Part IV

Validation and Field Tests

Chapter 7

Industrial Applications

One of the main objectives of this thesis was to propose a reference architecture, later converted into an actual software framework, that allows the easy development of aerial robotic systems for a wide range of applications. One relevant field is industrial applications, where the system must be robust to ensure correct operation in industrial facilities. Validating the capabilities of the proposed framework in these scenarios is crucial for detecting possible weaknesses.

In this chapter, we present two industrial inspection applications—a windmill inspection and a photovoltaic power plant inspection—developed using Aerostack2, demonstrating its versatility and adaptability to each specific use case.

7.1 Photovoltaic Panel Inspection

7.1.1 Introduction

The transition to sustainable energy requires innovative solutions to enhance the efficiency of renewable sources. Solar photovoltaics (PV) has been the fastest-growing power generation technology worldwide over the past decade (EUE, 2023) and plays a central role in achieving climate neutrality.

While PV adoption offers significant benefits, it also presents challenges, particularly in optimizing plant performance and integrating with the grid. A major hurdle is the monitoring and maintenance of large-scale PV plants, where manual inspections are time-consuming and inefficient. To address this, drone fleets equipped with color and thermal cameras provide an effective and scalable solution for automated defect detection (Colaprico et al., 2018).

This application relies on UAV-based inspection of operational PV plants, using multiple drones to survey designated panel rows simultaneously and collect georeferenced thermal and RGB images, which can later be processed to detect defects within the plant. The approach leverages a georeferenced map to define inspection paths, ensuring systematic coverage.

7.1.2 Hardware Setup

For this inspection, two different UAVs were used to survey the photovoltaic (PV) plants: the DJI M300, utilized in the single-UAV setup, and the DJI M350, deployed alongside the M300 in the swarm setup. Each UAV was equipped with an NVIDIA Xavier AGX board running the software pipeline onboard each one. Additionally, both drones were fitted with a DJI H20T camera, which integrates an RGB camera and an infrared (IR) sensor to capture both image types simultaneously.

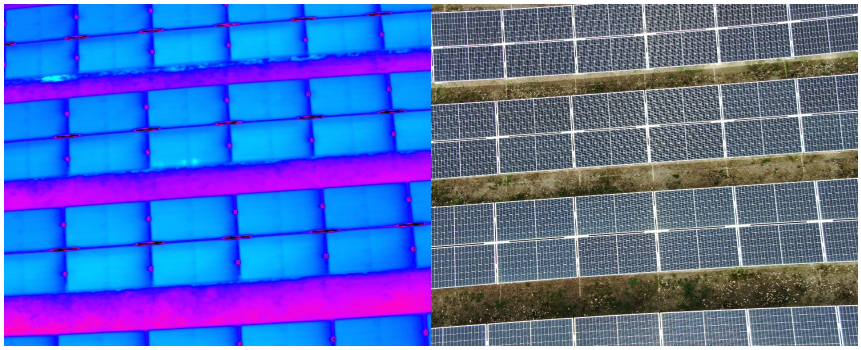


Figure 7.1: Image captured during the inspection: the left side corresponds to the IR image, while the right side shows the RGB image.

7.1.3 Software Architecture

For this application, the *Aerostack2* framework has been used. Figure 7.2 summarizes the different components used for this operation.

The components used in each layer are as follows:

- **Sensor-Actuator Interface:** For this application, the DJI Aerial Platform has been used. The M300 was controlled using the DJI OSDK version of this platform, while the M350 required the use of the newer DJI PSDK firmware. This difference was mitigated by using two different implementations of the Aerial Platform component.

- **Motion Controller:** For this application, we do not need to perform complex maneuvers, so the PID controller plugin has been used.

- **State Estimator:** To ensure very precise geopositioning of the images obtained, an RTK-GPS system has been used. The DJI drone integrates all this information, providing the software framework with an accurate estimation. The `ExternalEstimation` plugin was used.

- **Behaviors:** All the basic motion behaviors, such as *Takeoff*, *Land*, or *Follow Path*, have been used. Additionally, a *Point Gimbal* behavior has been used to ensure that the images are taken parallel to the panels.

- **Plan Execution:** For this application, the plan for each of the drones is generated before the start. Once the plan is verified and accepted by the user, it is loaded for each drone and interpreted using the *Plan Interpreter*.

- **Communications:** In order to limit the bandwidth transmitted, each drone has its own internal communication channel and interacts with the rest of the agents through an external communication channel. In this application, we use a Zenoh Router, which allows us to

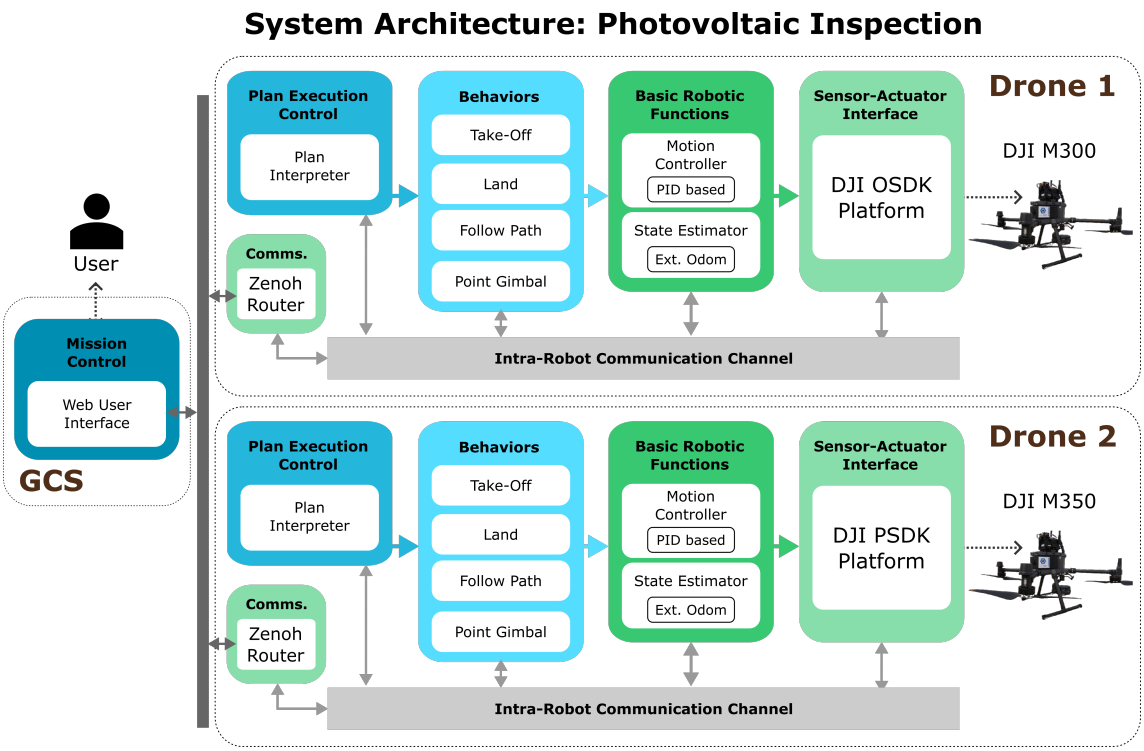


Figure 7.2: Overview of the system architecture used for the real photovoltaic inspection. Each drone has its own internal communication channel, which it uses to control its behavior. The relevant operational information is sent to the user through an external communication channel. The communication module is responsible for routing the required data between both channels.

define which messages will be transmitted, the protocol to use, and the network architecture. In this case, we use a star architecture with the Ground Control Station (GCS) in the middle.

- **Mission Planning:** The mission has been planned using the Web GUI, allowing representation of the total area to be covered and configuration of different mission parameters, such as flight altitude and inspection speed. The mission plan can then be automatically generated, loaded, launched, monitored, and controlled through this interface.

7.1.3.1 Simulated Environment

Before deploying the system in a real power plant, its development and tuning were conducted in simulation.

To simulate a photovoltaic plant, a 2D georeferenced model was developed in Gazebo, serving as the background floor. Additionally, a photorealistic environment was created in the Unity simulator for image capture purposes. To integrate the simulation with ROS 2, Flightmare (Song et al., 2020) was used (see Figure 7.3).

7.1.4 Results

The real-world experiments were conducted at the Repsol Technology Lab in Madrid. These facilities included different segments of photovoltaic panels around buildings. For this experiment, a subset of the panel rows was selected for testing. Video `https://vimeo.com/1002821624` demonstrates the complete process, from mission planning using the Web GUI to subsequent monitoring.

