera feeds each frame into a software depth model and a stack of recent grayscale images. A PPO policy, trained from scratch in a domain-randomised simulator with respawn-on-collision corridor episodes, reads that combined input and chooses among a small set of discrete actions. The whole stack runs on a familiar Holybro/PX4 + Jetson Orin + ROS base, and the same evaluation protocol is used in simulation and in a real Swedish forest. The next chapter gives the concrete implementation values for each component.

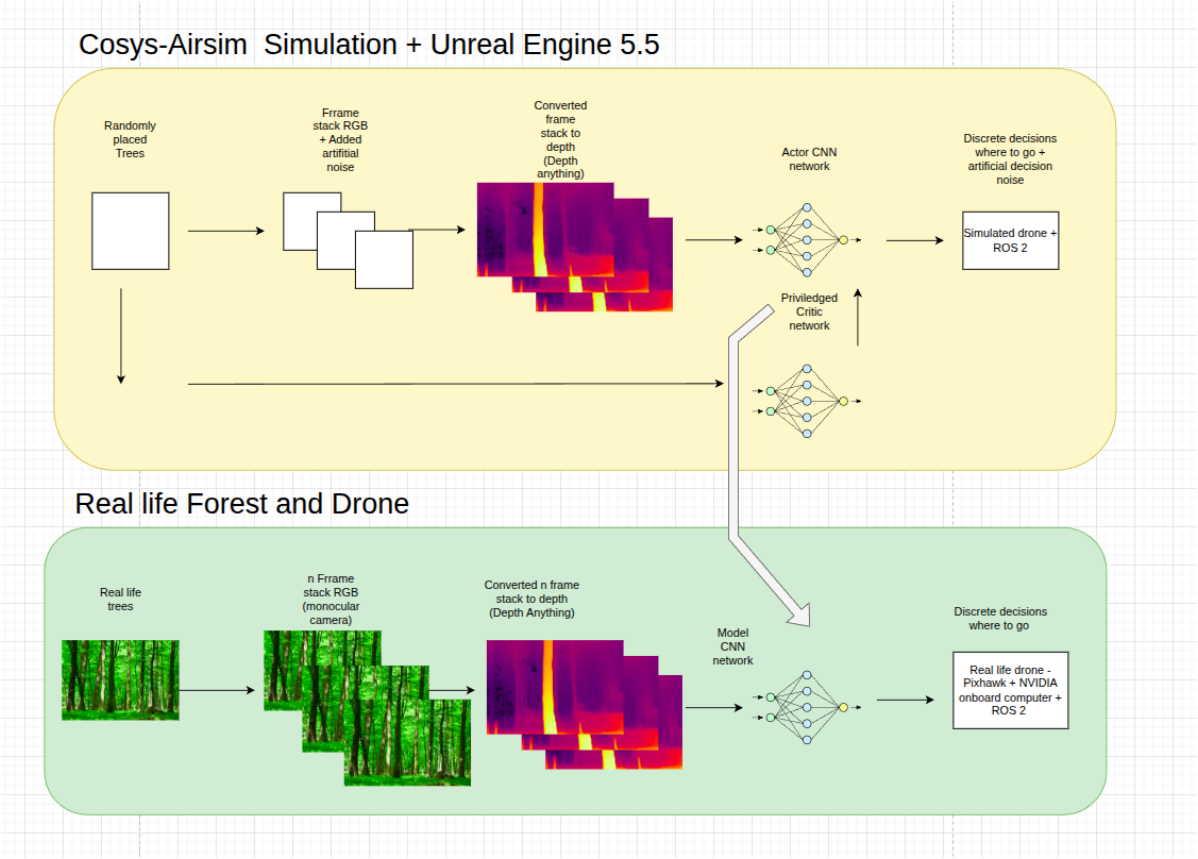


Figure 1. Simulation training setup in Cosys-Airsim with Unreal Engine 5.5

4. Experimental Setup

This chapter describes the concrete setup that turns the approach in the previous chapter into a working system. It covers the simulator, the corridor used for training and evaluation, the tree spawning, the camera and depth pipeline, the asymmetric actor–critic policy and action space, the simulation and real-world evaluation protocols, and the physical drone built to match the simulated platform.

4.1. Simulation Environment

All training and evaluation in simulation is run in *Cosys-AirSim*, a community fork of *Microsoft AirSim* [16], on top of *Unreal Engine 5.5*. Cosys-AirSim is used specifically because it exposes `simSetObjectPose` and `simSetObjectScale`, which allow tree positions, angles, and sizes to be changed between episodes without reloading the scene – this is what makes the heavy domain randomisation in this work practical.

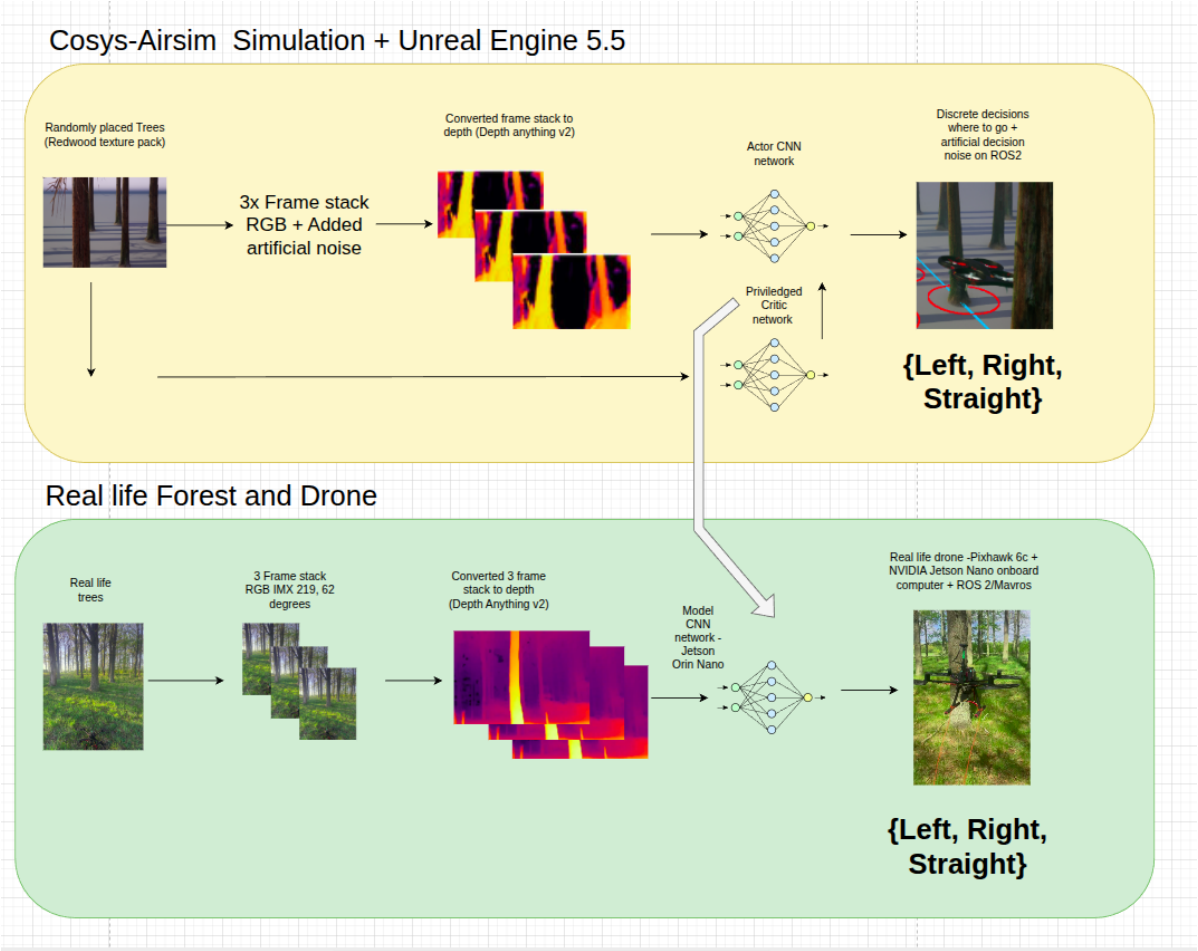


Figure 2. Simulation training setup in Cosys-Airsim with Unreal Engine 5.5

The simulated forest is a **15 m long, 4 m wide corridor** placed inside an Unreal scene built from a redwood texture pack. Since this thesis evaluates on naked tree trunks only, the specific tree species or texture is not central to the method; the texture pack provides a visually consistent set of trunks while still varying in shape and bark detail. Trees inside the corridor act as obstacles and trees placed outside the corridor act as visual background.

