Autonomous Perching Drones: Design, Perception and Control

RO57035 Robotics: MSc Thesis

Georg Miguel Strunck

Delft University of Technology

TU Delft

Autonomous Perching Drones: Design, Perception and Control

Georg Strunck*

MSc Thesis

Delft University of Technology

Delft, Zuid-Holland, 2628 CD, The Netherlands

September 9, 2025

Abstract

Autonomous perching enables micro aerial vehicles (MAVs) to act as quiet, non-intrusive sensing nodes for ecological monitoring and habitat assessment. We present a 1.2[kg] quadrotor designed for selective branch perching in natural environments, aiming to extend operational endurance and enable elevated observation in dense vegetation. We achieve autonomous quadrotor perching by addressing challenges in perception, planning, mechanical design and control. Our system, equipped with a stereo vision pipeline and a lightweight top-mounted actuated gripper arm, demonstrates fully autonomous perching across a wide range of branch and stem orientations - from horizontal to vertical - even in cluttered, near-natural forest scenes. In controlled laboratory trials, the MAV performs a 360° scan to identify candidate perches, verifies them with depth-based checks, executes a bottom-up, orthogonal approach, and supports repeated take-off and re-perching at different branches. This is the first time the complete pipeline of design, perception, planning, and control is addressed for perching drones. With that, our work lays the foundation for future deployment in ecological monitoring and restoration missions, where minimally invasive aerial observation could benefit remote or sensitive ecosystems.

Key words: CNN, machine-learning, branch detection, autonomous perching, stereo vision, aerial robotics, robotic grasping, MAV, environmental monitoring.

I. Introduction to Perching MAVs

Micro Aerial Vehicles (MAVs) have emerged as versatile tools for environmental monitoring, ecological research, and infrastructure inspection, where their agility enables access to areas that are otherwise difficult or hazardous for humans. Among the most challenging manoeuvres in these applications is perching—the act of landing and attaching to elevated structures such as tree branches. Perching allows MAVs to conserve energy, extend mission duration, and gain stable vantage points for high-quality sensing. These capabilities are particularly valuable for long-term wildlife observation, biodiversity surveys in forest canopies, and persistent monitoring of sensitive ecosystems [1].

In nature, birds and other flying or gliding animals demonstrate perching strategies honed over millions of years of evolution. They combine precise visual targeting, agile approach trajectories, and specialised grasping appendages to land on perches of varying orientation, diameter, and surface texture. Translating these capabilities into aerial robots has inspired a decade of research into branch detection, pose estimation, and adaptive grasping mechanisms. While many laboratory demonstrations have achieved reliable perching on known or structured targets, enabling fully autonomous perching in cluttered, unstructured outdoor environments—without external localisation aids—remains a key challenge [7].

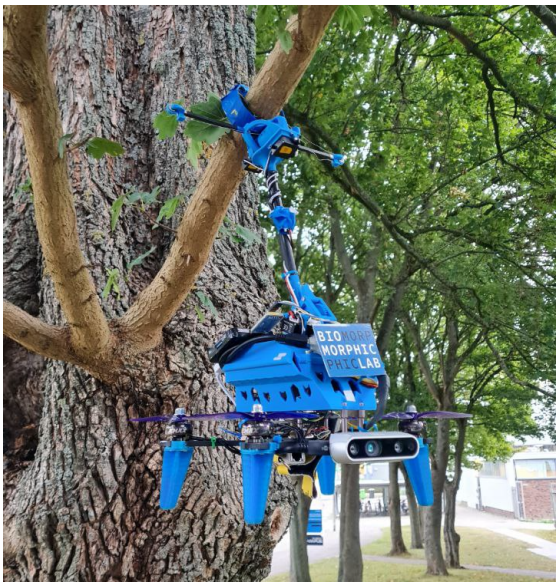


Figure 1: *Nagapie perched at a 30° inclined branch with camera pointing outwards for better observation space.*

This work presents a novel aerial platform specifically designed for autonomous branch perching in natural environments. The 1.2[kg] quadrotor integrates a forward-facing stereo camera and an NVIDIA

Jetson Xavier NX for onboard perception and planning, paired with a one-axis rotating arm carrying a lightweight "slapping" gripper. In perching mode, the MAV performs a 360° scan of its surroundings to identify candidate perches using a convolutional neural network (CNN) trained on branch imagery. A selected target is approached front-on for verification and depth-based geometric validation. The system then re-orients the arm, executes a controlled approach from beneath the perch in an orthogonal direction, and finally clasps onto the branch upon contact.

Key perception components include an RGB-D perch detection pipeline based on a fine-tuned YOLOv11s model, coupled with depth-consistency heuristics (patch-hole and depth-variation checks) to produce robust 3D pose estimates in cluttered canopies. Planning and execution are handled by a ROS2 based trajectory generation framework, integrating a soft-contact detection method that infers stable perch contact from motor thrust feedback and position stability - avoiding the noise sensitivity of accelerometer-based triggers.

By combining modern deep-learning-based detection with depth-validated 3D pose extraction, planning and a controlled perch approach, and a compliant grasping mechanism, this system addresses the three principal barriers to outdoor autonomous perching: (1) reliable detection of natural perches, (2) accurate perch pose estimation without external localisation, and (3) robust grasping under variable branch geometry and surface conditions. The resulting pipeline can be directly applied for ecological monitoring and ecosystem restoration, where silent, extended-duration observation from within the canopy can yield richer datasets with minimal environmental disturbance.

II. Background & Related Work

A. Biological Inspiration

Perching in flying animals is a mostly visually guided, tightly timed manoeuvre. Birds typically keep the perch continuously in view and adjust their final approach to minimize the distance moved in air after stall. Harris hawks, for instance, hold the perch within a small binocular foveal region throughout the landing, and transition from a dive to a climb such that the last low-speed, high-lift segment is as short as possible. These behaviours result in a characteristic "swoop–flare–stall" sequence immediately before touchdown.

A consistent theme across species is gaze stabilization and time-to-contact control using optic-flow cues. Experiments show pigeons maintain visual fixation above the beak during approach, regulate braking with the derivative of time-to-contact, and initiate the final flare based on simple, robust image measurements (e.g., apparent target size). For small birds with sufficient power margin, hovering touchdowns are also observed; larger birds favour the swoop-up strategy. These findings motivate MAV perception–control loops that (i) maintain the perch in view as long as possible and (ii) explicitly manage the final metres with aggressive braking and attitude changes.

B. Previous Perching MAV Research

Attachment mechanisms. A broad spectrum of grippers has been explored: bistable metal/composite claws that snap shut on cylindrical perches [17], passive, tendon-locking avian-style talons for robust wrap-and-hold [12], compliant microspines that engage bark asperities [6], gecko-inspired dry adhesives for smooth vertical surfaces [14], and electrostatic adhesion for very small flyers and overhangs [5]. Each mechanism trades payload, required approach precision, and surface characteristics.

Trajectory planning and control. Dynamic perching has been realised with polynomial trajectory frameworks to perch on branches inclined up to $\sim 40°$ [17], as well as nonlinear programming/MPC formulations that are perception-aware (penalizing loss of target visibility) and that plan recoveries if perching fails. Recent SE(3) planners replan in milliseconds, enabling perching on moving targets and at extreme attitudes (including flips for powerline perching) [10]. These approaches echo the avian strategy by explicitly rewarding target observability along the approach.

Sensing and state estimation. We follow the split into (i) perch detection, (ii) relative pose estimation in the last metres, and (iii) system integration, but emphasise that the present work targets (i) and (ii) only and uses an external motion-capture system (OptiTrack) for vehicle pose during experiments.

(i) *Perch detection:* Prior outdoor-oriented pipelines centre on stereo/RGB–D geometry with RANSAC cylinder fitting [8], sometimes preceded by colour-based filtering; on a laptop VM this ran at $\sim$14[Hz] but only $\sim$5[Hz] on-board (Jetson Xavier NX), with accuracy dominated by stereo depth quality at close range. Learning-based RGB–D methods that reconstruct woody structure (e.g., TransUNet/ESANet) show promise in clutter but remain slow (reported $\sim$3[Hz] on a server GPU) [4] and do not yet return perchable locations directly. Orchard-style segmentation and line/CNN hybrids exist but rely on highly structured scenes or controlled lighting [13, 16]. Several "last-centimetre" aids (line-scan cameras, whiskers, ToF arrays) have been proposed for final alignment under foliage [2, 18]. Our paper supersedes this detection stage by delivering a real-time RGB–D branch detector tailored for natural, unstructured environments and integrating depth-validation heuristics, enabling fast reliable detect-and-approach flights to natural branches.