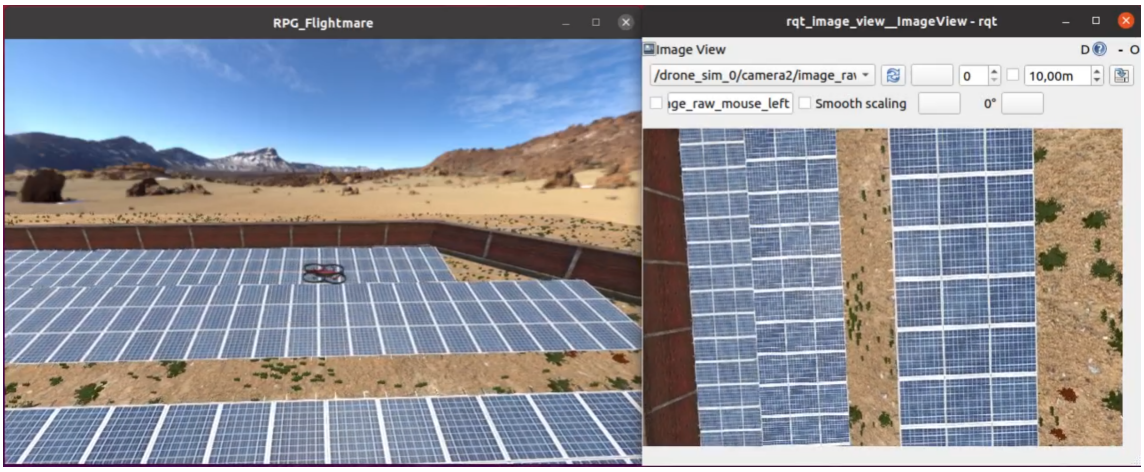


Figure 7.3: Photorealistic simulation of the photovoltaic plant used during the development process.

During the experiments, the system successfully inspected the designated areas autonomously five times at different altitudes, requiring human intervention only for mission definition.

From these experiments, we have extracted key performance indicators (KPIs) that can be used to compare this method with others and guide future improvements to the system:

1. **Inspection Time:** Two drones can cover 1 ha in an average of 13 minutes and 45 seconds.
2. **Non-Valid Images:** Approximately 12% of the gathered images are unsuitable for inspection due to blur or illumination issues. However, due to the redundancy in the images taken, the subsequent inspection can still be performed.
3. **Human Intervention Time:** Each mission requires an average of 7 minutes of operator intervention for setup and launch.

These results highlight the efficiency and potential of UAV-based inspection systems for photovoltaic plants, paving the way for further optimizations and broader adoption in industrial applications.

Figure 7.4: DJI M300 flying over the photovoltaic panels at the Repsol Technology Lab in Madrid.

7.2 Windmill Inspection

7.2.1 Introduction

The global shift towards renewable energy has led to a significant increase in wind turbine installations, making efficient maintenance practices increasingly important.

Regular inspections of wind turbines during normal operation are essential to maximizing their efficiency, lifespan, and reliability. Drones offer a highly effective and safe solution for conducting these operational inspections. Equipped with high-resolution cameras, they can quickly assess turbine blades, towers, and nacelles, capturing detailed information without exposing maintenance personnel to hazardous conditions. Their maneuverability allows drones to reach difficult-to-access areas, making them a cost-effective and efficient alternative to traditional manual inspection methods.

The objective of this application was to inspect a wind turbine during its regular operation. This case study is particularly hazardous, as both the blades and the nacelle are rotating. The nacelle is connected to the rotor, which drives blade rotation, and to the tower, which rotates along its z-axis.

The mission involves following multiple inspection points linked to the wind turbine rotor. These points are calculated based on various inputs, including blade length, intrinsic camera parameters, and the required safety inspection distance. This project was made for the company Aeromedia S.L., and the developments led into the creation of a spin-off of this company called Generadron S.L.

7.2.2 Hardware Setup

For this inspection, the DJI M300 was used, equipped with an NVIDIA Jetson Nano board running the software pipeline onboard. Additionally, each drone carried a payload consisting of a high-resolution camera and a LiDAR sensor, which synchronized the trigger with the movement of the rotating blade.

An external system (*Windmill Observer)* provided both prior and real-time data of the wind turbine. The prior data included the WGS84 coordinates of the turbine base, blade length, and rotor height. The real-time data consisted of the rotor's orientation, given as an azimuth (horizontal angle from north). This real-time information was transmitted to the drone through the remote controller.

7.2.3 Software Architecture

For this application, the *Aerostack2* framework was used. Figure 7.5 summarizes the different components involved in this operation.

The components used in each layer are as follows:

- **Sensor-Actuator Interface:** The DJI Aerial Platform with the DJI OSDK was used for this application. In addition to providing sensor and actuator data, this platform allows communication with the remote controller, enabling the transmission of mission status and real-time windmill yaw information.

- **Motion Controller:** Since complex maneuvers are not required, the PID controller plugin was used. The mission is planned with poses relative to the wind turbine rotors, requiring the nacelle's GPS coordinates and azimuth to be transformed into Cartesian positions. When

this information reaches the UAS, it is converted into a pose and integrated into the transformation tree relative to the global reference frame, facilitating navigation using relative poses within a global coordinate system.

- **State Estimator:** To ensure precise positioning relative to the windmill, an RTK-GPS system was employed. The DJI drone integrates this data to provide the software framework with accurate state estimation. The `ExternalEstimation` plugin was utilized for this purpose.

- **Behaviors:** Standard motion behaviors such as *Takeoff*, *Land*, and *Go To* were used. Additionally, a *Trigger Image* behavior was implemented to determine optimal moments for capturing images.

- **Plan Execution:** Each drone's mission plan is generated before execution. Once verified and accepted by the user, the plan is loaded onto the drone and interpreted using the *Mission Interpreter*. The mission can also be paused, stopped, or modified in real-time via the remote controller. A specialized mission planner was developed based on windmill parameters, including height, blade length, and GPS coordinates. Using these parameters, a predefined route is loaded onto the aircraft during setup.

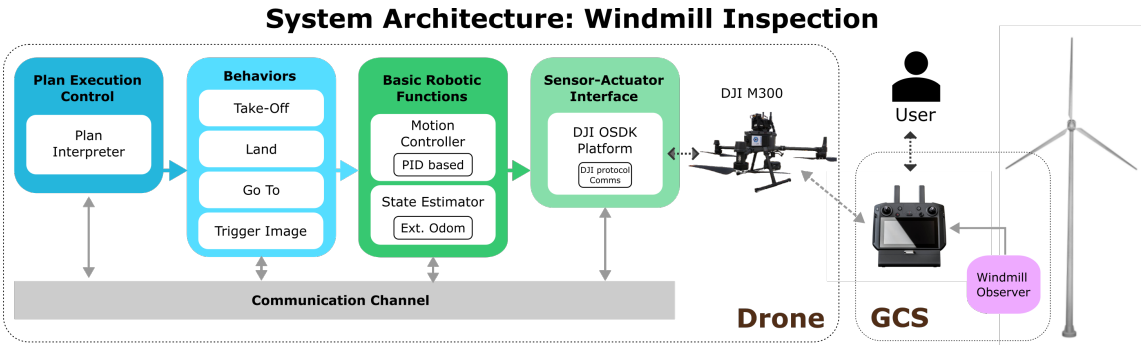


Figure 7.5: Overview of the system architecture used for real windmill inspection. In this setup, the system interfaces directly through the remote controller, which is also connected to the *Windmill Observer*, the component responsible for updating the nacelle's rotation over time.

7.2.3.1 Simulated Environment

For the simulation of the wind turbine, a model of a wind turbine has been created using the Blender (Community, 2018) tool. This model has been divided into three main parts, which are then converted into separate SDFormat[1] models and connected by rotary joints. These parts are:

- **Tower:** The static part of the wind turbine, linked to the ground. The base of the tower is set as the origin of the wind turbine coordinate system.

- **Nacelle:** A box-like structure that connects the tower with the rotor. The top of the tower is connected by a rotation joint that rotates in yaw (X-Y plane).

- **Blades:** This model contains the rotor with the blades. The rotor is connected to the front of the nacelle with another rotation joint that rotates in roll (Y-Z plane).

[1] `http://sdformat.org/spec?ver=1.9&elem=sdf`

To move the blades and the nacelle within the simulation, Gazebo's joint speed controller plugins have been used for each of the joints in the model. These plugins have then been bridged to ROS 2.

A simulated GPS sensor has been integrated into the nacelle to receive WGS84 coordinates. The Cartesian orientation of this sensor is used to calculate the azimuth, essential to obtain realistic data structures and values from the simulation.

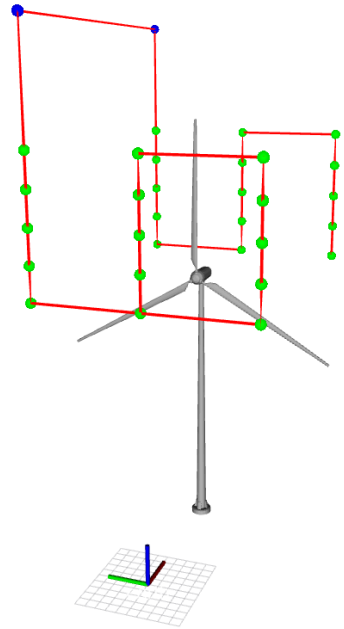


Figure 7.6: Simulated windmill and the predefined route points visualized

7.2.4 Results

This project resulted in the first Spanish-developed autonomous drone system capable of performing complete wind turbine inspections during normal operation. The key innovation lies in the system's ability to actively and dynamically compensate for continuous changes in the nacelle's rotation and rotor positioning in real-time. Unlike conventional inspection approaches, which typically require turbines to stop or significantly reduce speed, this solution allows for uninterrupted operation, offering substantial operational and economic advantages.

One major challenge the system successfully overcame was synchronizing drone navigation with the wind turbine's constantly rotating components. The developed software architecture tightly integrates real-time data from the *Windmill Observer*, ensuring precise drone positioning even under rapidly changing environmental conditions and varying rotational speeds. This integration allowed the drone to consistently adjust its trajectory and maintain safe distances from the rotating blades and nacelle, effectively addressing one of the primary operational challenges of drone-based turbine inspections.