4.2. Tree Spawning and Difficulty Stages

The training curriculum has **two difficulty stages**, both inside the same 15 m corridor:

- **Sparse** – 2 trees inside the corridor, placed at random positions.
- **Dense** – 3 trees inside the corridor, placed at random positions.

Across both stages, the in-corridor trees and the surrounding background trees are spawned in a random order, with random angles (tilted up to roughly $\pm 30°$–$45°$ off vertical) and random trunk thickness drawn within stage-specific bounds. Background trees outside the corridor are placed throughout the visible area so that domain randomisation does not just affect the obstacles themselves but also the visual context the camera sees [6].

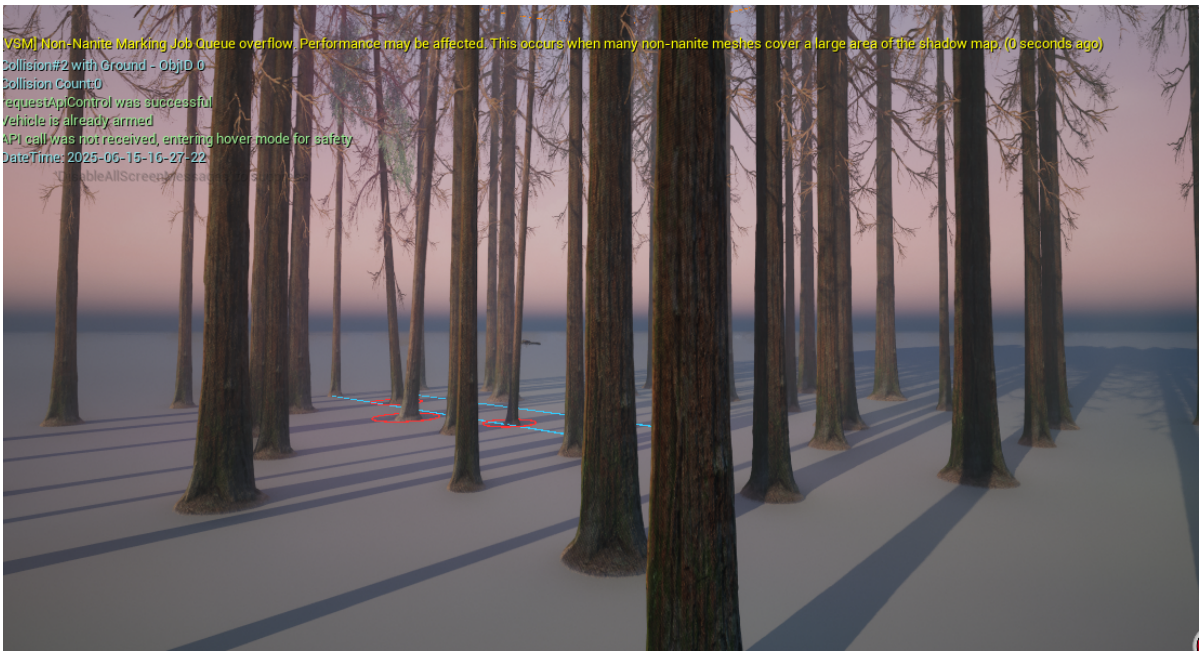


Figure 3. Simulation training setup in Cosys-Airsim with Unreal Engine 5.5

4.3. Camera, Frame Stack, and Depth Pipeline

The simulated camera matches the physical sensor exactly. It is a single forward-facing camera with a 62° field of view, the same FOV as the Arducam camera mounted on the real drone, so there is no FOV mismatch between simulation and reality [6], [7]. Each captured frame is resized to 126×224 px and pushed through Depth Anything V2 [21], which produces a depth image at the same resolution.

The policy does not see a single frame. It sees a **3-frame stack of depth images**, with frames sampled **5 control steps apart**; at the $\sim$10 Hz control rate of the agent, this means each new depth frame is roughly 0.5 s and $\sim$ 0.5 m of forward motion older than the next, so the 3-frame stack covers about a metre of recent motion. This gives the network implicit motion information without any optical-flow computation [16].

To make the policy robust to the differences between simulation and reality, several types of randomisation and noise are applied to this pipeline at training time:

- **Image noise.** Per-frame Gaussian noise with standard deviation drawn uniformly from [0.01, 0.03], plus per-episode brightness and contrast jitter [6], [7].
- **Motion blur.** Directional smear scaled by the recent lateral action, replicating the motion blur of a real camera during sharp avoidance manoeuvres.
- **Camera-mounting jitter.** The simulated camera is offset by a small random rotation per episode, replicating imperfect mounting on the real airframe.
- **Spawn yaw jitter.** The drone does not always start perfectly aligned with the corridor axis.
- **Velocity drift.** A small random drift (up to 0.3 m/s) is added to baseline velocities.

- **Inference / pipeline delay.** The action chosen at step t is not executed until step $t+\Delta$, with Δ randomly fixed to 1 or 2 control steps ($\sim$100–200 ms) per episode, replicating real-world Depth Anything inference latency on the onboard Jetson.
- **Action noise.** Small noise is also injected into the discrete-action selection during training to keep the policy from over-fitting to perfect, instantaneous responses.

These noise sources are derived from the sim-to-real strategies in [6], [7], [19], with the specific configuration of pipeline delay and motion blur tailored to the Depth Anything inference profile observed on the target Jetson hardware.

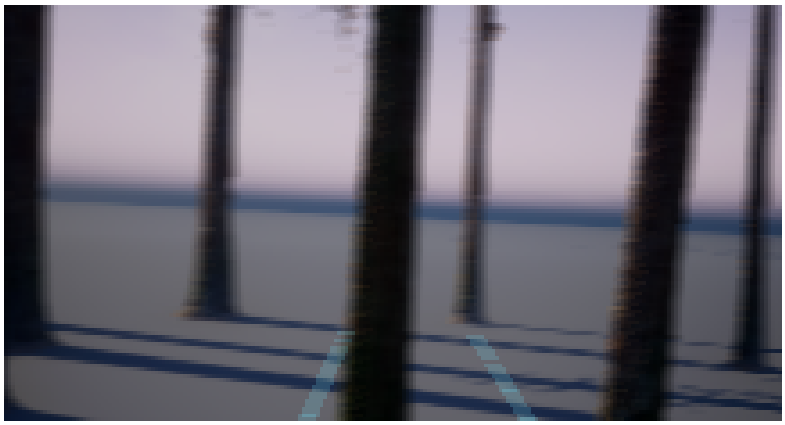


Figure 4. Image example in the simulation, together with the applied noise and 62 degree FOV

4.4. Reward, Corridor Constraint, and Episode Termination

The reward function is the dense-plus-sparse design described in Section 3 (forward-progress shaping, time penalty, centerline bonus, lateral penalty, terminal goal/collision/efficiency rewards [5], [6], [16], [23]). On top of that, the **corridor itself is a hard constraint**: the drone is allowed to move freely within the 15×4 m corridor, but any lateral excursion outside the corridor immediately ends the episode.

An episode therefore ends in one of three ways: *success* (the drone reaches the end of the corridor), *collision* (any contact with a tree), or *out-of-bounds* (the drone leaves the corridor sideways). All three trigger a respawn at the corridor entrance for the next episode. This respawn-on-failure / respawn-on-success structure is the same training pattern Kabas [16] used for AirSim corridor episodes and that Loquercio et al. [6] used during DAgger-style trajectory recovery.