(ii) *Relative pose estimation:* Robust *outdoor* MAV–perch relative pose at actionable rates remains open. The literature notes only limited work on reaching the perch without external references; VIO/VINS pipelines excel mainly in structured indoor scenes, with few demonstrations meeting the ac-

curacy and update-rate demands in vegetation [11]. In this project we therefore do not contribute to self-pose estimation and instead use OptiTrack to isolate and evaluate the perception and guidance stack.

(iii) *System integration:* Perception-aware planning and very fast SE(3) replanners exist (e.g., visibility-aware costs and MINCO-based perching/tracking) [10, 17], but demonstrations typically rely on instrumented targets (barcodes) and GNSS–VIO, or address dynamic tracking rather than natural-branch perching in unstructured clutter. An end-to-end loop that maintains target observability, verifies contact, and closes the final metres *outdoors* on natural branches without markers is being presented for the first time in our work.

Relation to our system. Relative to the three threads above, our paper (a) advances *(i) perch detection* by replacing classical stereo–RANSAC pipelines with an RGB–D CNN that outputs geometry-consistent 3D line poses in real time on a Jetson Xavier NX and validates range via local depth hole and variance checks; (b) deliberately does *not* address *(ii) relative pose estimation*—vehicle state comes from OptiTrack so we can isolate perception and guidance performance; and (c) contributes a targeted element of *(iii) system integration*: a detect–select–verify–approach–grasp loop that begins with a 360° scan to rank candidates, flies to a verification pose to re-check confidence with aligned depth, then orients a 1-DOF top-mounted arm to execute a bottom-up, orthogonal approach with a slapping gripper.

C. Ecological Monitoring Relevance

Perching MAVs extend mission endurance and observational quality in environmental science. By landing at elevated, unobtrusive vantage points (e.g., canopy level), a MAV can (i) conserve energy relative to sustained hover, (ii) exploit direct sunlight for solar trickle charging to lengthen deployments, and (iii) collect long-horizon, high-stability data (imagery, acoustics, microclimate) with minimal disturbance compared to repeated flyovers. These capabilities are especially valuable where animal life mostly inhabits the canopy layer as in rainforests and where terrain or moisture precludes safe ground landings.

Beyond ecology, perching increases safety and efficiency in infrastructure inspection (power lines, bridges), enables ad-hoc communications relays after disasters, and supports anti-poaching surveillance with persistent, elevated viewpoints. The same robotic competencies, reliable on-board perception, geometry-aware grippers, and visibility-aware approach planning, transfer directly to conservation applications that demand low acoustic/visual footprint and long waiting times.

III. System Overview

The *Nagapie* platform is a 1.2[kg] quadrotor designed to investigate autonomous perching on natural branches. Its design combines lightweight hardware components, embedded computing for real-time perception, a custom perching mechanism, and a modular software stack. The overall aim is not only to demonstrate repeatable perching in controlled settings but also to provide a system architecture that can be extended to more complex ecological monitoring scenarios. To this end, the vehicle integrates four principal subsystems: (i) the mechanical platform and perching mechanism, (ii) onboard perception hardware, (iii) computation, flight control, and auxiliary microcontrollers, and (iv) the software architecture that links sensing to action. Each subsystem has been selected with the dual requirements of flight feasibility and perching reliability in mind. The following paragraphs outline these subsystems in turn and describe how they interact during a complete perching sequence.

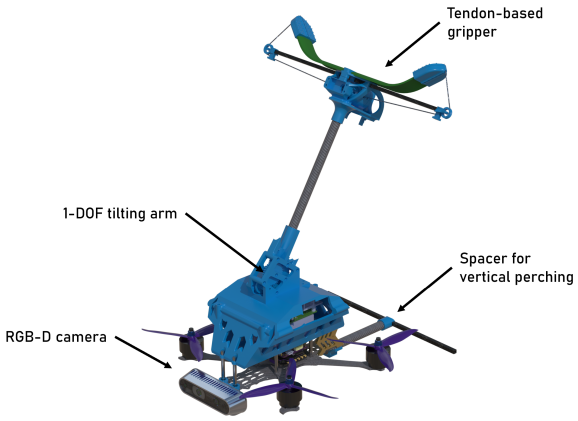


Figure 2: *Render of Nagapie showing the forward-facing camera, housing for the onboard computer, top mounted gearbox, arm and gripper (current position at 45°) as well as the backwards positioned spacer for vertical perching.*

Mechanical integration. The vehicle is based on a quadrotor frame (SpeedyBee FS225 v2) that provides sufficient space for the perching module and onboard computer. A 3D-printed housing encloses the Nvidia Jetson Xavier NX, offering protection while also serving as mounting structure for the perching mechanism. On top of the quadcopter, the 20[cm] rotating arm is mounted, actuated over a 0–90° range. At its end, a lightweight "slapping" gripper is attached, featuring a clutch-like rewind and release system. Three digital servos control the arm rotation, the gripper rewind, and the release function. These are driven by a Teensy microcontroller that interfaces with the onboard computer via a custom ROS 2 driver. For enabling vertical perching a distancing tail arm is attached to prevent rotating off the perch. The flight battery is attached beneath the frame and protected from ground contact by extended landing legs, ensuring safe takeoff and landing.

Perception hardware. Perch detection relies on an Intel RealSense D435 RGB–D camera. Its small form factor and low weight make it suitable for flight, while its active-stereo depth sensing provides dense

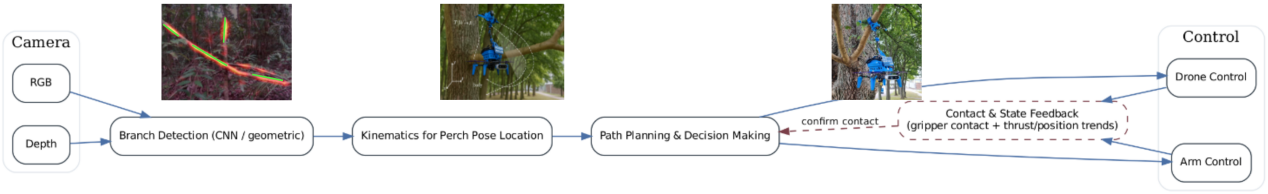


Figure 3: *Data flow diagram of the proposed perching system. On the left side the camera makes RGB-D images available, which are fed through the perch detection and transformation to the planner resulting in gripper and quadcopter setpoint goals on the right with contact feedback during the last approach.*

range information at close distances (typically below 3[m]), which is the regime of interest for perching. The sensor is operated with depth–to–color alignment enabled, producing directly comparable RGB and depth frames for downstream processing.

Onboard computation and flight control. An Nvidia Jetson Xavier NX serves as the onboard computer, running the perception and control software within an Ubuntu22 Isaac Docker environment. The GPU resources of the Xavier support real-time execution of convolutional neural networks for branch detection, while the CPU cores handle ROS2 nodes and middleware. Low-level stabilization is provided by a Pixracer-r15 flight controller running PX4 v1.15.4. Because enabling Fast DDS on the Pixracer saturates its CPU, the firmware was modified to publish only a minimal set of DDS topics required for off-board operation. Communication between PX4 and ROS2 is handled via micro-ROS, using the MicroXRCEAgent to bridge Fast DDS between the flight controller and the Xavier. Telemetry is transmitted via an XBee radio, while a separate RC link provides a safety pilot with direct override control.

Software architecture. The software stack follows a clear processing chain. The first node publishes aligned RGB and depth image streams. Then they are passed through a convolutional neural network trained on branch images, and generate candidate perches. Depth information is then used to filter out spurious detections and to adjust local consistency, reducing point-wise outliers, before publishing the results relative to the camera frame. These candidates are transformed into the global NED frame using the drones configuration geometry and global pose. The planning node then coordinates higher-level actions: during a 360° scan it accumulates candidate perches, selects one, and commands PX4's off-board action server to fly to a verification position 1[m] ahead of the perch. After depth validation, it instructs the Teensy to rotate the arm into alignment and then commands an orthogonal approach from below the branch. At contact, the gripper is commanded to clasp. The same node manages unperching by releasing the gripper and returning control to free flight.