The real-world experiments were conducted at a windmill farm in Galicia, Spain (see Figure 7.8). From these experiments, high-quality images of the windmill blades were obtained, captured perpendicular to their movement to maximize the visible blade area in each image (see Figure 7.7[2]). Qualitative evaluation demonstrated that the obtained images were of high quality, with clear visibility of blade surfaces. The captured imagery provided sufficient detail to allow identificaton of subtle surface defects, minor erosions, and potential microfractures, representing a qualitative improvement over traditional inspection methods.

Figure 7.7: Image of a windmill blade obtained during the inspection. All rights reserved by Generadron S.L. [3]

The successful demonstration of this innovative drone inspection solution directly contributed to the creation of the Generadron S.L. This venture underscores the commercial viability and significant potential impact of technology on the renewable energy sector. By enabling inspections without interrupting turbine operations, the system offers substantial economic benefits to wind farm operators through reduced downtime, improved operational efficiency, and enhanced safety for inspection personnel.

Looking ahead, there are considerable opportunities for future improvements. Potential next steps include integrating advanced artificial intelligence methods for automated defect detection, expanding inspections to coordinated multi-drone operations, and improving system robustness to handle harsher environmental conditions. Such developments could further amplify the effective-

[3] https://generadron.com/el-sistema/

ness, scalability, and overall impact of this technology across the wind energy industry.

Figure 7.8: Photograph of the drone autonomously flying behind the windmill.

Chapter 8

International Competitions

Another great opportunity to validate and extend the results of this thesis was participation in international robotics competitions. These competitions present new challenges to be addressed and allow for comparison of results with those of other research laboratories or institutions, driving technological advancements in various fields of interest.

Throughout the thesis period, we had the opportunity to participate in several competitions. In this chapter, we present the most relevant ones, the challenges each posed, how we approached them, and finally, the results obtained.

8.1 OpenCV AI Competition 2021

8.1.1 Challenge description

This competition challenges different competitors to showcase potential uses of the OAK-D Camera, a novel RGB-D camera developed by Luxonis.

In our proposal, we elaborate on the use case of *vision-based search and rescue with Micro Aerial Vehicles (MAVs) for catastrophic situations*.

During search-and-rescue operations following disasters, rapid victim localization is crucial for maximizing survival rates. The integration of state-of-the-art machine learning-based object detection algorithms, with MAVs holds significant potential for efficiently identifying victims and assessing structural damage, especially in hazardous environments where deploying first responders is risky. However, achieving full autonomy for MAVs remains challenging, as it requires reliable navigation in complex, cluttered, and dynamic environments, robust obstacle avoidance, precise self-localization, and accurate mapping capabilities, all while operating under intermittent or limited communication with ground stations.

8.1.2 Proposed Solution

The proposed solution consists of two main components that address critical challenges in autonomous MAV operations for disaster response:

1. Semantic SLAM: We propose integrating the Luxonis OAK-D visual sensing device with advanced SSLAM algorithms to enable the MAV to generate accurate semantic maps of disaster-affected areas. These maps will provide valuable contextual information, such as structural damage and victim locations, significantly enhancing situational awareness and supporting informed decision-making for rescue teams.

2. **Obstacle-Free Autonomous Navigation**: Utilizing the OAK-D sensor as the primary visual system, the MAV will generate real-time depth maps of unknown and cluttered environments. This capability will allow the MAV to safely navigate complex spaces, dynamically avoid obstacles, and rapidly locate victims, particularly in areas that are inaccessible or hazardous to human rescuers.

Both components rely on lightweight onboard processing units designed to maximize flight duration, facilitate robust onboard decision-making, and ensure reliable operation even in low-signal conditions.

In this thesis we will only cover the **Obstacle-Free Autonomous Navigation** component and we will treat it as an independent system.

8.1.2.1 Hardware Setup

The experimental aerial platform was a custom quadrotor based on the DJI F330 frame. It featured a Pixhawk 4 Mini autopilot, an OAK-D camera at the front, and an Intel RealSense T265 module at the back for state estimation. An NVIDIA Jetson Xavier NX SBC (6-core ARM v8.2, 64-bit) running Ubuntu 18.04 handled all onboard computations. With a 4-cell, 5000 mAh LiPo battery, the platform weighed 1.4 kg and had a flight time of approximately 8–13 minutes.

8.1.2.2 System architecture

For this application, the *Aerostack2* framework has been used. Figure 8.1 summarizes the different components used for this operation.

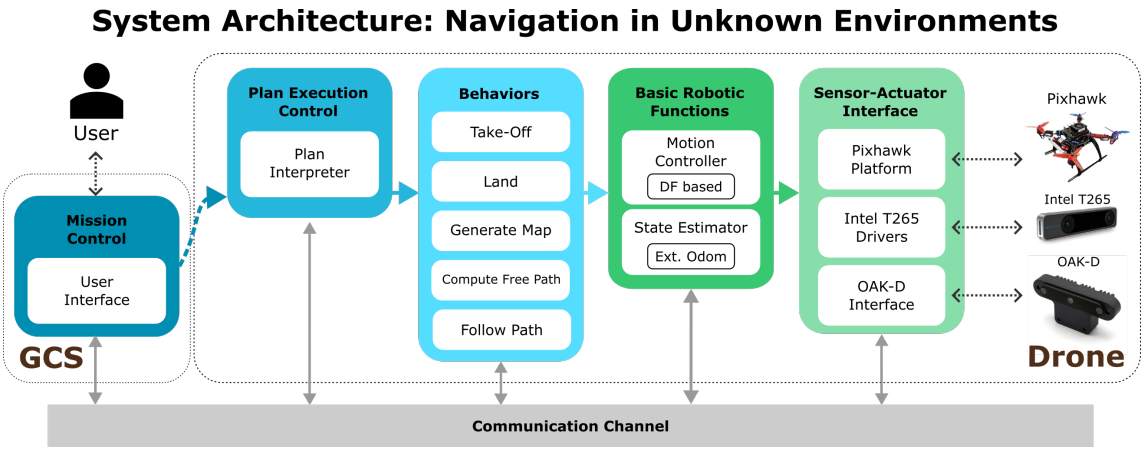
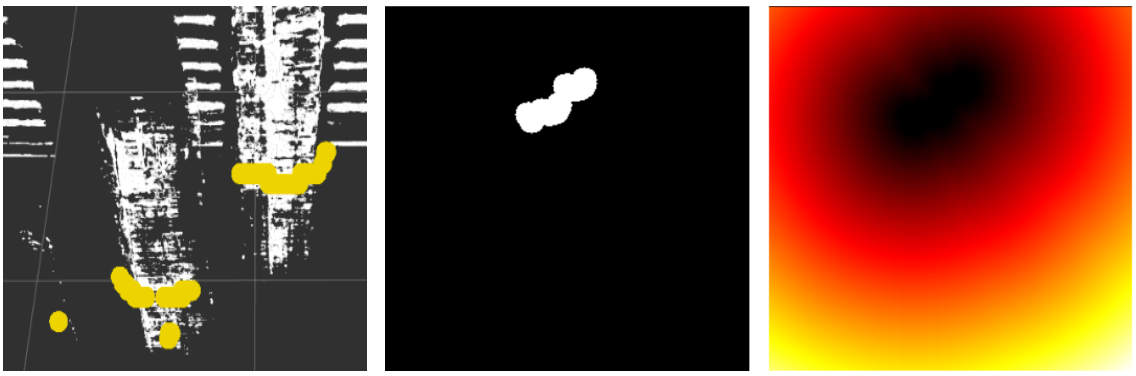


Figure 8.1: Overview of the system architecture.

The components used in each layer are as follows:

- **Sensor-Actuator Interface:** For this application, the Pixhawk platform was used, combined with components for interfacing with the two cameras mounted on the drone: the Realsense T265 and the Luxonis OAK-D.
- **Motion Controller:** The Differential Flatness (DF)-based controller plugin was used, enabling precise control of the aircraft.
- **State Estimator:** The Realsense camera provided visual odometry-based state estimation, utilizing the `ExternalEstimation` plugin.
- **Behaviors:** All basic motion behaviors, such as *Takeoff*, *Land*, and *Follow Path*, were used. Additionally, two custom behaviors were developed for this application:
  - *Generate Map:* Extracts a laser map representation from the depth image provided by the OAK-D sensor, creates a 2D occupancy grid based on the drone's pose, and finally computes the Euclidean distance field, which will be used for path planning. Figure 8.2 illustrates this process.
  - *Compute Free Path:* Plans a collision-free trajectory based on the Euclidean distance field using an A* search algorithms, see Figure 8.3. When new obstacles are detected along the flight path, a new path is generated for avoid collision.
- **Plan Execution:** The mission required navigating to a specific point in the scenario without collisions. A dedicated plan was created, directly interfacing with the *Compute Free Path* and *Follow Path* behaviors.



(a) Laser scan from depth map (b) 2D occupancy grid obtained from filtered laser scan (c) Euclidean distance field computed from the occupancy grid

Figure 8.2: *Generate Map* Behavior pipeline in three steps.

8.1.2.3 Results

To validate our approach and test the feasibility of the OAK-D camera for reactive navigation applications we designed some experiments that consisted of making a drone navigate through an indoor environment filled with obstacles without colliding, see Figure 8.4.

We performed experiments in two different scenarios by changing the goal position and the arrangement of the columns to test the robustness of our system (see Figure 8.5).