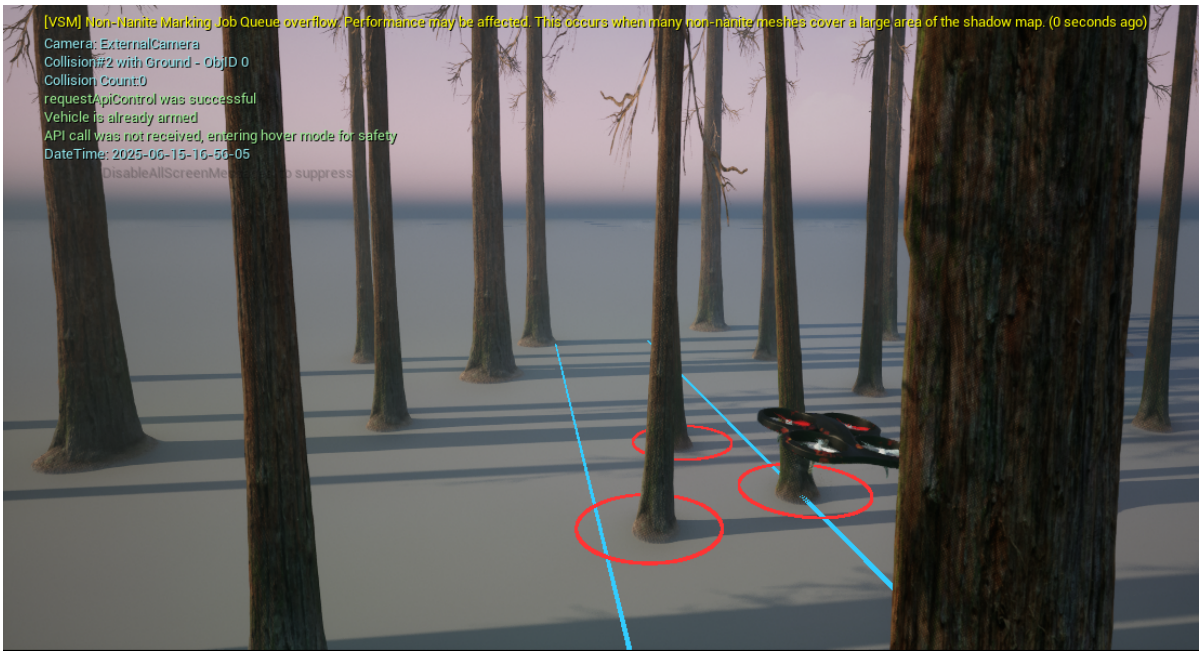


Figure 5. The 15x4m corridor labelled in blue, termination happens when the corridor end is crossed, corridor sides, or trees behaving as obstacles

4.5. Asymmetric Actor–Critic and Action Space

The policy is trained with PPO [5] in an asymmetric actor–critic configuration [22]. The two networks see different things on purpose:

- **Actor (the student).** Receives only the 3-frame depth stack from the camera. It runs a CNN over those depth images and outputs a categorical distribution over 3 discrete actions. The actor never sees tree positions, drone pose, or any other privileged simulator state.
- **Critic (the teacher).** Receives a 66-dimensional privileged vector containing the drone's position and velocity, the goal location, and the position and scale of up to 20 nearest trees. It does not see any image. Its only job is to provide a high-quality value estimate that the actor can learn from.

Because the actor only ever sees what will be available on the real drone (a depth-converted camera image), the policy never becomes dependent on signals that vanish at deployment. Because the critic exploits the full simulator state, it provides a tighter value baseline than an image-only critic could, which is the main argument of the asymmetric setup in Pinto et al. [22].

The actor's discrete action space is {Left, Straight, Right}. These map to a velocity command:

- **Straight** – forward 1.0 m/s, lateral 0 m/s. Cruise.
- **Left / Right** – forward 0 m/s, lateral ± 2.0 m/s. Hard avoidance step.

The forward-only 1 m/s cruise is intentionally conservative for forest flight, while the ±2 m/s lateral commit is sharp enough to clear a trunk before the drone passes it. The control loop runs at approximately 10 Hz effective decision rate (the 30 Hz physics simulation is queried with a frame-skip of 3). This rate is what makes the ∼100–200 ms randomised pipeline delay above realistic.

Yaw is locked to the direction of travel so the camera always faces forward, and altitude is held by an external P-controller, so altitude is not part of the RL action space.

4.6. Real-World Hardware

The physical drone is a **Holybro S500 V2** quadrotor (500 mm wheelbase, ∼ 1.75 kg all-up weight), chosen because its airframe comfortably accommodates the NVIDIA Jetson Orin Nano companion computer used for inference and because its dimensions match the simulated drone in Cosys-AirSim. Compute is split between the Jetson, which runs Depth Anything V2 inference and the PPO policy, and a **Pixhawk 6C** flight controller (running PX4) which receives velocity commands from the Jetson and produces the actual motor outputs. The Jetson talks to the Pixhawk over MAVLink.

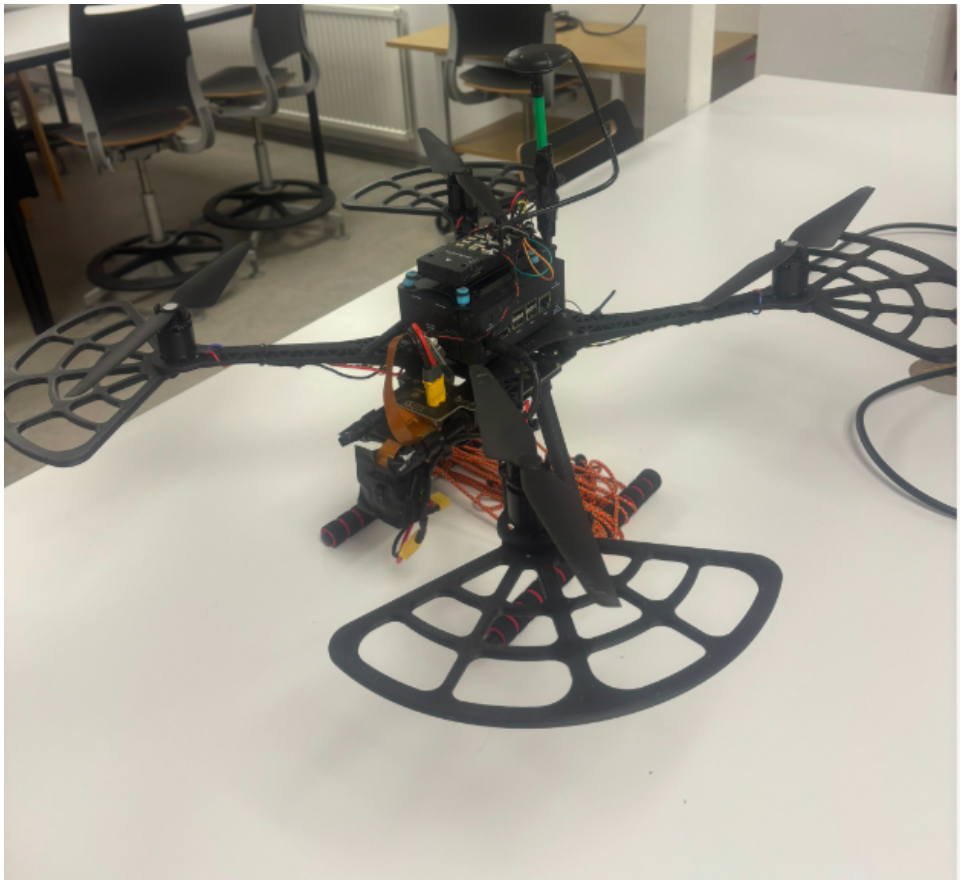


Figure 6. Physical drone, that includes the mentioned Jetson onboar computer, IMX219 camera, Pixhawk 6c flight controller

The camera is an Arducam IMX-series module configured to a 62° horizontal field of view, the same FOV as the simulated camera. At a unit price of around $20, it is the cheapest perception sensor on the platform by a wide margin compared with stereo ($300–800) or LiDAR ($1,000–10,000) alternatives [1], [8]. Table 2 lists every component.

Table 2. Physical drone hardware components.

Component	Specification
Frame	Holybro S500 V2 (500 mm wheelbase)
Flight controller	Pixhawk 6C, PX4 firmware v1.14+
Companion computer	NVIDIA Jetson Orin Nano (8 GB)
Camera	Arducam IMX219, 62° FOV, 128×128 @ 60 fps (tuned to 30 fps)
GPS	Holybro M10 GPS module
Telemetry	433 MHz SiK radio
Battery	4S 5000 mAh LiPo (5–10 min flight time)
Communication	MAVLink via UART

4.7. Middleware and Software

The same middleware stack is used in simulation and on the real drone, which keeps the codebase between the two environments as close to identical as possible. ROS 2 connects the Jetson to the rest of the system. The camera publishes frames on a ROS 2 topic, which an inference node consumes, runs through Depth Anything V2 and the PPO policy, and republishes as velocity commands. **MAVROS** bridges those ROS 2 commands to MAVLink and into the Pixhawk's velocity-control mode, and the same MAVROS bridge is used in simulation, where the Cosys-AirSim drone exposes a MAVLink-compatible interface. The result is that the inference graph that runs in simulation is the same graph that runs in the real forest – only the producer of the camera feed differs.