Ground truth pose estimation. In laboratory experiments, vehicle pose is obtained from an OptiTrack motion capture system. The data is streamed via a LAN connection to the Xavier, which also allows remote SSH control for starting and monitoring ROS 2 nodes. This setup ensures repeatable tests in controlled conditions. Outdoor tests will not use external tracking and will rely solely on onboard perception for state estimation.

Integration summary. The system brings together lightweight depth sensing, embedded GPU computing, a microcontroller-driven perching mechanism, and a flight controller configured for minimal data exchange over a FastDDS bridge. Together these elements form a processing and control chain that detects, verifies, and attaches to natural branches, while allowing unperching and further flight. The integration is designed to remain simple enough for repeatable experimentation, while enabling future extensions toward longer-term ecological monitoring missions.

IV. Mechanical Design

A central component of the Nagapie MAV is the perching arm, which is mounted on top of the vehicle's housing. The arm is approximately 20[cm] in length and is driven by a custom-designed gearbox actuated through a Feetech STS3032 multi-turn servo. At its end, the arm carries a bistable "slapping" gripper with a rewinding and release clutch mechanism. This section outlines the design considerations of both the gearbox and the gripper system.

A. Gearbox Architecture

The gearbox follows a multi-stage spur gear layout, with gears dimensioned using an involute tooth profile for reliable transmission. A three-stage reduction was implemented, each stage consisting of a 10-tooth pinion driving a 38-tooth gear, yielding an overall ratio of approximately 54:1. With the STS3032 servo capable of 15 effective input rotations (seven backward, eight forward), this reduction translates the multi-turn input into roughly 90° of output rotation. This range was selected to cover the required motion from a rearward-pointing horizontal arm to a vertical upward orientation.

The gear width of 5[mm] provided sufficient stiffness while allowing practical 3D-printed prototyping during the design phase. The three-stage configuration offered the torque multiplication necessary to reliably actuate the gripper arm, while maintaining ad-

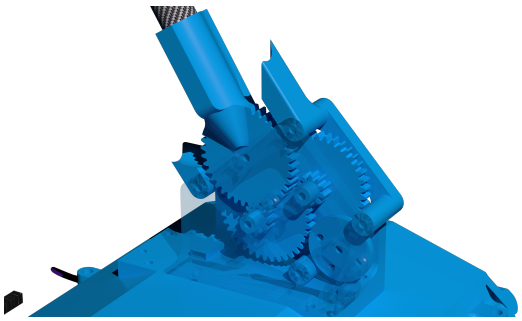


Figure 4: *Close-up of the gearbox (lid transparent for inside view) mounted on top of Nagapie. It has been kept to a minimum mass design, while ensuring the arm is operable even when moved to horizontal position when perching at vertical stems/branches.*

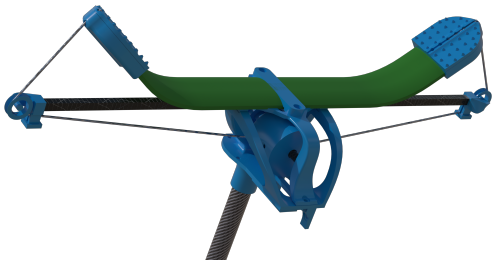


Figure 5: *Close-up of the gripper at the end of the arm. The bistable metal on top (green) curls around the branch when the spool (below center) is released from the rewinding clutch. Wire guides at the carbon rod endpoints rotate on ball bearings to reduce the servo loading.*

equate rigidity and minimizing backlash. Prototypes confirmed smooth meshing at module $m = 0.7$, which provided a balance between compactness and load capacity. The gearbox was integrated into a compact housing within the arm structure, directly coupled to the servo output shaft.

B. Gripper Design

At the free end of the 20[cm] arm, the gripper consists of a bistable metal element fabricated from five stacked spring steel strips with a curved cross-section. The bistability ensures two stable configurations: a straightened state and a curled, rolled-up state. The strips are fixed centrally to the arm, so that upon actuation the gripper rolls symmetrically around the perch. This passive bistability ensures that once triggered, the gripper rapidly and securely encloses the branch [8, 17].

To increase robustness and safety, the endpoints of the gripper are capped with 3D-printed 'claws'. These serve both to secure the metal sheets mechanically and to reduce sharp edges, while also providing anchor points for the rewinding cables. The gripper is mounted centered on top of a lightweight carbon tube that acts as a guide for the rewinding wires, supported at the tip on ball bearings to reduce the radial load on the servo.

C. Clutch and Rewind Mechanism

The opening of the bistable gripper is actively controlled through a dual-cable rewinding system. Both cables run from the gripper endpoints around the ball-bearing tip of the carbon guide rod to a spool mounted at the gripper's center base. The spool is driven by another STS3032 servo (same as used for the arm rotation), with a clutch mechanism enabling selective engagement. The clutch consists of a four-tooth disk interface: with a small Feetech SCS0009 servo pulling the spool forward, the clutch teeth disengage, allowing the spool to freewheel and the gripper to snap closed around a branch. For unperching, the SCS0009 pushes the spool back onto the clutch, where the stronger STS3032 rewinds the cables, forcing the bistable strips back into their straightened state.

This combination of a passive bistable closure with an active clutch-based rewinding mechanism reduces the load on the main servo during perching while providing reliable reset for subsequent use. The design achieves a compromise between strength, speed of actuation, and low mechanical complexity, tailored to the constraints of a 1.2[kg] MAV platform.

V. Branch Detection

For aerial robots tasked with autonomous perching, robust branch detection is crucial. The detection system must not only identify candidate branches but also estimate their three-dimensional (3D) geometry with sufficient precision for downstream trajectory planning. In this work, we systematically explore and compare several approaches to branch detection, ranging from repurposing existing grasp detection networks to custom-designed lightweight convolutional models and modern keypoint-based detectors. Figure 6 gives an overview of the datasets used and the learning-based results.

A. Branch Visual Features

Detecting branches suitable for perching is a challenging problem in natural environments. Branches differ widely in diameter, curvature, texture, and appearance depending on tree species and environmental conditions. Thin branches may appear as elongated silhouettes against the sky, while thicker branches can blend into cluttered, leafy backgrounds. Furthermore, partial occlusions, bark texture, and varying illumination add additional complexity to the task. Prior work indicates that diameters of approximately 4–7[cm] are optimal for MAVs around 1[kg] [4, 8, 17].

B. Dataset Collection & Processing

To train and evaluate branch detection methods, we created our own dataset combining both controlled laboratory and natural forest imagery, dedicated to branch and perch detection. Two complementary environments were used:

- **Cyberzoo dataset:** collected in the TU Delft Cyberzoo facility using a pseudo-tree structure without leaves made from real branches of varying thickness (see Figure 8). This controlled set-