In Scenario 1, the drone started at (0,0) with the goal at (-2,5). Columns blocked a direct path, requiring the system to adjust the trajectory. The drone maintained an altitude of 1.2 m throughout the flight.

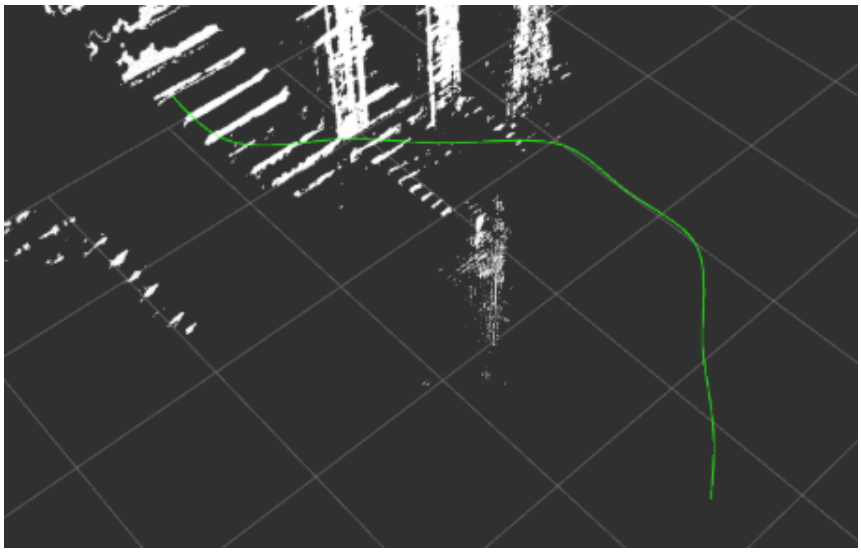


Figure 8.3: Trajectory generated by the *Compute Free Path* behavior, visualized along with the depth map in RViz.

In Scenario 2, the goal was set at (0,4), requiring two trajectory modifications to avoid obstacles and navigate through a narrow corridor. The drone again maintained a 1.2 m altitude.

The trajectories followed by the drone in both scenarios are shown in Figure 8.6. The results are summarized in Table 8.1.

Table 8.1: Experimental results for both scenarios.

Scenario	**Time (s)**	**Max Speed (m/s)**	**Avg Speed (m/s)**
1 (Goal: (-2,5))	19.1	0.83	0.36
2 (Goal: (0,4))	17.2	0.55	0.31

In both experiments, the system performed well. The obstacles present in the scenarios were accurately detected, and the distances between them and the aircraft were precisely estimated. This enabled the drone to generate a safe trajectory, successfully avoiding obstacles and reaching its goal.

The video showcasing the results of the complete proposed solution can be found at `https://vimeo.com/583816850`. With this approach, we achieved the **3rd Regional Prize** for Europe, Russia, and Australasia[1].

[1] `https://opencv.org/opencv-ai-competition-2021/`

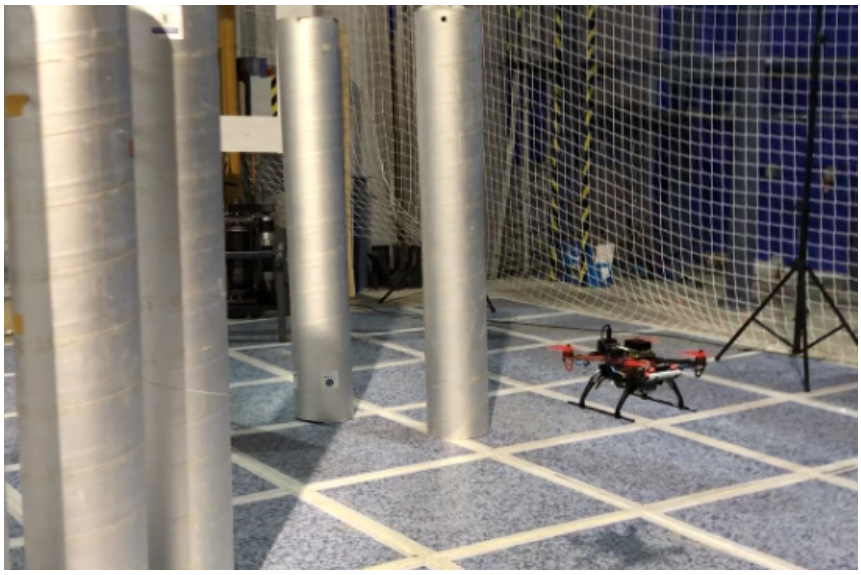


Figure 8.4: Experimental setup used to validate the proposed solution. The drone must navigate in a scenario with a set of columns.

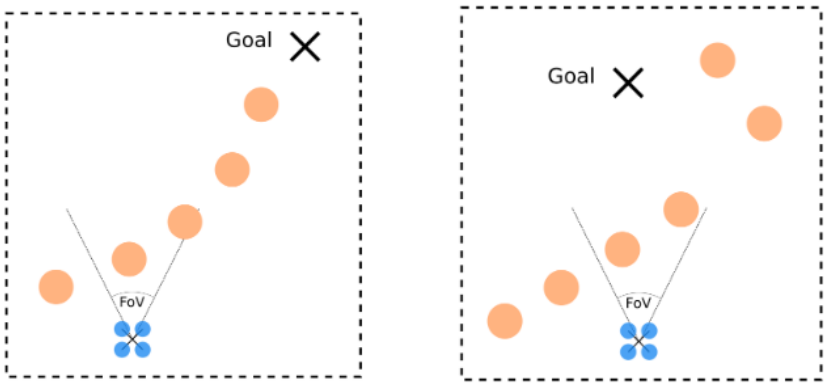


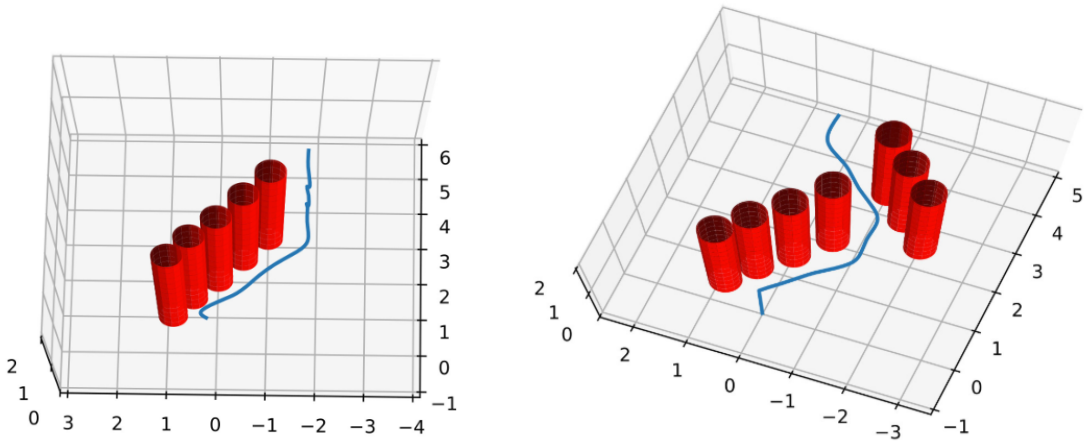
Figure 8.5: Arrangement of the drone's starting position, the columns, and the drone in two scenarios.

8.2 ICUAS 2022: UAV Competition

8.2.1 Challenge description

The ICUAS 2023 UAV Competition, organized by LARICS from the University of Zagreb, aimed to simulate the performance of a UAV in urban firefighting scenarios. The UAV had to autonomously navigate through complex environments, detect obstacles like buildings, and generate collision-free paths to the fire-affected area. Once there, it needed to locate the fire and deploy an extinguishing device, such as by releasing it through a window. The competition provided an external motion capture system for drone localization, eliminating the need for self-localization algorithms.

The challenge was structured as a simplified version of a real-world firefighting problem, requiring the UAV to complete three key tasks: navigating through a dense urban environment, detecting the fire location, and precisely delivering a fire-extinguishing ball to the target. The environment was divided into three zones: Zone 1 (a clear takeoff area), Zone 2 (an obstacle-dense navigation zone), and Zone 3 (a larger open space where the fire target was located), see Figure



(a) Trajectory followed in the first experiment.

(b) Trajectory followed in the second experiment.

Figure 8.6: Trajectories followed by the aircraft in both experiments.

8.7. The UAV's perception capabilities, speed, and agility were evaluated through sub-challenges, with the full details outlined in the competition rulebook.

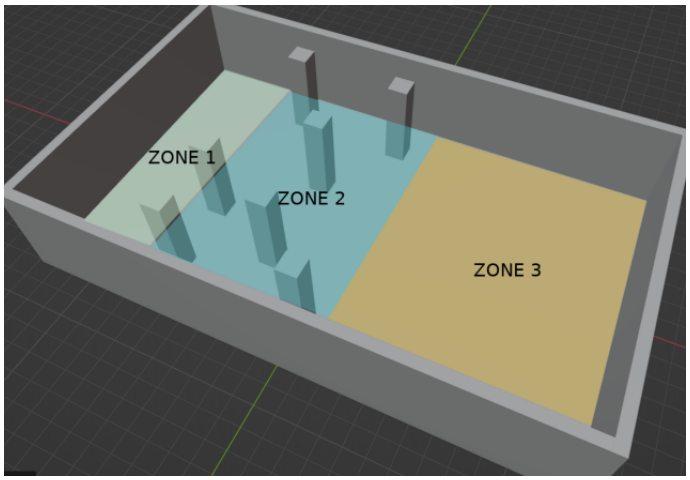


Figure 8.7: Virtual scenario divided into three zones: Zone 1, for taking off, Zone 2, for the exploration task, and Zone 3, for the precision delivery task.