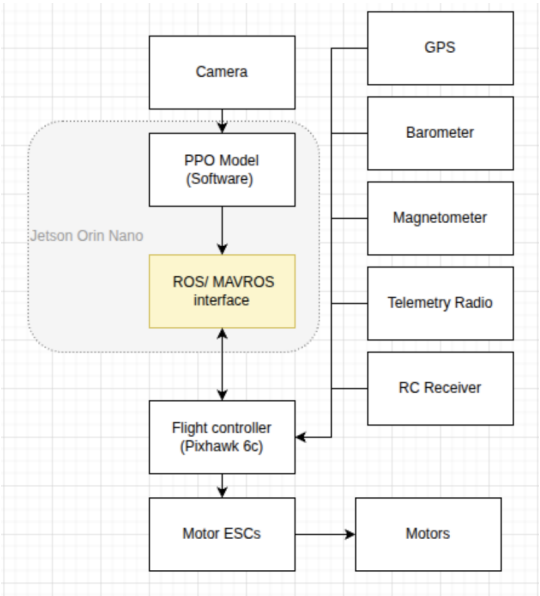


Figure 7. Middleware setup of real drone used in this paper

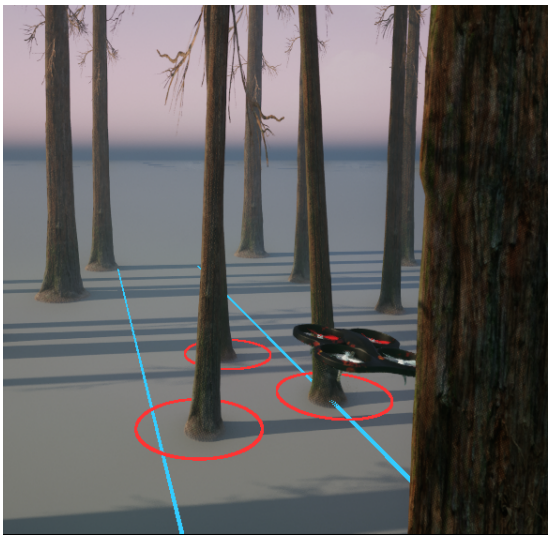
Table 3. Software stack for training and real-world deployment.

Component	Tool	Role
Simulation environment	Cosys-AirSim + UE 5.5	Physics, rendering, sensor simulation
RL framework	Stable-Baselines3 + PyTorch	PPO training
Image processing	OpenCV	Image capture
Camera interface	ROS2	Arducam frame streaming to Jetson
Sensor interface	MAVROS	Pixhawk IMU/GPS data to Jetson
Flight commands	MAVSDK-Python + MAVLink	Velocity commands to Pixhawk
Inference optimisation	TensorRT	Real-time policy inference on Jetson
Ground station	QGroundControl	Telemetry monitoring, safety override

4.8. Simulation and Real-World Evaluation

Simulation and real-world trials follow the same protocol so the two can be compared directly. In simulation, the policy is evaluated under fixed, deterministic conditions with noise and randomisation disabled and the curriculum fixed: **50 episodes in the sparse stage and 50 in the dense stage**, giving 100 simulation episodes.

The same policy is then deployed without fine-tuning on the physical drone, with **50 flights per stage** conducted in a Swedish forest of naked tree trunks; each flight uses a different physical corridor so no two episodes share the same tree configuration, and the two stages are matched to their simulation counterparts by average inter-trunk spacing measured before each session. Real flights run under controlled but realistic outdoor conditions (wind below 3 m/s, no precipitation, sufficient daylight). An episode counts as a success, in either environment, if the drone clears the corridor without tree contact, without leaving the corridor laterally, and – in the real world – without manual safety override.



(a) Simulated corridor (AirSim).

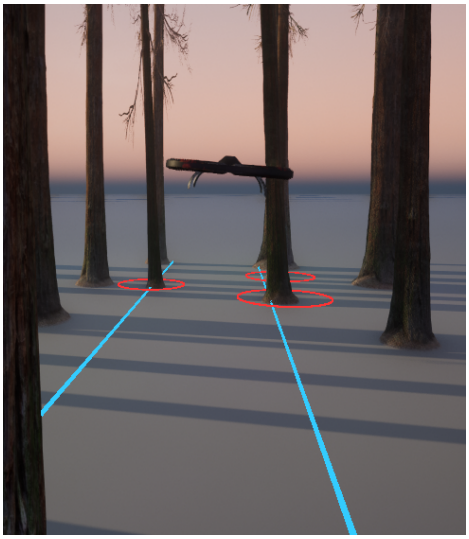
(b) Real Swedish forest corridor.

Figure 8. Simulated and real evaluation corridors, side by side.

(a) Simulated first-person view.

(b) Real first-person view with discrete-action arrow (left/straight/right) and the Depth Anything depth image.

Figure 9. First-person view, simulated and real, side by side.



(a) Simulated drone in AirSim.

(b) Real Holybro S500 V2 in flight.

Figure 10. Drone platform, simulated and real, side by side.

4.9. Variables

The same set of variables is logged in simulation and in real flights so the two can be compared directly. For every episode (sparse or dense, simulation or real forest), the following are recorded: the **tree density** (sparse / dense), the **trees in corridor** (2 or 3), the **trees avoided** (0–3), and the **episode result** (success or fail). The independent variables are *tree density* (sparse, dense) and *environment* (simulation, real forest). The dependent variables, derived from the per-episode log, are the avoidance rate (trees avoided over trees in corridor) and the success rate (successful episodes over total). Drone hardware, camera parameters, control frequency, weather, drone speed, and cruise altitude are treated as control variables and held constant across conditions. The sim-to-real transfer ratio is computed as the real-world success rate divided by the simulation success rate at the matching density. The full variable definitions, the hyperparameter table, and the per-episode result tables are listed in the appendix.

Table 4. Independent, dependent, and control variables. Logged identically in simulation and in the real-world flights.

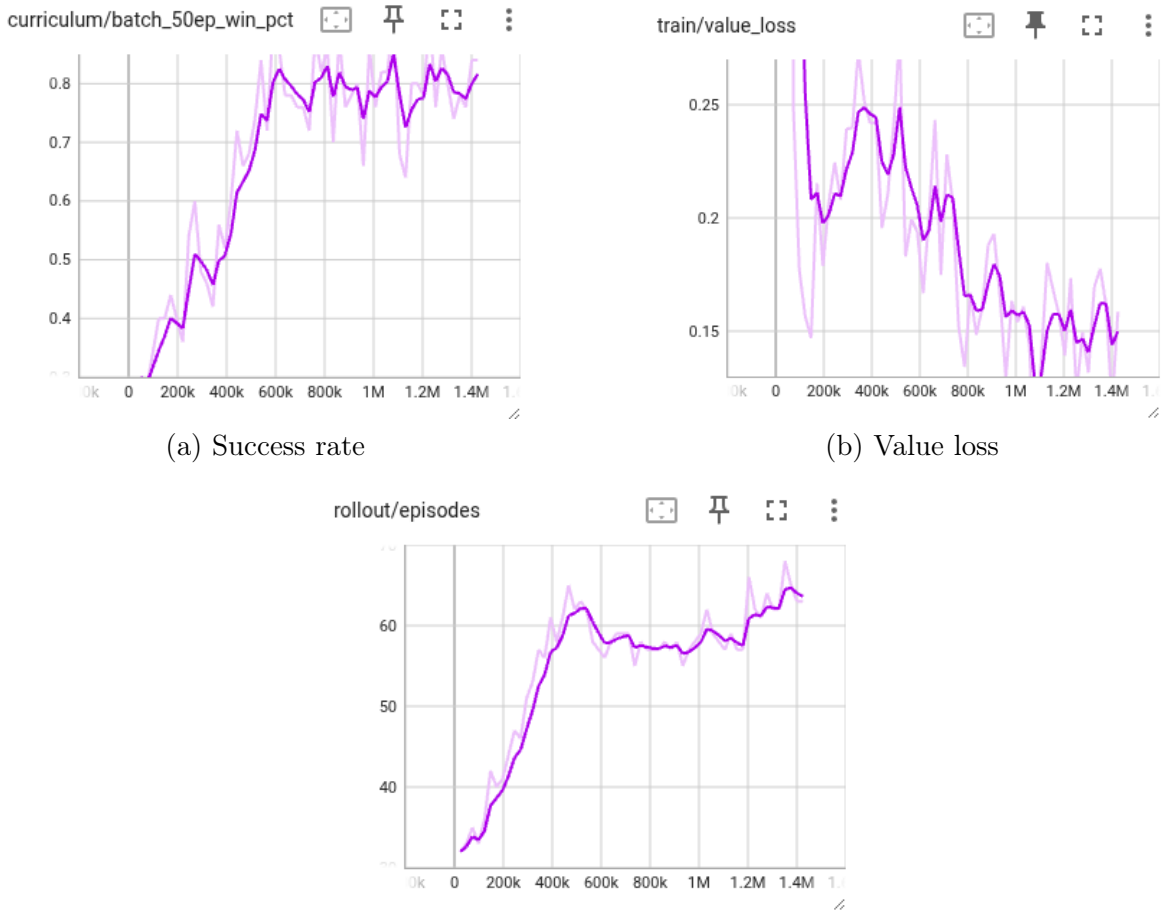
Variable type	Variable	Values / Range
Independent	Tree density	Sparse (2 trees) / Dense (3 trees)
Independent	Environment	Simulation / Real forest
Dependent	Trees avoided	0–3 (per episode)
Dependent	Episode result	Success / Fail
Dependent	Avoidance rate	Trees avoided / Trees in corridor × 100
Dependent	Success rate	Successful episodes / Total episodes × 100
Dependent	Sim-to-real ratio	Real success rate / Sim success rate (per density)
Control	Forward cruise speed	1.0 m/s
Control	Lateral avoidance speed	2.0 m/s
Control	Cruise altitude	1.5 m
Control	Camera FOV	62°
Control	Camera resolution	126×224 px
Control	Control frequency	∼10 Hz
Control	Weather conditions	Wind < 3 m/s, daylight, no precipitation