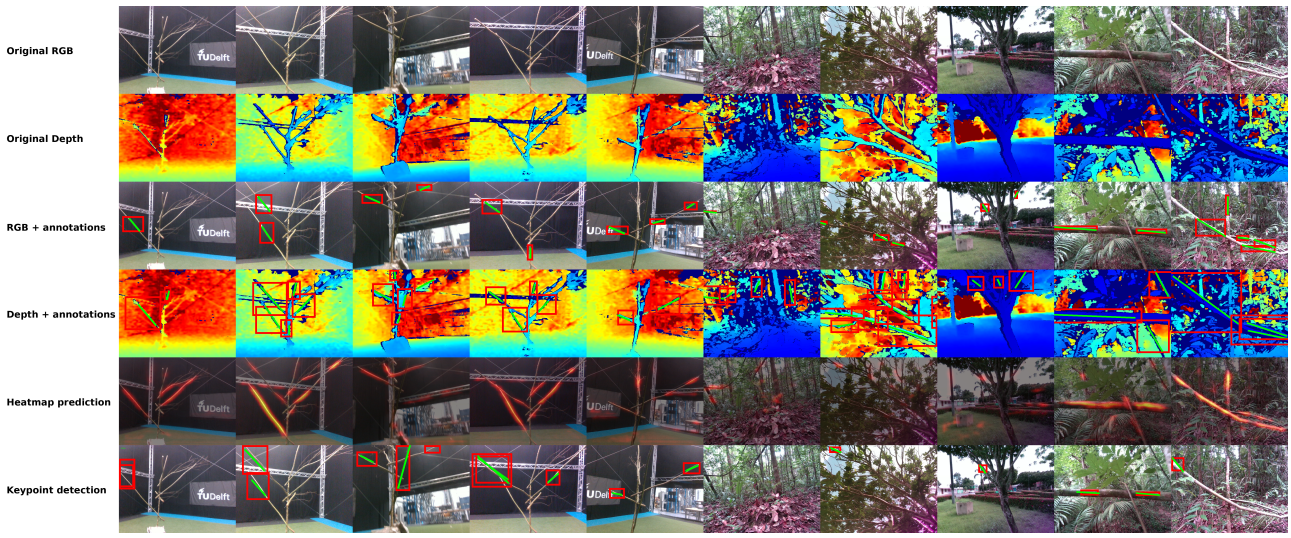


Figure 6: *A random subsample of Cyberzoo (5 columns left) and Rainforest (5 columns right) datasets. The first row displays the original RGB image and the 2nd row the original depth image with heatmap applied for better visualization. Below that are the respective manual annotations (bounding box red, perch line green) of the dataset for the RGB (row 3) and depth (row 4) images. The 5th row shows the heatmap prediction of the Greyloerie CNN ('hot' overlay on RGB image), while the last row shows the keypoint pair detection (green lines, red bounding boxes) of the trained Yolo keypose model.*

ting allowed us to capture clear views of branches and remove confounding background clutter.

- **Rainforest dataset:** collected from drone videos recorded in the Brazilian rainforest, providing natural scenes with dense vegetation, clutter, and challenging lighting.

The dataset contains both RGB and depth images recorded with an Intel RealSense camera. For annotations, each branch was labeled manually with a straight line defined by two endpoints, representing the centerline of a perch, as well as a bounding box covering the local context of the branch. This ensured that models trained on the data would learn not only to localize elongated structures but also to understand their orientation in context. Importantly, perches were selected across a wide range of orientations, from horizontal to vertical, to avoid bias towards particular viewing angles or perch geometries. A subsample space of both datasets is displayed in Figure 6, showing the original images along with their annotations as well as the heatmap and keypose detection.

Annotation statistics. Table 1 summarizes the dataset statistics across environments and modalities.

Table 1: *Annotation statistics for RGB and depth datasets in Cyberzoo and Rainforest environments.*

	Images	Annotations	Resolution	Avg per image
RGB Cyberzoo	411	994	640×480	2.42
RGB Rainforest	346	663	848×480	1.92
Depth Cyberzoo	411	574	848×480	1.4
Depth Rainforest	346	1231	848×480	3.56

The rainforest dataset provided clutter-rich natural conditions, while the Cyberzoo dataset served as a clean baseline. This dual structure allowed us to evaluate generalization across both controlled and real-world scenes.

C. State-of-the-Art Branch Detection

1. *GraspNet-based Perch Detection*

We initially explored whether GraspNet, an RGB-D grasp prediction framework for 6-DoF tabletop manipulation, could be adapted for branch detection [3]. Since branches resemble elongated cylindrical objects, the model produced plausible grasp poses on both indoor test objects and rainforest branch imagery. However, its inference speed of 1-6[s] per frame proved too slow for real-time use on embedded hardware, leading us to favor lighter, branch-specific detection methods instead.

2. *Classical Non-learning Approaches*

We investigated edge detection, Hough transforms, morphological filtering, depth-based segmentation, and contour analysis as lightweight approaches for branch detection. Edge and Hough methods could highlight straight structures but were easily disrupted by curvature, textured bark, or background clutter. Morphological and contour filters sometimes emphasized elongated shapes but required already good preprocessed images, while depth-based segmentation was highly sensitive to missing pixels and IR noise. Although these pipelines ran in real time (30-50[Hz] on laptop CPU), their lack of robustness across natural scenes ultimately motivated a transition to learning-based methods.

3. *Greyloerie: Custom Heatmap CNN*

To directly capture branch-like elongated structures, we developed Greyloerie, a lightweight CNN inspired by heatmap-based pose estimation networks [13, 16]. The shallow encoder–decoder used convolutional lay-

ers with PReLU activations, a single downsampling, and a transposed convolution to predict per-pixel branch likelihood maps. The network remained compact (∼10-20k parameters) while retaining a receptive field suited to elongated features.

Training. Branches annotated as two-point lines were converted to Gaussian heatmaps ($\sigma = 8$) to provide broad supervision signals. Training used MSE loss with geometric augmentations (flips, rotations, crops, zooms), encouraging the model to highlight continuous line structures rather than isolated pixels.

Post-processing and performance. Heatmaps were smoothed, thresholded, skeletonized, and vectorized, with depth fused to reject noisy candidates. This produced clean 2D/3D branch lines, but the reliance on skeletonization and graph extraction added latency, limiting real-time use. Still, Greyloerie robustly identified branches in cluttered rainforest imagery and demonstrated the promise of heatmap-based learning for elongated structures. Without postprocessing the heatmap prediction runs at about 40-60[Hz] on the Jetson Xavier.

4. YOLOv11 Pose Detection and Perch Keypoint Detector

The final and most effective approach used YOLOv11 in its pose/keypoint variant [15], fine-tuned to predict two endpoints per branch. Unlike heatmap-based Greyloerie, this provided end-to-end perch detection without skeletonization or heavy post-processing. We trained a YOLOv11-small model initialized from COCO keypoint weights, using RGB images at 128×128 resolution with two-point annotations. Standard Ultralytics augmentations (flips, rotations, crops, zooms) were applied, while depth-only training yielded inferior results due to sensor noise and annotation inconsistency, making RGB the primary input.

The trained model was deployed as the *detector node*, the entry point of the perception stack. It subscribes to synchronised ($\approx$ 20[ms]) RGB-D frames from the Intel RealSense D435 and runs onboard the NVIDIA Jetson Xavier NX at 7-50[Hz] (avg. 15[Hz]). The detector executes three refinement stages:

1. **Depth validation:** For each endpoint, a circular patch in the aligned depth image is checked for valid-pixel ratio and variance; invalid or noisy patches are rejected. Surviving endpoints are snapped to the nearest valid depth pixel.

2. **3D back-projection:** Validated keypoints (u, v, d) are reprojected into metric coordinates $\mathbf{P}(u, v, d)$ using the camera intrinsics $\mathbf{K}$:

$$\mathbf{P}(u, v, d) = \begin{bmatrix} (u - c_x)\, d/f_x \\ (v - c_y)\, d/f_y \\ d \end{bmatrix}. \tag{1}$$

   Candidate segments are gated by physical plausibility (length range $[0.06,\ 1.40]\,\mathrm{m}$, z-disparity $\leq 1.0\,\mathrm{m}$).

3. **Pose representation:** From 3D endpoints $\mathbf{P}_1, \mathbf{P}_2$ we compute midpoint $\mathbf{m}$, direction $\hat{\mathbf{x}}$, and quaternion $\mathbf{q}_{\text{perch}}^{(\text{cam})}$ in the camera frame:

$$\begin{aligned} \mathbf{m} &= \tfrac{1}{2}(\mathbf{P}_1 + \mathbf{P}_2), \\ \mathbf{d} &= \mathbf{P}_2 - \mathbf{P}_1, \\ \hat{\mathbf{x}} &= \frac{\mathbf{d}}{\|\mathbf{d}\|}. \end{aligned} \tag{2}$$

   Orientation is defined by the minimal rotation from a reference axis $\mathbf{a} = -\mathbf{e}_x$ onto $\hat{\mathbf{x}}$:

$$\theta = \arccos(\mathbf{a}^\top \hat{\mathbf{x}}), \tag{3}$$

$$\mathbf{r} = \frac{\mathbf{a} \times \hat{\mathbf{x}}}{\|\mathbf{a} \times \hat{\mathbf{x}}\|}, \tag{4}$$

$$\mathbf{q}_{\text{perch}}^{(\text{cam})} = \begin{bmatrix} \cos(\theta/2) \\ \mathbf{r}\, \sin(\theta/2) \end{bmatrix}. \tag{5}$$

   Degenerate cases are handled explicitly: if $\hat{\mathbf{x}} \approx \mathbf{a}$ then $\mathbf{q} = [1, 0, 0, 0]^\top$, while if $\hat{\mathbf{x}} \approx -\mathbf{a}$, a π rotation about an orthogonal axis is applied.