The UAV had to perform three main tasks autonomously, each contributing to the final score. The exploration task required the UAV to navigate from Zone 1 to Zone 3 while avoiding unknown static obstacles in Zone 2. The target detection task involved identifying an AR tag placed in Zone 3, with scoring based on the accuracy of localization. The precision delivery task required the UAV to launch a fire-extinguishing ball at the target, with performance measured by the proximity of impact. The total mission time also factored into the final score.

Several constraints influenced the UAV's design and operation. The drone relied on an external localization system and used only an RGB-D camera for obstacle detection. The UAV had to execute the launch maneuver without a dedicated deployment system and operate under position and trajectory control, as low-level controllers were restricted. Additionally, all computations had to be performed onboard, and teams had no access to the real UAV before the on-site competition, making the transition from simulation to real-world performance a challenge.

Hardware description: The hardware used in the real challenge was the *Kopterworx Eagle* platform running *Ardupilot* firmware with a *Pixhawk* flight controller. The UAV was equipped with an onboard *Intel NUC* computer, which communicated with the flight controller via the Mavlink protocol, and an *Intel Realsense D435* camera providing both RGB and depth information.

Onboard the computer, a Docker container with ROS (Quigley et al., 2009), supplied by the competition organizers, was executed. This setup managed data from the motion capture system and the camera while also applying control signals for UAV movement and the ball launch actuator.

The hardware was provided by the organizers, meaning there was no access to it before the competition. However, a simulated environment was available in Gazebo (Koenig and Howard, 2004), replicating both the competition area and the UAV hardware, including its sensors.

8.2.2 Proposed Solution

In this competition, the proposed system architecture followed a Tier II design, optimized for this purpose, with a focus on behaviors, the task manager, and the controller. Aerostack2 was not used due to the system operating in ROS 1. However, some of the algorithms developed for this challenge have been integrated into the latest version of Aerostack2. The Hardware Interface and State Estimator modules were provided by the organizers and are therefore not considered in our proposed solution. Figure 8.8 illustrates how the components of the proposed solution fit within the Tier II architecture pattern.

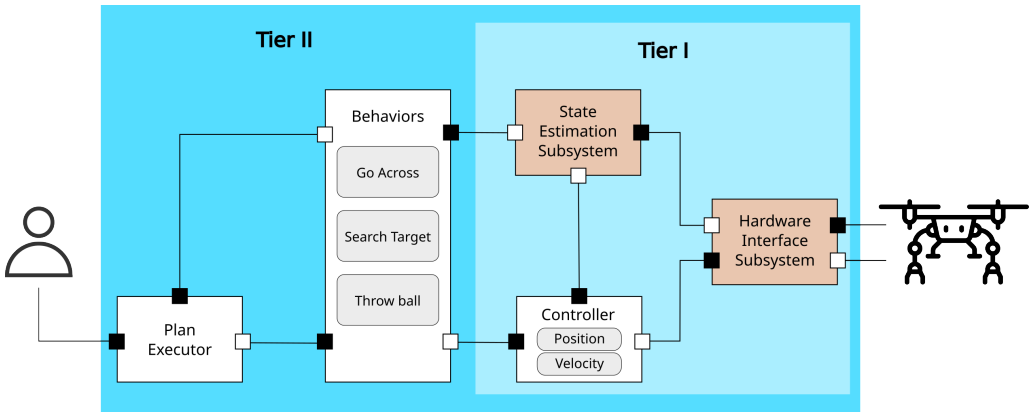


Figure 8.8: Overview of the Tier II architecture used for the competition. The brown blocks were provided by the organizers.

Throughout the mission, only two modules operated continuously: the **Plan Executor** and the **Motion Controller**. The remaining behavior modules were dynamically loaded and activated as needed for each stage of the mission.

- **Plan Executor:** Manages the activation and deactivation of behavior modules, ensuring the UAV executed tasks in sequence. It initiated necessary modules, monitored task completion, and transitioned between mission stages (see Figure 8.9).
- **Controller:** Provides precise control over the UAV's movement. Due to competition constraints, low-level commands such as attitude and speed were unavailable, so only position and trajectory control were allowed. For launch maneuvers requiring speed control, a secondary controller was implemented on top of the UAV's internal position controller. This controller modulated speed by adjusting the relative distance between the position reference and the UAV's current position, leveraging the aircraft's cascade PID control system.

- **Behaviors:** Three key **Behavior Modules** were designed based on the motion references handled by the controller:
  - **Go Across Behavior:** This module generated collision-free paths using A* graph search over an online occupancy grid. To optimize movement, waypoints along straight-line paths were simplified, reducing unnecessary stops and improving flight fluidity (see Figure 8.10).
  - **Search Target Behavior:** This module operated in two phases: search and inspection. In the search phase, the UAV followed a predefined sweeping pattern at a safe distance, d_{search}, from obstacles, ensuring full camera coverage of the search area. Once the target was detected, the inspection phase began, where the UAV hovered at an inspection waypoint until positional measurements stabilized within a threshold of $th_{inspection} = 5$ mm before sending the final target position to the mission planner.
  - **Throw Ball Behavior:** Given the restriction on low-level control, complex throwing maneuvers were avoided. Instead, a simple constant-speed maneuver was used for ball deployment. The UAV positioned itself at $d_{launch} = 6$ m from the target at a height of $z_0 = 1.5$ m. It then flew at a constant speed of $v_x = 3.5$ m/s along a predefined X-Z trajectory. During this motion, the system continuously evaluated the ball's falling dynamics to determine the optimal release moment, ensuring accurate target impact (see Figure 8.11).

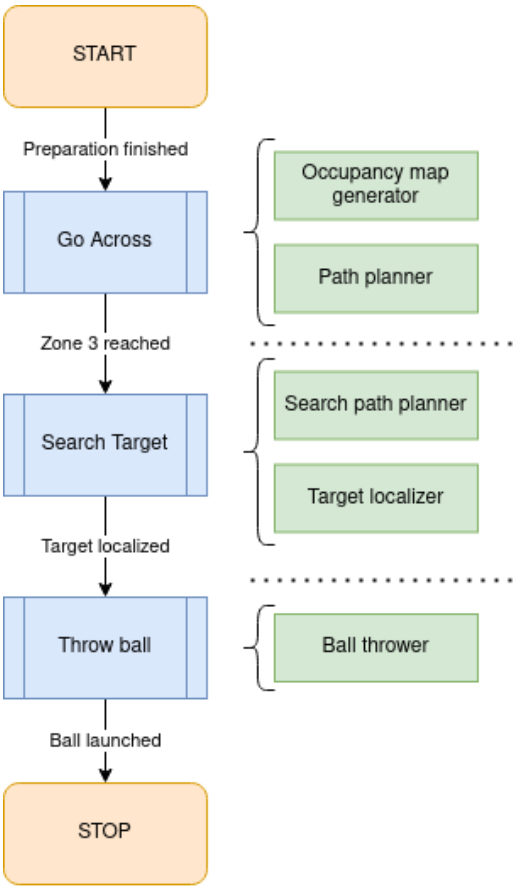


Figure 8.9: Plan executor flow diagram.

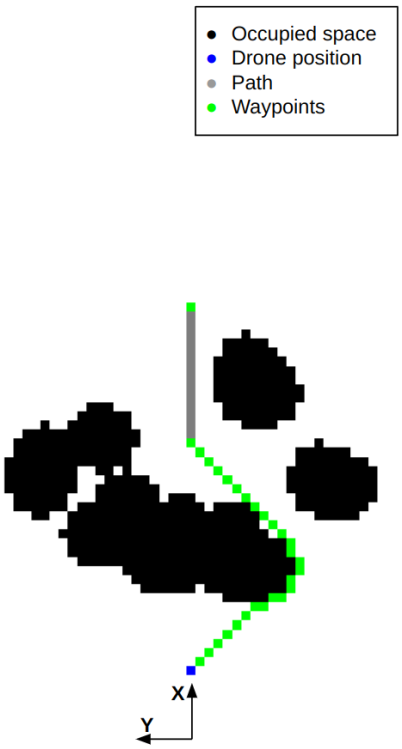


Figure 8.10: Occupancy map with the path free of obstacles generated by the path planner. The path calculated is represented in gray, and the waypoints of this path that are sent to the controller are marked in green. Each pixel represents a 0.5 meter square in the real world.