5. Results

This section presents the experimental results from both the simulation evaluation and the real-world flight trials. Results are reported at two levels: episode-level success rate (the drone completes the full 15 m corridor without any collision) and per-obstacle success rate (individual trees successfully avoided out of total trees encountered). All conditions were evaluated over 50 episodes per density configuration. The full episode-by-episode datasets for both simulation and real-world trials are available in Appendix A.

5.1. Training Stability

Figure 11 shows the PPO training curves recorded in TensorBoard over the full training run. The exponential success rate increase (a) and the value-loss downward curve (b) demonstrate the characteristic steady improvement and convergence behaviour of PPO [5], with no catastrophic policy collapse observed across the curriculum. Episode length (c) grows over the course of training, indicating that the drone simply survives longer in the corridor as it learns to avoid early collisions rather than crashing within the first few seconds.



(a) Success rate

(b) Value loss

(c) Episode length

Figure 11. PPO training curves from TensorBoard over the full training run: (a) success rate, reaching up to 92% in the dense stage; (b) value loss, showing steady convergence; (c) episode length, growing as the agent learns to survive longer in the corridor.

5.2. Simulation Results

The trained policy was evaluated over 100 episodes in simulation: 50 sparse (2 trees in corridor) and 50 dense (3 trees in corridor). Table 5 summarises the results. Figure 12 visualises the episode-level and per-obstacle success rates across both density conditions.

Table 5. Simulation evaluation results across density conditions.

Condition	Episodes	Episode Success Rate	Per-Obstacle Success Rate
Sparse (2 trees)	50	94.00%	97.00%
Dense (3 trees)	50	84.00%	94.67%
Total	**100**	**89.11%**	**95.60%**

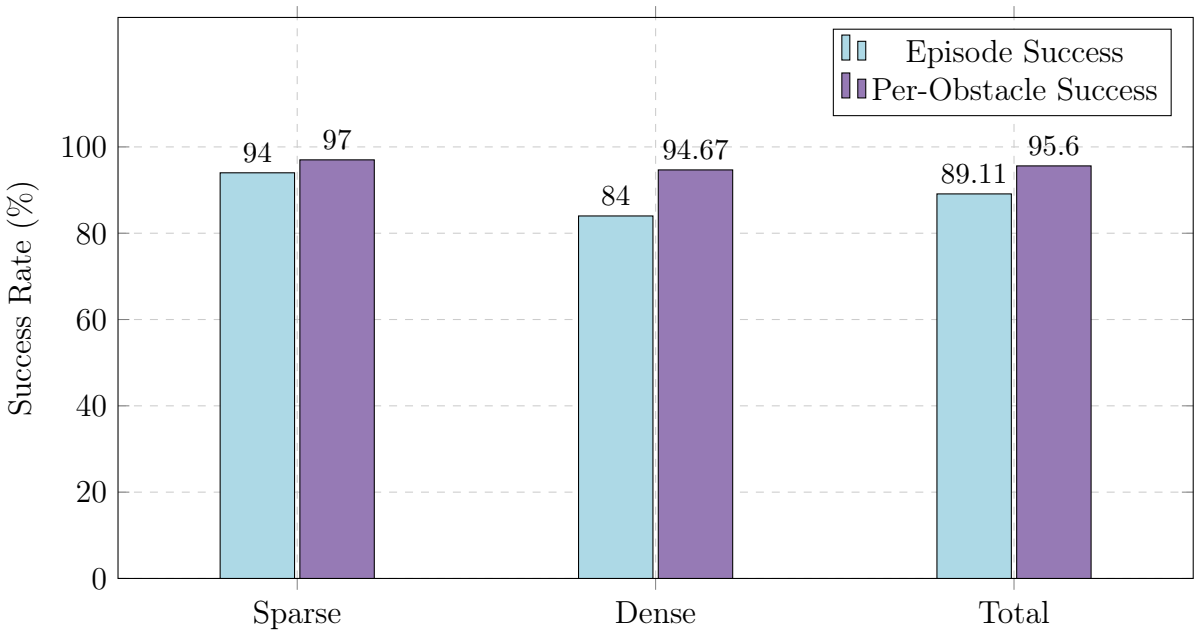


Figure 12. Simulation episode-level and per-obstacle success rates by tree density.

The agent achieved a 94.00% episode success rate in sparse conditions and 84.00% in dense conditions. The per-obstacle success rates were consistently higher than the episode-level rates in both conditions, reflecting that the agent frequently avoided most trees in an episode even when one collision ultimately terminated the run.

5.3. Real-World Results

The same policy was deployed without fine-tuning on the physical Holybro S500 V2 platform. A total of 100 real-world flights were conducted: 50 sparse and 50 dense. Table 6 summarises the results. Figure 13 shows the success rates by condition.

Table 6. Real-world flight results across density conditions.

Condition	Episodes	Episode Success Rate	Per-Obstacle Success Rate
Sparse (2 trees)	50	74.00%	83.00%
Dense (3 trees)	50	70.00%	78.52%
Total	**100**	**72.28%**	**80.32%**

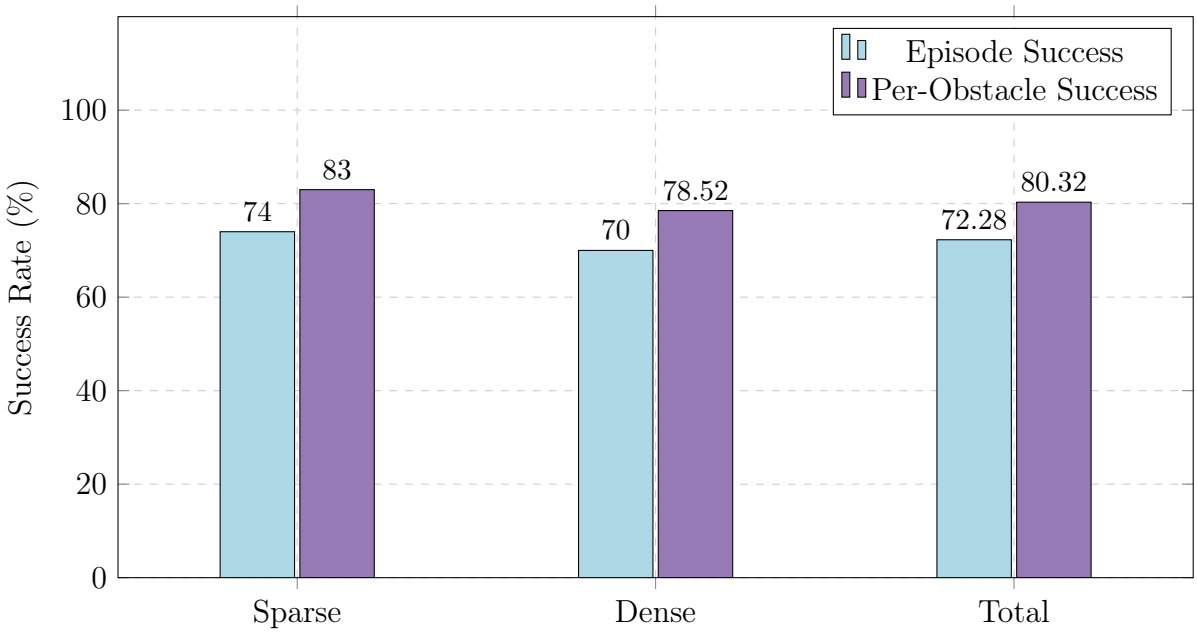


Figure 13. Real-world episode-level and per-obstacle success rates by tree density.