Filtering parameters balance recall and reliability, for example, confidence threshold 0.3-0.4, minimum 2D elongation $\sim$ 40[px], depth-valid ratio > 0.7, and adaptive patch radius $0.35\times$ line length. These constraints prune spurious and less suitable detections while preserving good perches.

The detector outputs relative perch candidates in the camera frame, each published as a custom `Perch` message. This message contains 2D endpoints, elongation, and confidence, as well as 3D endpoints, metric elongation, and centre pose (position + quaternion). Multiple detections are grouped in a `PerchList` for use by the perch planner.

Overall, YOLOv11 pose detection offered the best trade-off among all tested approaches: fast enough for on-board use, robust across varied forest imagery, and directly producing usable perch lines. Combined with lightweight depth validation, it enables practical deployment, providing accurate 3D perch geometry for downstream planning. Additionally, it would also allow for the detection of perches using only a small 128x128 camera paired with a single rangefinder.

VI. Kinematics and Control

In this section we outline the coordinate transformations linking camera, body, and global frames, after which we describe the robot arm kinematics that align the gripper orthogonal to the perch. Finally, we detail the offboard control interface for vehicle motion, including the guarded contact-check trigger.

A. Global Transformations

The transformation node converts relative perch detections from the camera frame into the global NED frame used by PX4 and the planner. It subscribes to the perch list, vehicle local position, and vehicle attitude, and its outputs are the perch lists in the global frame.

Pose buffering. To compensate for time differences induced by processing, the node logs a ∼1.5[s] buffer of vehicle state $(\mathbf{p}, \mathbf{q})$ at ∼100[Hz]. For each perch list, it finds the closest state in time to the image stamp, thereby minimizing projection errors due to latency.

Transform and offset. Camera coordinates $(x_{\text{cam}}, y_{\text{cam}}, z_{\text{cam}})$ are first mapped into the NED convention through a fixed axis permutation:

$$\begin{bmatrix} x_{\text{NED}}^{(\text{cam})} \\ y_{\text{NED}}^{(\text{cam})} \\ z_{\text{NED}}^{(\text{cam})} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \end{bmatrix}. \tag{6}$$

This reorients the Intel RealSense convention into the body-fixed NED frame. A static offset vector $\Delta\mathbf{p}_{\text{cam}} = (0.06, 0.011, 0.05)^\top$ m is then applied to account for the camera's mounting displacement relative to the vehicle body center.

Given the vehicle attitude $\mathbf{q}_{\text{veh}}$ and position $\mathbf{p}_{\text{veh}}$ in NED, a perch point $\mathbf{P}_{\text{cam}}$ is transformed into global coordinates as:

$$\mathbf{P}_{\text{glob}} = \mathbf{p}_{\text{veh}} + R(\mathbf{q}_{\text{veh}}) \left(\mathbf{P}_{\text{cam}}^{(\text{NED})} + \Delta\mathbf{p}_{\text{cam}} \right), \tag{7}$$

where $R(\mathbf{q}_{\text{veh}})$ is the 3×3 rotation matrix corresponding to the quaternion.

Orientation quaternions are composed analogously:

$$\mathbf{q}_{\text{glob}} = \mathbf{q}_{\text{veh}} \otimes \mathbf{q}_{\text{cam}\to\text{NED}} \otimes \mathbf{q}_{\text{perch}}^{(\text{cam})}, \tag{8}$$

with $\otimes$ denoting quaternion multiplication and $\mathbf{q}_{\text{cam}\to\text{NED}}$ representing the static camera-to-NED rotation.

B. Robot Arm Kinematics

The perching mechanism uses three actuators: a gearbox servo rotating the arm (0–90°), a micro-servo controlling the clutch for release of loaded gripper, and a spool servo for rewinding the gripper. The commanded arm angle is $\theta \in [0, 90]^\circ$, with 0° as the parked position. Because of the gearbox reduction, servo turns map linearly to θ, but initialization at $\theta = 0^\circ$ corresponds to a negative servo index. The mapping is

$$n_{\text{turns}} = k_{\text{deg}\to\text{turn}}\, \theta, \tag{9}$$

$$targetpos = n_{\text{turns}} + center\_index, \tag{10}$$

where n_{turns} is the number of servo turns, $k_{\text{deg}\to\text{turn}}$ [turn/deg] is the conversion factor from arm angle to servo rotations based on gearbox ratio, and center_index is the neutral servo index around which motion is referenced.

C. MAV Offboard Control

The offboard server exposes a ROS 2 `NavigateToPose` action for commanding PX4 in offboard mode. It accepts goals in the global NED frame, translates them into continuous `TrajectorySetpoint` streams, and maintains the `PX4 OffboardControlMode` through heartbeat updates.

Goal handling and primitives. Special primitives are parsed from the goal's message id: `"takeoff"` maintains current x, y, ψ while rising to commanded z; `"land"` forwards a PX4 land command; `"contact_check"` initiates guarded approach. Standard navigation extracts (x, y, z, ψ) from the goal pose. Each goal is subject to a timeout and convergence check.

Setpoint streaming and convergence. For a goal pose $\{\mathbf{p}_g, \psi_g\}$, the node streams position setpoints $\mathbf{s}(t) = [x_g, y_g, z_g]$ and yaw ψ_g. Convergence is detected if

$$|x - x_g|,\ |y - y_g|,\ |z - z_g| < \varepsilon_p, \tag{11}$$

$$|\text{wrapTo}_\pi(\psi - \psi_g)| < \varepsilon_\psi, \tag{12}$$

with $\varepsilon_p = 0.05$[m] and $\varepsilon_\psi = 0.2$[rad]. The default loop period is 0.1[s] (10[Hz]).

Contact-check guarded approach. For perch approaches, the `"contact_check"` mode evaluates buffered signals over a $T_w = 3$[s] window:

1. **Position plateau:** let $\mathbf{p}_0$ and $\mathbf{p}_N$ be the first/last buffered positions. Contact-like stall is

$$\|\mathbf{p}_N - \mathbf{p}_0\| \ < \ \delta_p, \quad \delta_p \approx 0.05 \text{ [m]}. \tag{13}$$

2. **Motor divergence:** compute the mean per-motor outputs $\bar{m}_i$ over the window and the maximum pairwise deviation

$$\Delta_m \ = \ \max_{i,j\le4} |\bar{m}_i - \bar{m}_j| \ > \ \tau_\Delta, \quad \tau_\Delta \approx 0.25. \tag{14}$$

3. **Sustained high thrust:** from the per-sample max output $m_{\max}(k)$, require

$$\frac{1}{N}\sum_{k=1}^{N} m_{\max}(k) \ > \ m_{\text{hover}} + \tau_h, \tag{15}$$

$$m_{\text{hover}} \approx 0.46,\ \tau_h \approx 0.15. \tag{16}$$

If any trigger activates, the server overwrites the goal with the current pose and holds, ensuring safe termination without external stop commands.

VII. Planning & Control

The `nagapie_perch_planner` node serves as the central decision-making component, implemented as a finite state machine (FSM). It sequences perception, trajectory execution, and actuator control into a structured perching manoeuvre. In addition, the node implements logging functionality for post-flight analysis, recording all state transitions, issued commands, and perception results.