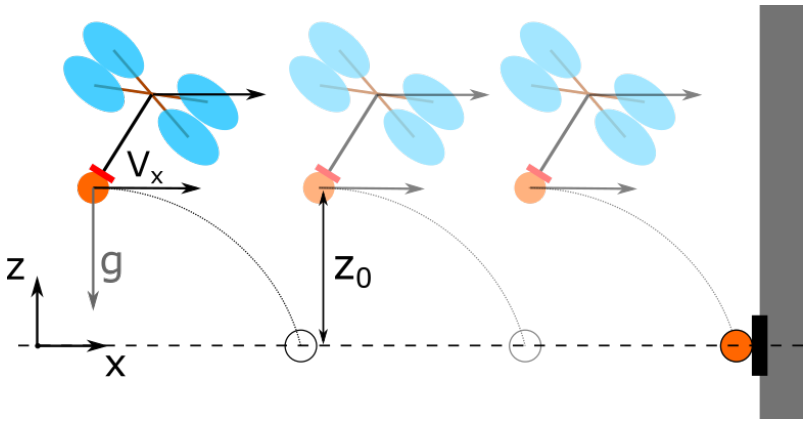


Figure 8.11: Ball throwing maneuver scheme. The drone will attempt to achieve a constant speed while maintaining a relative height to the target. When the system predicts that the ball will land on the target, it is released, and the drone moves away from the wall.

8.2.3 Results

8.2.3.1 Classification: Simulation environment

For this phase, we show the metrics obtained in the evaluation performed by the organizers. Since the first evaluations were done to check that our solution worked correctly in the evaluation test, we have chosen the last 10 evaluations. The results are shown in Tables 8.2 and 8.3.

Table 8.2: Average time from 10 last evaluations during the classification phase.

Task	Time(s)
Obstacle zone crossing	7.99 ± 2.65
Target localization	22.79 ± 19.10
Ball throwing	3.75 ± 0.25
Total mission time	34.54 ± 18.69

Table 8.3: Average error from 10 last evaluations during the classification phase.

Task	Error(m)
Target localization	0.08 ± 0.06
Target localization (y)	0.04 ± 0.06
Target localization (z)	0.05 ± 0.04
Hit accuracy	0.17 ± 0.05

The metrics indicate that the localization of the target was highly accurate, with an average error of 0.08 meters—less than half the size of the AR tag. This error is similarly distributed along the y-axis (horizontal) and z-axis (vertical). There was no error along the x-axis (depth) due to prior knowledge that the AR tag would be on a wall.

Regarding throwing accuracy, the average error is 0.17 meters, approximately the size of the AR tag. Given that the ball has a diameter of 0.4 meters, this implies that the ball typically lands close to the target.

Fig. 8.12 illustrates the trajectories of both the drone and the ball during a simulation trial, along with the distances between the tag position, target localization, and ball contact point.

Based on these results, we ranked first among the 15 international teams that participated in this phase[2].

[2] https://github.com/larics/icuas22_competition/discussions/68We

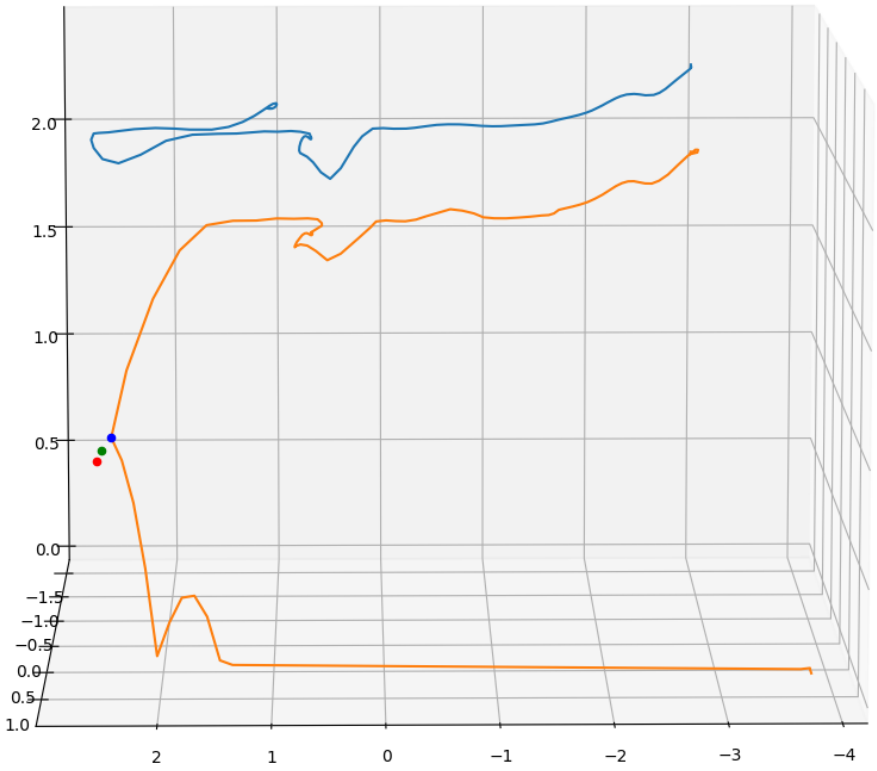


Figure 8.12: Ball throwing maneuver showing drone trajectory (blue), ball trajectory (orange), tag actual position (red), target localization (green), and ball contact point (blue).

8.2.3.2 Finals: Real Environment

The top five teams from the previous phase qualified for the finals. The target was a 190 × 190 mm AR tag printed on A3 paper (297 × 420 mm) and placed on a structure with a 45-degree face bend. The surface was twice the size of the paper, so accuracy was measured based on half the diagonal distance:

- **AR Tag (Goal):** 0.13 m
- **Paper (White Surface):** 0.26 m
- **Structure (Black Surface):** 0.51 m

If the ball missed all surfaces, it was excluded from accuracy calculations.

Adjustments for the Real Environment Differences from simulation required modifying several parameters. The map size and search path limits were reduced, and the drone's maneuvering parameters were adjusted accordingly. The target inspection was conducted at a distance of 2 meters and a height of 1 meter, while the ball launch was executed from 3 meters away at a height of 1.5 meters with a velocity of 2 m/s. Additionally, the tag detection threshold was increased to 0.1 meters to account for reduced real-world accuracy. For obstacle avoidance, the drone's maximum speed was limited to 1 m/s, which was still significantly faster than other teams. As a result, the drone achieved an average mission completion time of 20 seconds—nearly twice as fast as the second-best team. However, due to the absence of comprehensive team-wide metrics, direct comparisons were not entirely conclusive.

Table 8.4 shows mission times, while Table 8.5 details accuracy.

As the last team scheduled, we encountered **magnet overheating**, causing delayed ball releases and trajectory deviations. To counteract this, we adjusted launch parameters ($d_{launch} = 1m, z_{launch} = 1m$) and fine-tuned t_{delay} based on magnet response. After three successful hits—the only team to do so—our shift was stopped by the organizers.

Table 8.4: Average time (s) from completed trials during finals in each scenario.

Task	Scenario 1	Scenario 2
Obstacle zone crossing	7.28 ± 0.76	8.50 ± 0.55
Target localization	3.57 ± 0.79	4.33 ± 0.82
Ball throwing	6.67 ± 0.82	7.33 ± 0.52
Total mission time	17.67 ± 0.82	20.17 ± 0.98

Table 8.5: Number of hits on each surface of the throwing completed trials during the finals in each scenario, and estimated hit accuracy error.

Surface	Scenario 1	Scenario 2
Goal	1	2
White	2	0
Black	1	2
Out	2	2
Accuracy error (m)	0.29 ± 0.16	0.32 ± 0.22

Our solution achieved great precision in the throwing maneuver, with a mean minimum distance between the ball and the target of 0.17 meters in simulation, and successfully throwing the ball with a real drone, repeatedly hitting a target of 19×19 mm, see Figure 8.13.

With these results, the team qualified for the finals with the highest score in the simulation classification phase. In the real environment, our team successfully traversed the arena and obtained three successful goals, ending the competition as the winner of the challenge (see Figure 8.14). Additionally, it was the fastest solution, completing the mission with an average time of 20 seconds[3]. Video `https://vimeo.com/776655431` shows both the simulation and the real-world competition recordings.

[3] `http://www.uasconferences.com/2022_icuas/winners-of-the-uav-competition/`

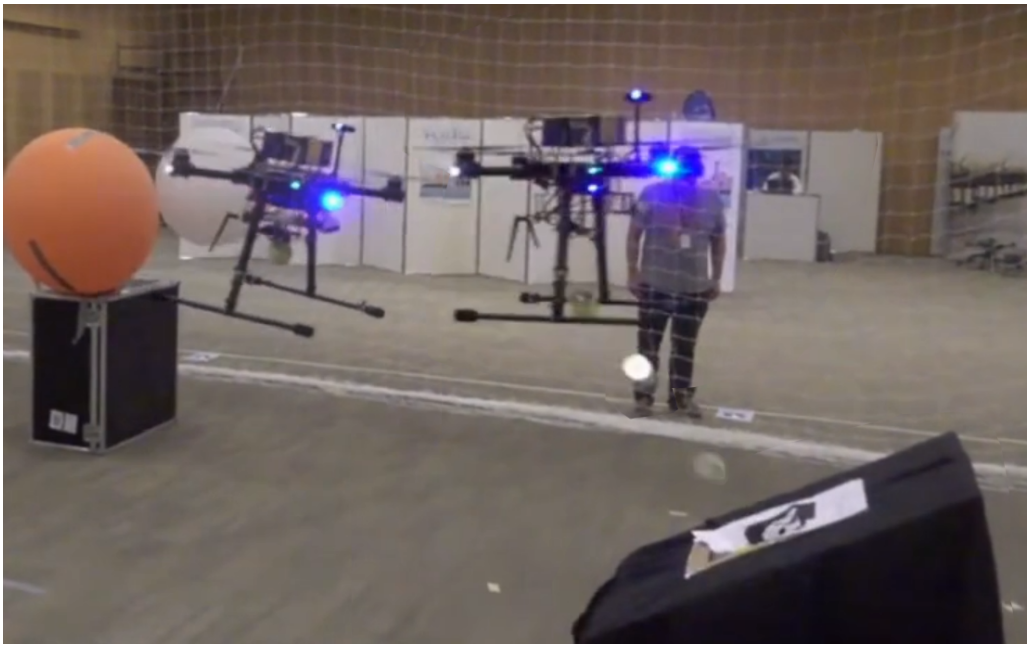


Figure 8.13: Timelapse of the drone scoring the ball

Figure 8.14: ICUAS 2022 Winner Diploma awarded to our team

8.3 IMAV 2022: Nanocopter AI challenge

8.3.1 Challenge Description

The International Micro Aerial Vehicles (IMAV) 2022 Competition, organized by MAVLab from the Technical University of Delft, aimed to advance vision-based autonomous flight capabilities of nano-scale drones, specifically employing Bitcraze's Crazyflie drone equipped with a single camera and a small onboard neural network processor (Bitcraze's AI Deck). Teams were required to guide their drones through an obstacle course as rapidly as possible within a predefined arena, covering the maximum distance within an allotted time.