5.4. Simulation vs. Real-World Comparison

Figure 14 directly compares episode-level success rates between simulation and real-world for each density condition.

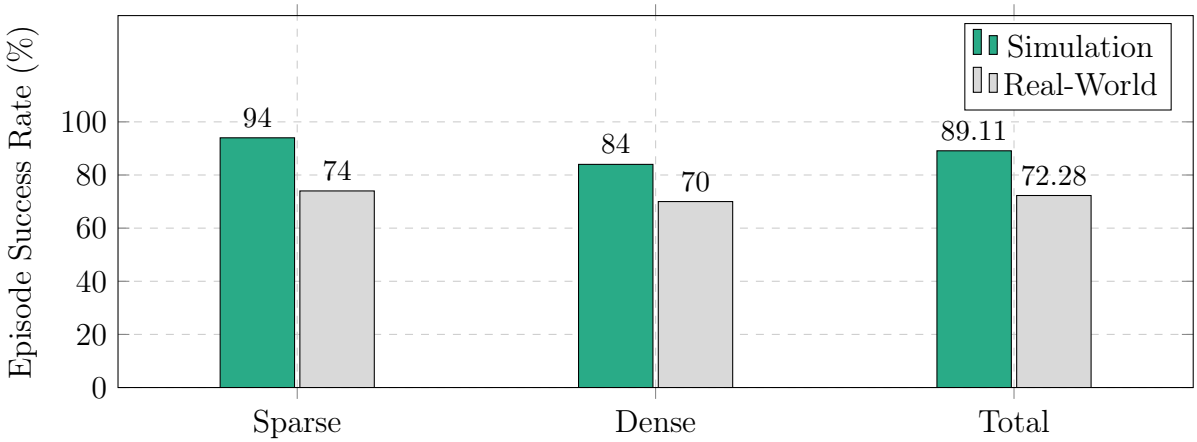


Figure 14. Episode-level success rate comparison between simulation and real-world across density conditions.

5.5. Sim-to-Real Transfer Efficiency

Transfer efficiency is computed as the real-world success rate divided by the simulation success rate for each matched condition [7], [19]. Table 7 reports transfer efficiency at both the episode and per-obstacle levels. Figure 15 visualises the transfer efficiency across conditions.

Table 7. Sim-to-real transfer efficiency by density condition.

Condition	Episode-Level Transfer	Per-Obstacle Transfer
Sparse (2 trees)	78.72%	85.57%
Dense (3 trees)	83.33%	82.95%
Total	**81.11%**	**84.02%**

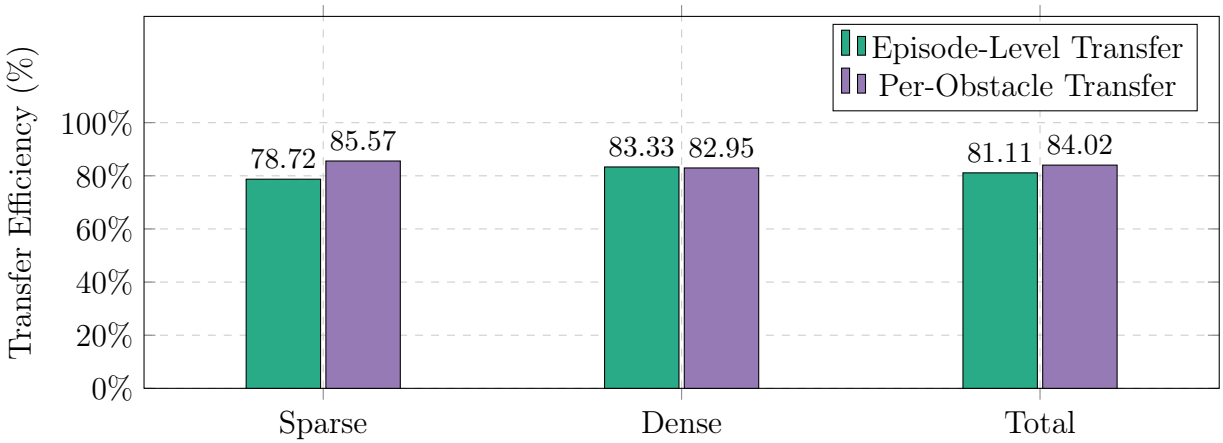


Figure 15. Sim-to-real transfer efficiency by density condition at episode and per-obstacle levels.

The overall episode-level transfer efficiency of 81.11% indicates that approximately one in five episodes that succeeded in simulation did not succeed in the real world. Per-obstacle transfer of 84.02% shows the agent retained most of its avoidance capability, with the gap primarily attributable to full-episode failures rather than a general degradation in obstacle avoidance behaviour.

6. Analysis and Discussions

6.1. Sparse vs. Dense Performance

The agent consistently achieved higher success rates in sparse conditions than in dense conditions across both simulation and real-world trials. In simulation, episode-level success dropped from 94.00% (sparse) to 84.00% (dense), and in real-world trials from 74.00% to 70.00%. This pattern is expected: with 3 trees in the corridor instead of 2, the agent faces a higher probability of consecutive avoidance decisions with less lateral recovery space between them. A single failed avoidance causes a collision and ends the episode regardless of how well the agent handles all other trees.

The per-obstacle success rates tell a complementary story. In dense real-world trials the agent avoided 78.52% of individual trees, yet only completed 70.00% of full episodes. This gap between per-obstacle and episode-level performance indicates that most failures are not caused by the agent being unable to avoid trees in general, but by the cumulative difficulty of avoiding all trees in sequence without any single error. The agent's avoidance capability degrades less with density than its episode completion rate suggests.

6.2. Sim-to-Real Transfer

The overall episode-level transfer efficiency of 81.11% and per-obstacle transfer of 84.02% represent a strong zero-shot sim-to-real result, meaning the policy was deployed on the physical drone without any real-world fine-tuning or adaptation. This is consistent with the best results reported in the sim-to-real literature for vision-based drone navigation [6], [20], [25].

We attribute this to the domain randomisation strategy applied during training. By randomising tree positions, angles, and trunk sizes across episodes, applying per-frame Gaussian image noise with brightness and contrast jitter, adding directional motion blur scaled by the lateral action, introducing camera-mounting jitter and spawn yaw jitter, injecting velocity drift of up to 0.3 m/s, and randomising the inference pipeline delay between 100 and 200 ms, the agent was exposed to a wide distribution of visual and physical conditions. This prevented overfitting to simulation-specific visual patterns and encouraged the policy to generalise to the real-world environment, where lighting, bark texture, and background vary in ways that cannot be fully anticipated in simulation.

Despite this, the remaining 18.89% episode-level gap between simulation and real-world performance reflects residual sources of reality gap that domain randomisation did not fully bridge. The most likely contributors are: aerodynamic differences between the simulated and physical Holybro S500 V2 platform, actuation latency between the moment a velocity command is sent to the PX4 flight controller and the moment the motors physically execute it, and minor camera calibration differences between the simulated sensor and the physical Arducam IMX219.

The per-obstacle transfer rate of 84.02% being higher than the episode-level rate suggests that the gap is not primarily a perception failure but a control precision issue: the agent sees the obstacles correctly but occasionally mistimes the avoidance manoeuvre in the real world due to physical dynamics the simulation did not perfectly capture.

6.3. Dense Conditions Show Higher Transfer Efficiency

A notable finding is that transfer efficiency was slightly higher for dense conditions (83.33%) than for sparse (78.72%) at the episode level. This is counterintuitive at first glance, since dense environments are harder. One plausible explanation is that the agent's learned policy in dense conditions relies more heavily on reactive, close-range avoidance behaviour that is less sensitive to small differences in flight dynamics between simulation and reality. In sparse conditions, the agent has more time and space to build up lateral velocity, meaning small differences in physical response can result in a wider trajectory deviation by the time the next tree is reached. In dense conditions, the shorter inter-tree spacing keeps the agent's corrections smaller and more frequent, making the policy more robust to individual control errors.

6.4. Limitations Observed in Results

The evaluation was conducted on cylindrical trunks under controlled conditions. The agent was not tested on branching trees, mixed species, or environments with ground cover that could occlude the lower trunk region. Performance in a genuine forest with greater visual complexity would likely be lower than reported here, and the current results should be interpreted as an upper bound for these controlled conditions rather than a direct estimate of deployment performance in operational bark beetle monitoring.

The dataset is also limited to sparse and dense configurations (2 and 3 trees in the corridor). A more complete evaluation would include a wider range of densities and longer corridors to better characterise how performance scales with environmental complexity.