State Machine Design. The FSM is composed of a finite set of discrete states $\mathcal{S}$ representing mission

Table 2: *Offboard server: parameters and effects.*

Parameter	Value	Effect
Setpoint loop (typ.)	10–50 [Hz]	Higher rates improve OFFBOARD robustness; 10 Hz used during development.
Position tolerance ε_p	0.05 [m]	Convergence threshold; tighter ⇒ longer settle.
Yaw tolerance ε_ψ	0.2 [rad]	Avoids oscillation near goal heading.
Goal timeout	60 s	Aborts non-converging goals.
Contact window T_w	3 [s] @ 100 [Hz]	History time length for plateau/ divergence/ high-thrust checks.
Plateau threshold δ_p	0.05 [m]	Triggers contact check if displacement over T_w is $< \delta_p$.
Divergence threshold τ_Δ	0.25 (norm.)	Large mean inter-motor spread indicates contact (tilting due to contact-induced moment).
Hover baseline m_{hover}	0.46 (norm.)	Approx. 50% hover; used for high-thrust test.
High-thrust margin τ_h	0.15 (norm.)	Sustained >0.61 indicates contact.

phases, and transitions $\mathcal{T}$ driven by conditions on perception and execution outcomes. A state $s_k \in \mathcal{S}$ is represented by the tuple

$$s_k = \{\mathbf{p}, \mathbf{q}, \sigma\}, \tag{17}$$

where $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{q} \in SO(3)$ denote the vehicle's pose in the NED frame, and σ is the discrete mode (e.g. `scan`, `verify`, `align`, `approach`, `contact`, `grasp`, `perched`). State transitions are governed by

$$s_{k+1} = f(s_k, o_k, u_k), \tag{18}$$

where o_k are perception outcomes (e.g. perch list validity, depth checks) and u_k are control action results (success, timeout, or contact).

Mission Flow. The FSM proceeds through the following major stages:

1. **Scan:** perform a 360°yaw sweep to collect perch candidates. A perch is selected based on perch elongation, relative distance, and orientation.

2. **Verification:** navigate to a frontal verification pose at distance d_a, re-run depth checks, and confirm suitability.

3. **Alignment:** rotate the perching arm to align the gripper orthogonal to the branch orientation quaternion $\mathbf{q}_{\text{perch}}$ (cf. Equation 5).

4. **Approach:** execute a straight trajectory from beneath the perch, orthogonal to the perch along the rotated arm direction, maintaining heading.

5. **Contact-check:** engage guarded approach (subsection C); upon stall detection, hold position.

6. **Grasp:** trigger the bistable gripper closure.

7. **Perched / Re-perch:** remain attached or initiate release and return-to-flight sequence.

Table 3: *Planner/Logger node: parameters and effects.*

Parameter	Value	Effect
Hover height	1.7 [m]	Nominal altitude used for initial scanning during testing.
Perch merge radius	0.5 [m]	Merges nearby perch detections into one candidate to avoid duplicates.
Perch max angle	60°	Rejects candidates steeper than threshold; allows perch geometry selection.
Approach distance d_a	1.0 [m]	Stand-off distance for initial verification/approach pose.
Arm offset	0.3 [m]	Offset from perch to gripper during orthogonal approach; prevents premature contact.
Arm length	0.23 [m]	Effective reach from rotation axis to gripper; used for contact pose calculations.
Rot. axis offset (x, y, z)	(-0.034, 0.01, 0.067) [m]	Position of arm rotation axis relative to body center; required for precise arm–vehicle alignment.
Soft touchdown Δz	0.15 [m]	Additional downward offset applied after contact to ensure secure clasping.
Release wait	7.0 [s]	Hold time after gripper release before returning to flight; ensures full reset.
Perch discard radius	1.0 [m]	Prevents selecting new perches too close to the previous one during reperching, promoting diversity of attempts.

Pose Calculations. Verification and pre-grasp approach are computed relative to the perch midpoint $\mathbf{m}$ and orientation $\hat{\mathbf{x}}$. The approach direction $\mathbf{n}$ is defined as the projection of the global up vector $\mathbf{u} = (0, 0, -1)^\top$ onto the plane orthogonal to $\hat{\mathbf{x}}$:

$$\mathbf{n} = \frac{\mathbf{u} - (\mathbf{u} \cdot \hat{\mathbf{x}})\,\hat{\mathbf{x}}}{\|\mathbf{u} - (\mathbf{u} \cdot \hat{\mathbf{x}})\,\hat{\mathbf{x}}\|}. \tag{19}$$

The corresponding rotation-axis approach position is

$$\mathbf{p}_{\text{rot}} = \mathbf{m} - d_a\,\mathbf{n}, \tag{20}$$

with d_a the standoff distance (arm offset + arm length). The commanded vehicle waypoint $\mathbf{p}_{\text{body}}$ is obtained from $\mathbf{p}_{\text{rot}}$ by adding fixed actuator offsets from the manipulator's rotation axis to the body frame.

The vehicle yaw is chosen depending on the flight phase: during the approach phase, the yaw is set such that the camera faces the perch center (Eq. (21)); during the grasp phase, it aligns with the perch's x-axis, flipped if pointing downward (Eq. (22)).

$$\psi_{\text{approach}} = \text{atan2}(m_y - y,\, m_x - x), \tag{21}$$

$$\psi_{\text{grasp}} = \text{atan2}(\hat{x}_y,\, \hat{x}_x). \tag{22}$$

Logging. Each FSM transition appends a log entry containing:

- timestamp and current state σ,
- selected perch ID and properties,
- commanded vehicle pose and arm angle,
- offboard action result (success/failure).

VIII. Autonomous Perching

This section reports on the experimental validation of the proposed perching pipeline. After confirming

the correct operation of individual subsystems during development, we evaluated the fully integrated system through progressively more demanding tests. The aim was not only to demonstrate autonomous perching capability, but also to assess robustness under varying geometric and visual conditions. These experiments highlight both the achievements and the difficulties of enabling a 1.2[kg] quadrotor to reliably detect, approach, and attach to natural branches using only onboard sensing and control.

A. Testing Setup

Simulation Environment Before flight experiments, a software-in-the-loop (SITL) setup combined PX4 and ROS 2 to reproduce the full perception and control pipeline. Known inputs and recorded rosbags were replayed to verify closed-loop behaviour, enabling consistent debugging while reducing hardware risks during early development.

Artificial Forest Environment Real-world experiments were conducted in the TU Delft Cyberzoo arena, where an artificial tree was built to emulate forest conditions under controlled settings. Detachable wooden branches with diameters between 2.5–7[cm] were mounted at varying inclinations from horizontal to vertical, providing diverse perching targets with natural curvature and bark texture. This modular design enabled systematic testing of approach strategies across different perch geometries. To increase the ecological validity of the setup, the background was covered with artificial vegetation and large printed forest-scene banners. This ensured a visual environment distinct from the training data, thereby mitigating overfitting effects and providing a more representative challenge for outdoor deployment. Figure 7 illustrates the artificial tree with multiple branch configurations.

(a) (b)

Figure 7: *Controlled testing environment with the constructed 'tree'. The branches are easy to swap and adjustable for different angles. For CNN training and initial flight testing, the leafless tree was used (a). Green artificial foliage and forest scene banners are added in the background to add complexity on the right image (b).*

Evaluation Metrics The evaluation of perching manoeuvres combined one quantitative and several qualitative measures. The sole quantitative metric was the *success rate*, defined as the ratio of completed perches to attempted trials for a given configuration. Since successful configurations proved perfectly reproducible, a single success confirmed feasibility. It should be noted, however, that branch detection depends on the convolutional neural network used for perch identification. Thus, the number and type of available perches in a scene are influenced by environmental complexity and the training data distribution. Beyond this, qualitative observations assessed system capabilities under broader conditions, including the *range of branch orientations* (from horizontal to near-vertical), the *number of unique perches detected and utilised*, and the MAV's ability to *maintain attachment*, *unperch reliably*, and *re-perch within the same mission*. Together, these measures characterised the robustness, repeatability, and operational envelope of the perching pipeline.

B. Baseline Demonstrations

To establish a reference, the full control and actuation pipeline was first flight-tested using ground-truth perch locations. OptiTrack markers were attached to branches, providing precise position and orientation data that were streamed into the planning stack. Under this setup, the vehicle consistently achieved successful perches at branch angles of 0°, 30°, and 60°. These flights also confirmed the correct geometric offset of the gripper in the control system, which clasped reliably at the intended location without requiring contact feedback. This stage validated both the actuator design and the approach trajectory generation.

C. Static Scene Verification

The next step evaluated branch detection and global transformation accuracy in a controlled setting. The vehicle was mounted on a pedestal facing branches at varying distances (Figure 8). Detections were robust and consistently aligned with the global coordinate frame. A depth calibration error was initially observed, leading to biased distance estimates; this was corrected through re-calibration of the stereo camera. Overall, these tests established that the perception pipeline can deliver correct perch pose transformations under static conditions.