The arena consists of an 8 x 8 m effective flying space within a 10 x 10 m designated area. Participants will encounter obstacles whose precise types, locations, and even movements (dynamic but not directly obstructing the immediate drone path) are unknown prior to flight, see Figure 8.15. Additional scoring opportunities are available by successfully navigating through orange gates positioned within the arena. These gates are intended to provide a target for more reliable detection methods compared to general obstacle avoidance.

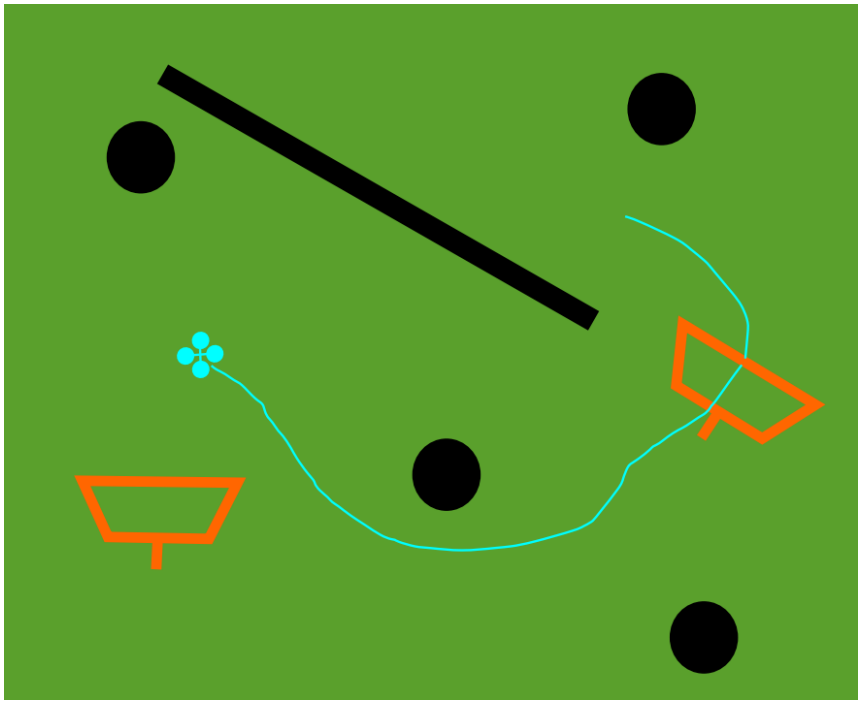


Figure 8.15: Simplified schema of the arena. The white drone attempts to traverse the maximum distance, passing through the red gates while avoiding the black obstacles.

Scoring considers multiple factors, including total flight distance, the number of successfully navigated gates, and whether processing is onboard or offboard. Teams can adjust obstacle interaction complexity, affecting the environmental scoring factor:

$$env\_factor = \begin{cases} \text{Only gates} : 1 \\ \text{Gates and obstacles} : 5 \\ \text{Gates with dynamic obstacles} : 10 \end{cases}$$

Each successfully navigated gate is equivalent to traversing an additional 10 meters, with two gates present in total. The processing mode further impacts scoring, rewarding onboard im-

age processing significantly ($processing\_factor = 5$), compared to offboard image processing ($processing\_factor = 1$).

Teams had two five-minute flights, with their highest-scoring attempt determining their final standing.

8.3.2 Proposed Solution

For this competition we used Aerostack2 software running on an ground station that received the images from the camera through WiFi and communicates with the aerial platform through a Radio Antenna.

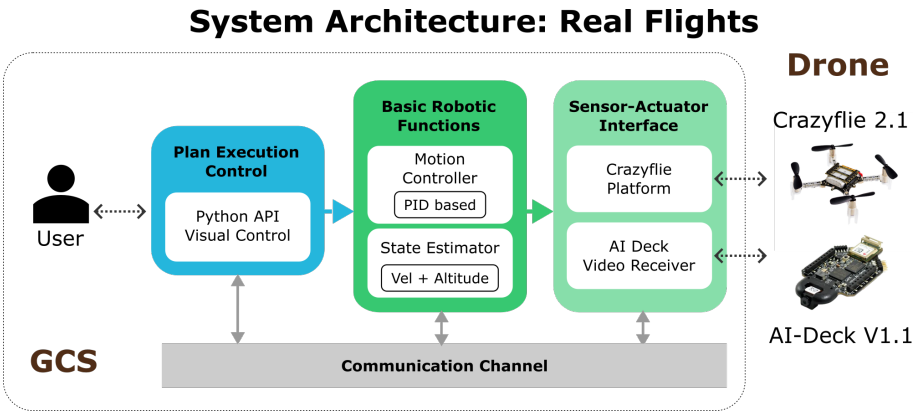


Figure 8.16: Overview of the system architecture used for the IMAV challenge. All computations were performed within the GCS, and the output commands were sent to the drone via radio.

For this competition we opted for a reactive visual approach, focused in detecting the gates and traverse them. In this reactive approach we take advantage of the flexibility that Aerostack2 supports for deciding which ammount of layers are required limitating our use to the following ones:

- **Sensor-Actuator Interface:** The Crazyflie platform has been used for communicating with the aerial platform. To communicate with the AI Deck (the camera board), a UDP video bridge has been used to stream the image into the ROS 2 network.
- **Motion Controller:** For this application, we only provided speed commands through a PID controller plugin.
- **State Estimator:** This approach was reactive, so only the attitude, velocity, and altitude information of the aircraft was obtained.
- **Visual Control Mission:** In this application, we take advantage of the *Python API*, allowing us to directly receive the image and provide speed references to the motion controller. With this, we generate a monolithic component that receives the image, filters it to obtain the position of the gates in the image, and, based on this, generates the motion commands, see Figure 8.17.

We considered that if the drone is able to see a gate directly, it will be able to traverse it without colliding. After that, if no gates are found, a constrained search can be performed to find the next one.

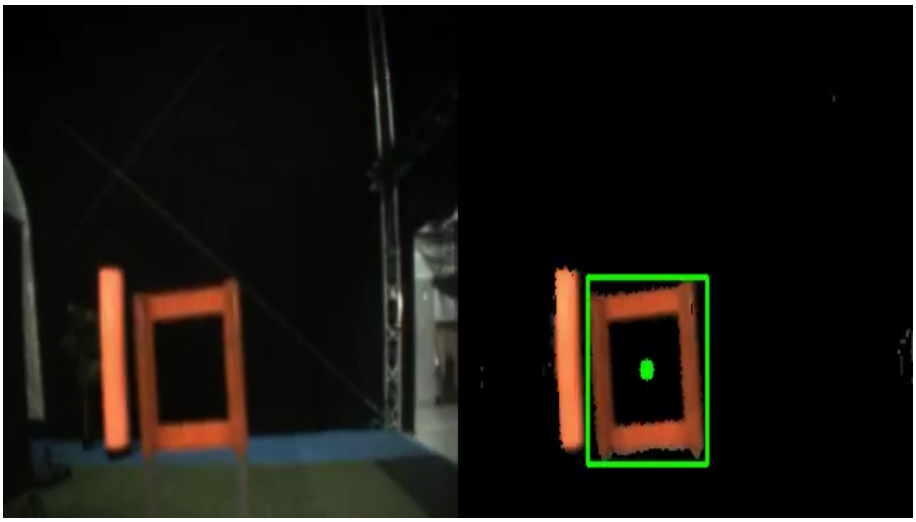


Figure 8.17: Visual control approach. On the left, the obtained image; on the right, the masked version of the gate. The rectangle encloses the gate, and the green point marks the reference point at the center of the gate. The algorithm tries to center this point in the middle of the image while continuously advancing forward.

8.3.3 Results

With this approach, we were able to consistently detect the gates and pass through them. An example of one such passage can be seen in Figure 8.18. However, when no gates were in clear sight, the drone was unable to avoid obstacles properly, leading to collisions.