The domain randomisation suite used in this thesis, while broad, leaves room for improvement. More aggressive randomisation (including dynamic obstacles, rain effects, varied lighting conditions, and a wider range of tree types and bark textures) could further close the sim-to-real gap. Whether the additional training complexity would yield proportional gains is an open question that motivates future work.

7. Conclusions

Taken as a whole, the experimental results are encouraging. The simulated drone reached an episode-level success rate of 89.11 % (and a per-obstacle success rate of 95.60 %) across the two density conditions, and that performance carried over to the real forest with an episode-level transfer efficiency of 81.11 % and a per-obstacle transfer efficiency of 84.02 %. Neither figure is 100 %, which means the system is clearly not production-ready yet, but the gap between the simulated and real-world numbers is small enough – and the absolute numbers are high enough across 200 evaluation episodes – that the results are statistically meaningful rather than noise.

The numbers become more impressive once the platform constraints are taken into account. The drone was equipped with only a single $20 monocular camera, no LiDAR or stereo, and was carrying that camera on a full-size Holybro S500 V2 frame whose footprint is much larger than the small agile racers usually used in vision-based avoidance studies. The camera itself runs at a narrow 62° field of view, which is unusually restrictive for forest flight and clips obstacles from the periphery sooner than the wide-angle lenses used in comparable work. That this configuration still reaches above 80 % transfer is the central empirical result of this thesis.

The fact that those numbers were obtained with such a minimal sensor setup also points forward. The camera is small enough that the airframe could likely be shrunk significantly without losing perception capacity, and either widening the field of view or adding a second camera to recover lost peripheral information would be a natural next step that could plausibly push the transfer numbers higher than the ones reported here.

8. Future Work

This section outlines the most meaningful directions for extending the work presented in this thesis, ordered from the most immediate next steps to broader longer-term research directions.

8.1. Lighter and More Agile Drone Platform

The most impactful next step is to migrate the navigation system onto a smaller, lighter airframe. The Holybro S500 V2 was chosen for this thesis primarily for convenience and stability, as the focus was on the policy and the sim-to-real pipeline rather than on hardware optimisation. However, the key finding that the Arducam IMX219 — an extremely small and lightweight camera — is sufficient for reliable tree avoidance opens the door to a fundamentally different hardware profile.

A smaller drone with less mass and lower inertia would respond to velocity commands faster and more precisely, directly improving avoidance performance without any changes to the policy. The same navigation model could be deployed on a platform weighing a fraction of the S500, making the overall system cheaper, more portable, and scalable to fleet operation. Building a purpose-built lightweight platform around this camera-and-policy stack is therefore the most direct path to improving real-world performance and realising the democratisation goal of this thesis.

8.2. Application Integration: Forest Monitoring

This thesis establishes the autonomous navigation capability that makes repeated aerial forest surveys practical. The natural next step is to build a full monitoring application on top of this base. One concrete example is bark beetle detection: by integrating a multispectral or RGB-based infestation detection model alongside the navigation policy, the drone could autonomously survey a forest corridor, avoid obstacles, and simultaneously classify tree health from the same camera feed. This would realise the original BeetleSense use case that motivated this investigation, and the same pattern applies to other forest monitoring applications such as wildfire early detection or biodiversity surveys.

8.3. Real-World Policy Fine-Tuning

The policy in this thesis is deployed zero-shot, with no real-world adaptation after simulation training. While the 81% transfer efficiency achieved here is strong, the remaining gap could be reduced by collecting a small set of real flight episodes and using them to fine-tune the sim-trained policy. Techniques such as domain adaptation and real-world data augmentation have been shown to close the sim-to-real gap significantly with minimal additional data [26]. Applying these methods to the trained policy would be a low-effort intervention likely to produce a measurable improvement in real-world success rates without requiring a full retraining from

scratch.

8.4. More Complex Environments

The evaluation in this thesis was conducted on naked cylindrical trunks under controlled conditions. A critical next step is testing the policy in environments that better represent a real operational forest: trees with branches, mixed species with varying bark textures and colours, ground vegetation that partially occludes the lower trunk, and uneven terrain. It is expected that performance would degrade in these settings, and understanding where and why would provide a clear roadmap for the next generation of domain randomisation and training strategies.

8.5. Full Autonomy Stack

The obstacle avoidance capability demonstrated in this thesis is one component of a fully autonomous forest drone. A complete system would also require GPS-denied localisation to operate under forest canopy where satellite signals are degraded, waypoint-level path planning to define and execute a survey route across a large area, and return-to-home or safe-landing behaviour on battery or fault events. Integrating the learned avoidance policy into such a full autonomy stack, where it operates as the low-level reactive layer within a higher-level mission planner, is the architectural direction required for operational deployment.

References

[1] M. Z. Butt, N. Nasir, and R. A. Rashid, "A review of perception sensors, techniques, and hardware architectures for autonomous low-altitude UAVs in non-cooperative local obstacle avoidance," *Robotics and Autonomous Systems*, vol. 174, p. 104 629, 2024. DOI: 10.1016/j.robot.2024.104629

[2] J. Fu and F. Yang, "Application of reinforcement learning in UAV tasks: A survey," *Unmanned Systems*, 2024. DOI: 10.1142/s2301385026300015

[3] A. Merei, H. Mcheick, A. Ghaddar, and D. Rebaine, "A survey on obstacle detection and avoidance methods for UAVs," *Drones*, vol. 9, no. 3, p. 203, 2025. DOI: 10.3390/drones9030203

[4] D. Debnath, F. Vanegas, J. Sandino, A. F. Hawary, and F. Gonzalez, "A review of UAV path-planning algorithms and obstacle avoidance methods for remote sensing applications," *Remote Sensing*, vol. 16, no. 21, p. 4019, 2024. DOI: 10.3390/rs16214019

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

[6] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019. DOI: 10.1109/TRO.2019.2942989

[7] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020. DOI: 10.1109/SSCI47803.2020.9308468

[8] S. Hrabar, "An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance," *Journal of Field Robotics*, vol. 29, no. 2, pp. 215–239, 2012. DOI: 10.1002/rob.21404

[9] T. Kloušek, J. Komárek, P. Surový, K. Hrach, P. Janata, and B. Vašíček, "The use of UAV mounted sensors for precise detection of bark beetle infestation," *Remote Sensing*, vol. 11, no. 13, p. 1561, 2019. DOI: 10.3390/rs11131561

[10] V. Bozzini et al., "Drone-based early detection of bark beetle infested spruce trees differs in endemic and epidemic situations," *Forest Ecology and Management*, vol. 563, p. 121 972, 2024. DOI: 10.1016/j.foreco.2024.121972

[11] S. Bijou, L. Kupková, L. Červená, and J. Lysák, "Assessing early bark beetle infestation in Norway spruce using multitemporal hyperspectral UAV imagery: Is detection during the green stage always timely?" *Ecological Indicators*, vol. 178, p. 113 869, 2025. DOI: 10.1016/j.ecolind.2025.113869

[12] E. Turkulainen et al., "Comparison of deep neural networks in the classification of bark beetle-induced forest damage," *Remote Sensing*, vol. 15, no. 20, p. 5253, 2023. DOI: 10.3390/rs15205253

[13] H. R. Khanzada, A. Maqsood, and A. Basit, "Reinforcement learning for UAV flight controls: Evaluating continuous space reinforcement learning algorithms for fixed-wing UAVs," *PLoS ONE*, 2025. DOI: 10.1371/journal.pone.0334219

[14] M. R. Vizzotto, G. Kohl, C. E. Pereira, M. Tropea, and E. P. de Freitas, "A DRL-based trajectory planning strategy for UAV navigation: A comparison of the PPO algorithm with DDQN, DDPG, and A*," in *Proceedings of the IEEE International Conference on Advanced Robotics (ICAR)*, 2025. DOI: 10.1109/ICAR65334.2025.11338666

[15] M. S. Tayar et al., "Autonomous UAV flight navigation in confined spaces: A reinforcement learning approach," in *Proceedings of the IEEE Latin American Robotics Symposium (LARS)*, 2025. DOI: 10.1109/LARS69345.2025.11273007

[16] B. Kabas, "Autonomous UAV navigation via deep reinforcement learning using PPO," in *Proceedings of the IEEE Signal Processing and Communications Applications Conference (SIU)*, 2022. DOI: 10.1109/SIU55565.2022.9864769

[17] R. Kumar, A. M. Vanjare, and S. N. Omkar, "Autonomous drone navigation using monocular camera and light weight embedded system," in *Proceedings of the IEEE International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2023. DOI: 10.1109/ICONAT57137.2023.10080483

[18] A. P. Kalidas et al., "Deep reinforcement learning for vision-based navigation of UAVs in avoiding stationary and mobile obstacles," *Drones*, vol. 7, no. 4, p. 245, 2023. DOI: 10.3390/drones7040245

[19] 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, 2021. DOI: 10.1109/ACCESS.2021.3126658

[20] Y. Jang, S. Yoon, J. Baek, and S. Han, "Sim-to-real policy transfer in deep reinforcement learning," in *Proceedings of the IEEE International Conference on Control, Automation and Systems (ICCAS)*, 2024. DOI: 10.23919/ICCAS63016.2024.10773220

[21] L. Yang et al., "Depth Anything V2," in *Advances in Neural Information Processing Systems (NeurIPS)*, arXiv preprint arXiv:2406.09414, 2024.