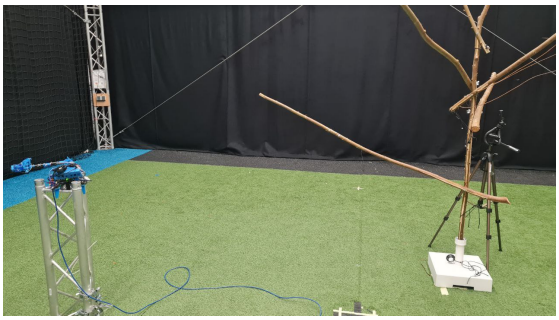


Figure 8: *Static testing configuration with the drone mounted on a pedestal in front of branches at varying distances. Used to validate detection and transformation accuracy.*

D. Autonomous Perching on a Leafless Tree

Fully autonomous perching trials were then conducted on a single artificial tree without foliage or added vegetation to minimize clutter and match the CNN training data. For the first tests, the filter-

ing parameters were set to favour horizontal and nearby branches. In the initial flights, the system detected perches but rejected them upon close verification, subsequently returning to the home position. This confirmed the proper functioning of the verify/reject and abort routines. After relaxing the filter thresholds, the first fully autonomous perch was achieved: the drone initially selected a branch lying on the ground (rejected after inspection due to lacking depth disparity) before successfully grasping a horizontal branch. A follow-up flight with a branch angled at 30° exposed a failure mode where the vehicle pressed against the branch without achieving its global target setpoint, resulting in excessive thrust and instability. This prompted the implementation of the contact detection routine, which monitors thrust increase, lack of positional change, and motor output divergence due to tilt. With this feature, subsequent flights achieved successful perches at angles up to 45°. At 60°, grasping was possible but sliding occurred on thinner branches. After adding a passive spacer arm these problems were resolved and perching up to the vertical (90°) was achieved.

E. Perching on Foliated and Cluttered Scenes

To evaluate performance in more realistic and visually complex settings, artificial foliage was added to the original 'blank' tree setup and later embedded into a cluttered scene composed of additional plants and printed forest banners. This transition introduced the visual variability typical of natural habitats, where background textures, occlusions, and non-branch structures create significant challenges for vision-based detection.

With foliage added, the CNN-based detector continued to provide stable branch candidates despite the altered colours and textures present. Several successful autonomous perches were performed, demonstrating that the perception and filtering pipeline generalizes beyond the bare training geometry. The positive results on the foliated tree already indicated resilience to changes in appearance, an essential requirement for real deployment in forested environments.

The most demanding experiments were conducted in this cluttered setup with multiple artificial plants and background forest banners. In this setting, the drone had to discriminate suitable perches from visually similar objects. With this working well, the next step was to demonstrate autonomous take-off from a successful perch. With the correct setpoint ordering mirroring the approach phase, the system achieved stable perch–unperch–land cycles. A representative example is shown in Figure 9, where overlaid video frames illustrate the full sequence from approach, grasp, and release to recovery in hover.

Next to demonstrating perching at branches angled up to 60°we also demonstrated vertical perching and takeoff on the stem of a tree. Adding a passive distance spacer at the bottom of the drone counteracts the moment rotating the drone off from the gripped branch (see Figure 9e).

The final series of trials extended autonomy to repeated operations within a single flight. Here the drone first perched successfully, detached, returned to above the home point, and then selected, verified and perched at a second perch on a different branch. To our knowledge, this "hop" between perches, performed without external intervention, constitutes the first demonstrations of a quadrotor autonomously executing multiple perch–unperch–reperch cycles in a forest-like environment. Importantly, these flights confirm that the branch selection, approach planning, grasping, release, and recovery routines can operate robustly in succession, even under significant visual clutter.

Overall, these cluttered-scene experiments provide the strongest evidence for the robustness and maturity of the system. They show that beyond isolated successes, the proposed pipeline enables repeated and flexible perching behaviors in conditions approximating real-world forest environments, a critical step towards ecological monitoring applications.

F. Failures

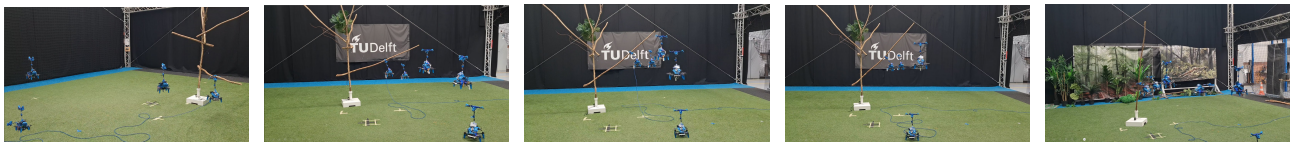
Across the test phase, the main causes of failure were: (i) erroneous depth estimates, (ii) incorrect detections, (iii) drift induced by arm rotation during approach, and (iv) sliding on thin, strongly inclined branches. Nonetheless, once clasped, the gripper provided stable attachment, even when only partially engaged. An overview of flight outcomes is given in Table 4. Notably, even though cable sway and arm inertia induced drift during scanning, the pose buffering ensured accurate association between detections and flight poses.

Table 4: *Test flight conditions and results.*

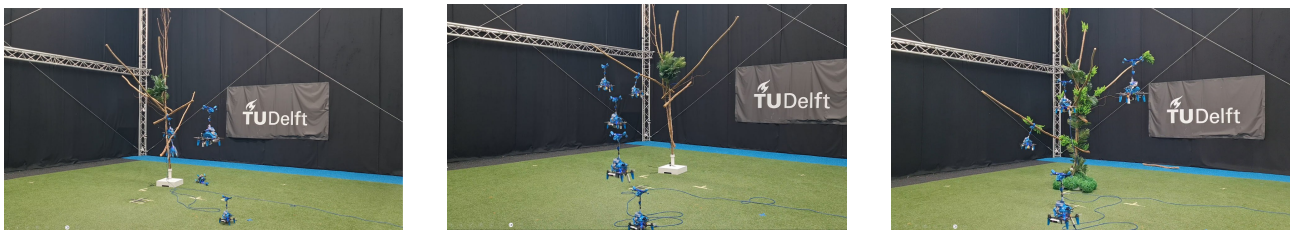
Test condition	Outcome	Notes / Failure modes
Ground-truth OptiTrack	2 successes	All tested angles (0°, 45°).
Bare tree, 0°	1 success	/
Bare tree, 25°	1 success, 2 fails	Depth error; branch push.
Bare tree, 45°	1 success, 1 fail	Depth error.
Bare tree, 60°	1 success, 1 fail	Gripper cable stuck; sliding after perch and crash.
Bare tree, new branch positions	2 success, 1 fail, 1 aborted	Too close to stem due to arm-rotation induced drift (aborted); depth error.
Green tree	1 success, 1 fails	Depth error.
Green tree + forest background	6 successes, 1 fail	Final demo with multi-branch hopping.
Green vertical stem	2 successes	After adding passive spacer and code handling of vertical perch.

G. Key Insights

The experiments demonstrate that RGB-based detection combined with simple 3D depth probing is sufficient to achieve autonomous perching without full 3D scene understanding. Training data that emphasizes accessible branches proved effective in guiding the CNN towards feasible perches, naturally avoiding obstructed or occluded locations that could lead to collisions in flight. Remaining limitations include



(a) *0° Leafless* (b) *25° Leafless* (c) *45° Leafless* (d) *60° Leafless* (e) *90° Leafless*



(f) *Different branch A* (g) *Different branch B* (h) *35° Foliated*

(i) *45° added forest background* (j) *45° perch→unperch* (k) *45° perch→unperch→perch→unperch*

Figure 9: *Perching across environments and conditions.* ***(a–e)*** *Leafless tree, varying branch inclinations (0°, 25°, 45°, 60°, 90°), demonstrating approach and attachment across a wide range of angles.* ***(f–g)*** *Leafless tree with different branches, showing generalization beyond a single branch.* ***(h–k)*** *Foliated tree and added forest backdrops: (h) 35° on foliated tree; (i) 45° with added forest background; (j) 45° perch–unperch–land sequence; (k) 45° perch–unperch–perch-unperch-land sequence on a new branch with added forest background.*

sensitivity to depth errors and mechanical slippage on steep, thin branches. The results constitute a complete autonomous perching cycle: detection, verification, approach, grasp, unperch, and reperch.