[22] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, *Asymmetric actor critic for image-based robot learning*, arXiv preprint arXiv:1710.06542, 2017.

[23] G. Del Col, V. Karjalainen, T. Hakala, and E. Honkavaara, "Autonomous drone navigation in forest environments using deep learning," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-2/W11-2025, UAV-g 2025*, Espoo, Finland, 2025, pp. 87–94. DOI: 10.5194/isprs-archives-XLVIII-2-W11-2025-87-2025

[24] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022. DOI: 10.1109/TPAMI.2020.3019967

[25] R. Azzam et al., "Learning-based navigation and collision avoidance through reinforcement for UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, 2024. DOI: 10.1109/TAES.2023.3294889

[26] P. Qin, T. Zhao, Q. Su, and H. Yang, "Mobile robot navigation via domain randomization and real-world adaptation," in *Proceedings of the IEEE International Conference on Unmanned Systems (ICUS)*, 2025. DOI: 10.1109/ICUS66297.2025.11294152

Appendix

A. Datasets

A.1 Real Flight Dataset

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
1	Sparse	2	2	Success
2	Dense	3	3	Success
3	Dense	3	1	Fail
4	Dense	3	3	Success
5	Dense	3	3	Success
6	Sparse	2	2	Success
7	Sparse	2	0	Fail
8	Sparse	2	1	Fail
9	Sparse	2	2	Success
10	Sparse	2	2	Success
11	Dense	3	2	Fail
12	Dense	3	3	Success
13	Dense	3	3	Success
14	Dense	3	3	Success
15	Dense	3	1	Fail
16	Dense	3	3	Success
17	Sparse	2	2	Success
18	Sparse	2	2	Success
19	Sparse	2	2	Success
20	Sparse	2	2	Success
21	Sparse	2	1	Fail
22	Sparse	2	2	Success
23	Sparse	2	0	Fail
24	Dense	3	3	Success
25	Dense	3	0	Fail
26	Dense	3	3	Success
27	Dense	3	3	Success
28	Dense	3	3	Success
29	Dense	3	0	Fail
30	Dense	3	3	Success
31	Dense	3	0	Fail
32	Dense	3	0	Fail
33	Dense	3	3	Success
34	Dense	3	3	Success
35	Sparse	2	2	Success

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
36	Sparse	2	2	Success
37	Dense	3	3	Success
38	Dense	3	3	Success
39	Dense	3	3	Success
40	Sparse	2	0	Fail
41	Sparse	2	0	Fail
42	Sparse	2	2	Success
43	Sparse	2	2	Success
44	Dense	3	2	Fail
45	Dense	3	3	Success
46	Dense	3	1	Fail
47	Dense	3	3	Success
48	Sparse	2	2	Success
49	Dense	3	3	Success
50	Sparse	2	2	Success
51	Dense	3	3	Success
52	Dense	3	3	Success
53	Dense	3	2	Fail
54	Dense	3	0	Fail
55	Dense	3	1	Fail
56	Dense	3	0	Fail
57	Dense	3	2	Fail
58	Dense	3	3	Success
59	Sparse	2	2	Success
60	Sparse	2	2	Success
61	Sparse	2	2	Success
62	Dense	3	3	Success
63	Dense	3	3	Success
64	Sparse	2	2	Success
65	Sparse	2	2	Success
66	Sparse	2	1	Fail
67	Dense	3	3	Success
68	Sparse	2	2	Success
69	Sparse	2	2	Success
70	Dense	3	1	Fail
71	Dense	3	3	Success
72	Sparse	2	2	Success
73	Dense	3	3	Success
74	Sparse	2	2	Success
75	Dense	3	3	Success

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
76	Sparse	2	2	Success
77	Dense	3	3	Success
78	Sparse	2	2	Success
79	Sparse	2	1	Fail
80	Dense	3	3	Success
81	Sparse	2	2	Success
82	Sparse	2	2	Success
83	Dense	3	3	Success
84	Sparse	2	1	Fail
85	Sparse	2	1	Fail
86	Dense	3	3	Success
87	Sparse	2	2	Success
88	Sparse	2	2	Success
89	Dense	3	3	Success
90	Sparse	2	2	Success
91	Sparse	2	2	Success
92	Sparse	2	2	Success
93	Sparse	2	1	Fail
94	Sparse	2	2	Success
95	Sparse	2	2	Success
96	Sparse	2	1	Fail
97	Sparse	2	2	Success
98	Sparse	2	1	Fail
99	Sparse	2	2	Success
100	Sparse	2	2	Success

A.2 Simulation Dataset

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
1	Sparse	2	2	Success
2	Sparse	2	2	Success
3	Sparse	2	2	Success
4	Sparse	2	2	Success
5	Sparse	2	2	Success
6	Sparse	2	2	Success
7	Sparse	2	2	Success
8	Sparse	2	2	Success

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
9	Sparse	2	2	Success
10	Sparse	2	2	Success
11	Sparse	2	2	Success
12	Sparse	2	2	Success
13	Sparse	2	2	Success
14	Sparse	2	1	Fail
15	Sparse	2	2	Success
16	Sparse	2	2	Success
17	Sparse	2	2	Success
18	Sparse	2	2	Success
19	Sparse	2	2	Success
20	Sparse	2	2	Success
21	Sparse	2	2	Success
22	Sparse	2	2	Success
23	Sparse	2	2	Success
24	Sparse	2	1	Fail
25	Sparse	2	2	Success
26	Sparse	2	2	Success
27	Sparse	2	2	Success
28	Sparse	2	2	Success
29	Sparse	2	2	Success
30	Sparse	2	2	Success
31	Sparse	2	2	Success
32	Sparse	2	2	Success
33	Sparse	2	2	Success
34	Sparse	2	2	Success
35	Sparse	2	2	Success
36	Sparse	2	2	Success
37	Sparse	2	2	Success
38	Sparse	2	2	Success
39	Sparse	2	2	Success
40	Sparse	2	2	Success
41	Sparse	2	2	Success
42	Sparse	2	2	Success
43	Sparse	2	2	Success
44	Sparse	2	2	Success
45	Sparse	2	2	Success
46	Sparse	2	2	Success
47	Sparse	2	2	Success
48	Sparse	2	2	Success

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
49	Sparse	2	1	Fail
50	Sparse	2	2	Success
51	Dense	3	3	Success
52	Dense	3	3	Success
53	Dense	3	2	Fail
54	Dense	3	3	Success
55	Dense	3	3	Success
56	Dense	3	3	Success
57	Dense	3	3	Success
58	Dense	3	3	Success
59	Dense	3	3	Success
60	Dense	3	3	Success
61	Dense	3	2	Fail
62	Dense	3	3	Success
63	Dense	3	3	Success
64	Dense	3	3	Success
65	Dense	3	3	Success
66	Dense	3	2	Fail
67	Dense	3	3	Success
68	Dense	3	3	Success
69	Dense	3	3	Success
70	Dense	3	3	Success
71	Dense	3	3	Success
72	Dense	3	3	Success
73	Dense	3	3	Success
74	Dense	3	3	Success
75	Dense	3	3	Success
76	Dense	3	3	Success
77	Dense	3	3	Success
78	Dense	3	3	Success
79	Dense	3	3	Success
80	Dense	3	3	Success
81	Dense	3	3	Success
82	Dense	3	3	Success
83	Dense	3	3	Success
84	Dense	3	2	Fail
85	Dense	3	3	Success
86	Dense	3	3	Success
87	Dense	3	2	Fail
88	Dense	3	3	Success

Episode	Tree Density	Trees in Corridor	Trees Avoided	Result
89	Dense	3	2	Fail
90	Dense	3	3	Success
91	Dense	3	3	Success
92	Dense	3	3	Success
93	Dense	3	2	Fail
94	Dense	3	3	Success
95	Dense	3	3	Success
96	Dense	3	2	Fail
97	Dense	3	3	Success
98	Dense	3	3	Success
99	Dense	3	3	Success
100	Dense	3	3	Success