IX. Future Research

This work demonstrated that a lightweight MAV equipped with a rotating arm and CNN-based perception can autonomously detect, approach, and perch on branches in structured indoor and semi-natural environments. The successful operation in cluttered and foliated scenes, as well as repeated autonomous hops between perches, indicates that the core perception and control pipeline is viable. At the same time, the transition towards fully outdoor deployment highlights several areas where further development would enhance robustness, efficiency, and ecological applicability.

First, improvements in close-range sensing could substantially increase reliability. During hovering below the perch and in the final pre-grasp phase, initial stereo depth estimates show to be irregular. Lightweight supplementary sensors could mitigate these errors. Examples include line-scan cameras for high-contrast branch detection [18], tactile whisker-like appendages for contact-based localization in dense foliage [2], or a single rangefinder aligned with the arm that allows lateral adjustment until the branch is detected at expected range.

Second, accurate self-pose estimation remains a key challenge in forest environments where GPS is degraded. Visual-inertial odometry (VIO) systems have been shown to provide robust onboard localization [11]. While not strictly necessary if close-range sensors are improved, integrating VIO would increase overall reliability and allow more complex autonomous behaviours in outdoor settings.

Third, the current computing pipeline could be optimized towards lighter and more accessible platforms. The tests revealed that CNN inference speed did not differ significantly between CPU and GPU execution on a laptop, suggesting that embedded alternatives such as a Raspberry Pi-class computer could suffice. This would reduce mass and cost, facilitating development.

Developing a detection and planning pipeline based on a single RGB camera coupled with a rangefinder would reduce complexity and power consumption. However, this would require new hardware-specific perception strategies.

Better integrating the arm kinematics into the flight control system would prevent arm motions inducing backwards drift. Incorporating the arm's dynamics into the state-space controller would reduce drift and improve stability during perching manoeuvres.

Mechanical redesign also offers potential improvements. Relocating the servo and clutch closer to the vehicle's centre of gravity and transmitting actuation through a lightweight cable system would reduce the inertial load on the top-mounted gripper. A lighter end effector would lessen the effect of arm motion on

body dynamics and enhance energy efficiency.

Finally, full outdoor testing will require obstacle detection and avoidance. Integrating vision- or range-based avoidance modules would allow the MAV to approach candidate branches safely in dense vegetation, enabling deployment in realistic ecological monitoring scenarios.

X. Acknowledgements

I would like to express my sincere gratitude to Salua Hamaza, who has been incredibly patient and always found the time to discuss not only thesis-related topics but also our shared interest in applying technology for nature conservation. I am also deeply thankful to Liming Zeng, whose insights were invaluable when considering design choices. My thanks further extend to all members of the MAVLab, both staff and students, for their collective support—whether through knowledge, spare parts, or simply encouragement along the way. Finally, I wish to thank my family, whose unwavering support has carried me through both this thesis and my projects beyond the university.

References

[1] K. Anderson and K. J. Gaston. Lightweight unmanned aerial vehicles will revolutionize spatial ecology. *Frontiers in Ecology and the Environment*, 11(3):138–146, 2013. doi: 10.1890/120150.

[2] L. Andresen, E. Aucone, and S. Mintchev. Whisker-based haptic perception system for branch detection in dense vegetation. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 911–916. IEEE, 2022. doi: 10.1109/ROBOSOFT54090.2022.9762143.

[3] U. Asif, J. Tang, and S. Harrer. Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 4875–4880, 2018.

[4] C. Geckeler, E. Aucone, Y. Schnider, A. Simeon, J.-P. von Bassewitz, Y. Zhu, and S. Mintchev. Learning occluded branch depth maps in forest environments using rgb-d images. *IEEE Robotics and Automation Letters*, 9(3):2439–2446, 2024. doi: 10.1109/LRA.2024.3355632.

[5] M. A. Graule, P. Chirarattananon, S. B. Fuller, N. T. Jafferis, K. Y. Ma, M. Spenko, R. Kornbluh, and R. J. Wood. Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion. *Science*, 352(6288):978–982, 2016. doi: 10.1126/science.aaf1092.

[6] S. Kirchgeorg, B. Benist, and S. Mintchev. Soft gripper with adjustable microspines for adhering to tree branches. In *Lecture Notes in Networks and Systems 530*. ETH Zurich, Springer, 2022. doi: 10.1007/978-3-031-15226-9_9. URL `https://doi.org/10.3929/ethz-b-000574190`. Funded by SNF under grant 186865 - CYbER - CanopY Exploration Robots.

[7] M. KleinHeerenbrink, L. A. France, C. H. Brighton, and G. K. Taylor. Optimization of avian perching manoeuvres. *Nature*, 607:91–96, 2022. doi: 10.1038/s41586-022-04861-4.

[8] S. McGinley. Vision-guided quadrotor perching on imperfectly cylindrical structures. Master's thesis, Delft University of Technology, 2023. URL `http://resolver.tudelft.nl/uuid:21b4203a-abbb-47d7-bbd5-df042d8d7b53`.

[9] OpenAI. Chatgpt, 2024. URL `https://www.openai.com/chatgpt`. Used for rewording and latex code completion.

[10] J. L. Paneque, J. R. Martínez-de Dios, A. Ollero, D. Hanover, S. Sun, Á. Romero, and D. Scaramuzza. Perception-aware perching on powerlines with multirotors. *IEEE Robotics and Automation Letters*, 7(2):3077–3084, April 2022. doi: 10.1109/LRA.2022.3145514.

[11] T. Pritchard, S. Ijaz, R. Clark, and B. B. Kocer. Forestvo: Enhancing visual odometry in forest environments through forestglue. *IEEE Robotics and Automation Letters*, 10(6), 2025. doi: 10.1109/LRA.2025.3557738.

[12] W. R. T. Roderick, M. R. Cutkosky, and D. Lentink. Bird-inspired dynamic grasping and perching in arboreal environments. *Science Robotics*, 6(63):eabj7562, December 2021. doi: 10.1126/scirobotics.abj7562. Accessed from Delft University on November 16, 2023.

[13] R. Silva, J. M. Junior, L. Almeida, D. Gonçalves, P. Zamboni, V. Fernandes, J. Silva, E. Matsubara, E. Batista, L. Ma, J. Li, and W. Gonçalves. Line-based deep learning method for tree branch detection from digital images. *International Journal of Applied Earth Observation and Geoinformation*, 110:102759, 2022. doi: 10.1016/j.jag.2022.102759.

[14] J. Thomas, M. Pope, G. Loianno, E. W. Hawkes, M. A. Estrada, H. Jiang, M. R. Cutkosky, and V. Kumar. Aggressive flight with quadrotors for perching on inclined surfaces. *Journal of Mechanisms and Robotics*, 8(5):051007–1, October 2016. doi: 10.1115/1.4032250. Manuscript received September 21, 2015; final manuscript received December 3, 2015; published online May 4, 2016. Assoc. Editor: James Schmiedeler.

[15] Ultralytics. Ultralytics documentation: Pose estimation tasks. `https://docs.ultralytics.com/tasks/pose/`, 2024. Accessed: 2025-09-02.

[16] H. Wan, Z. Fan, X. Yu, M. Kang, P. Wang, and X. Zeng. A real-time branch detection and reconstruction mechanism for harvesting robot via convolutional neural network and image segmentation. *Computers and Electronics in Agriculture*, 192:106609, 2022. doi: 10.1016/j.compag.2021.106609.

[17] L. Zheng and S. Hamaza. Albero: Agile landing on branches for environmental robotics operations. *IEEE Robotics and Automation Letters*, 9(3):2845–2852, 2024. doi: 10.1109/LRA.2024.3355632.

[18] R. Zufferey, J. Tormo-Barbero, D. Feliu-Talegón, S. R. Nekoo, J. Á. Acosta, and A. Ollero. How ornithopters can perch autonomously on a branch. *Nature Communications*, 13:7713, 2022. doi: 10.1038/s41467-022-35356-5. Accessed on November 16, 2023.