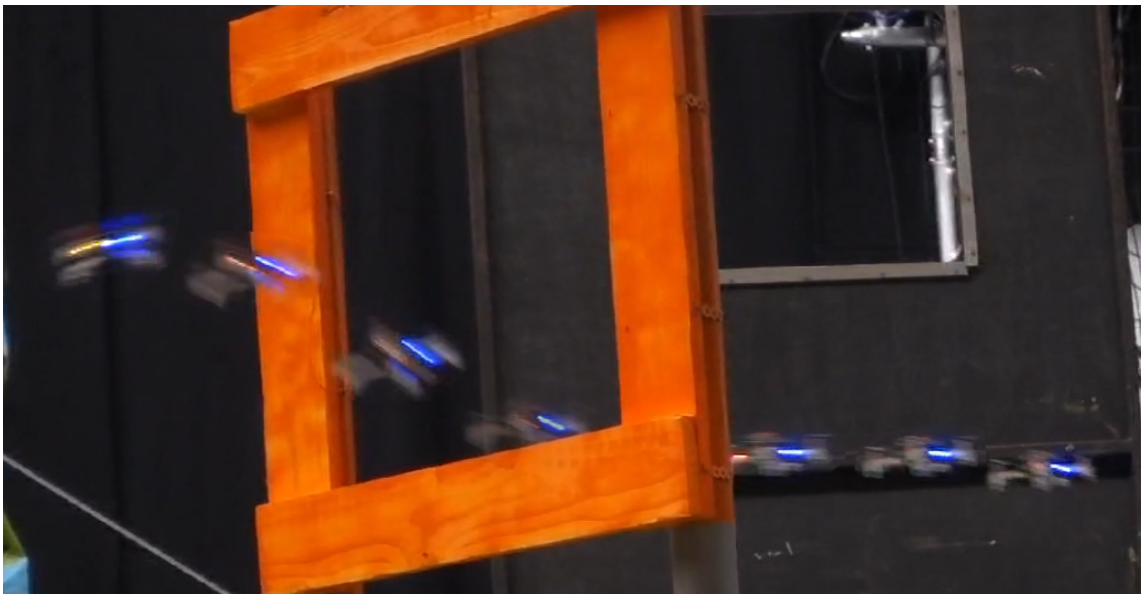


Figure 8.18: Timelapse of the Crazyflie crossing the gate in the IMAV22 competion

Nevertheless, with this approach, we achieved the Special Achievement Award, "Best Gate Passing," due to our consistency in passing through the different gates (see Figure 8.19).

Figure 8.19: Special Achievement Award diploma obtained from our participation in the IMAV 22 Nanocopter AI Challenge.

Our participation in this competition was the first time that Aerostack2 was used in an onsite competition. Through this experience, we were able to validate some of the foundational capabilities of this framework and identify faults that were subsequently resolved. A notable example was the Crazyflie Aerial Platform, which caused many connectivity issues during the competition and was corrected afterward.

8.4 ICUAS 2023:

8.4.1 Challenge Description

The ICUAS 2023 UAV Competition, organized by LARICS from the University of Zagreb and CATEC from Seville, revolved around UAVs performing infrastructure inspection tasks within unknown environments. This scenario required UAVs to showcase their capabilities in autonomous navigation, defect detection, and precise pose estimation. The competition consisted of two main phases: simulation-based qualifiers and live trials at the ICUAS '23 conference venue. The competition was subdivided into three different benchmarks:

- **Benchmark 1: Exploration.** This benchmark tested the UAV's capability to autonomously navigate unknown areas, identify points of interest, scan for defects, and safely return to the starting point. The performance evaluation considered the effectiveness of autonomous exploration, obstacle avoidance, and identification of points of interest within a dense 3D environment. Successful completion required UAVs to avoid crashes, minimize obstacle contacts, and conclude the run within specified time constraints.
- **Benchmark 2: Perception.** UAVs were evaluated on their capability to accurately detect infrastructure defects from captured images. The accuracy of defect detection was assessed using the Intersection over Union (IoU) metric between estimated detections and true labels, expressed via the Critical Success Index (CSI).
- **Benchmark 3: Pose Estimation.** UAVs aimed to accurately estimate their position and orientation using onboard sensors alone. The accuracy of pose estimation was measured by calculating the Root Mean Squared Error (RMSE) against ground truth data from a precise motion tracking system.

Hardware Description: The hardware used in the real challenge for the first benchmark was the same as that used in the ICUAS 2022 Competition: a *Kopterworx Eagle* platform running *Ardupilot* firmware with a *Pixhawk* flight controller. The UAV was equipped with an onboard *Intel NUC* computer, which communicated with the flight controller via the MAVLink protocol, and an *Intel RealSense D435* camera providing both RGB and depth information.

8.4.2 Proposed Solution

For the competition, we worked on all three benchmarks. However, for this thesis, we will primarily focus on how the exploration benchmark was approached, as the other two were evaluated using datasets rather than real-time system performance. Nevertheless, a brief description of the techniques used for these benchmarks is as follows:

- **Benchmark Perception:** For this task, a YOLOv4 (Bochkovskiy et al., 2020) implementation was used and trained on the provided dataset. To enhance algorithm performance, a data augmentation pipeline was implemented.
- **Pose Estimation:** In this case, ORB-SLAM v3 (Mur-Artal et al., 2015) was used. The system was fine-tuned using some of the datasets recorded by the organization and evaluated until the best results were obtained.

For the **Exploration** benchmark, the pose of the UAV was obtained through a MoCap system. However, for onboard defect detection, a tiny YOLOv4 version was used, trained with the same dataset as in the corresponding benchmark.

Due to the hardware constraints and similarities with the previous ICUAS competition, many of the system components were reused without modification.

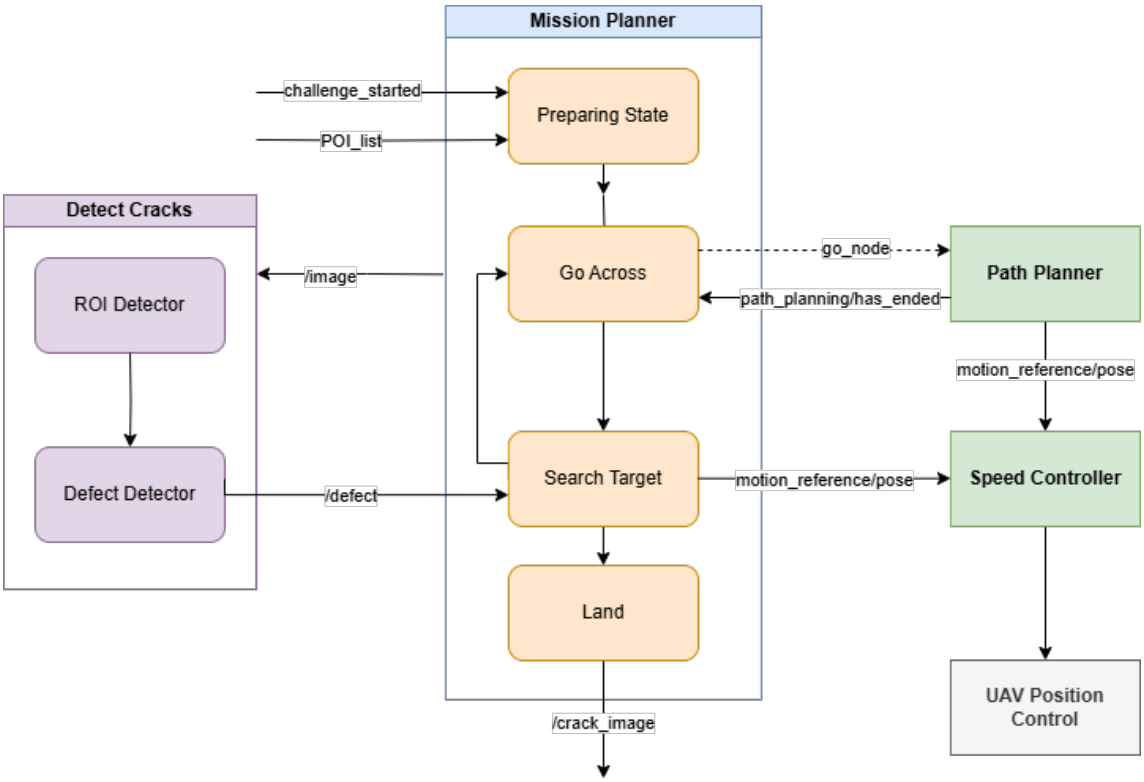


Figure 8.20: Schema of the software architecture for the ICUAS23 competition. The square boxes represent processes, while the rounded ones indicate logical components of each process

Similarly, throughout the mission, the **Plan Executor** and the **Motion Controller** operated continuously. The remaining behavior modules were dynamically loaded and activated as needed for each stage of the mission, see Figure 8.20.

- **Plan Executor:** Receives the interest points to be inspected. It then manages the activation and deactivation of behavior modules, ensuring that the UAV executes tasks in sequence. It initiates necessary modules, monitors task completion, and transitions between mission stages (see Figure 8.9).
- **Controller:** Provides speed commands to the drone, a functionality not supported by the organizers, who only allowed position control.
- **Behaviors:** Three key **Behavior Modules** were designed based on the motion references handled by the controller:
  - **Go Across Behavior:** Generates collision-free paths using A* graph search over an online occupancy grid, which is generated from the depth image of the RealSense camera. To optimize movement, waypoints along straight-line paths are simplified, reducing unnecessary stops and improving flight fluidity (see Figure 8.10).
  - **Search Target Behavior:** Operates in two phases: search and inspection. In the search phase, the UAV follows a predefined sweeping pattern while moving up and down at a safe distance from obstacles, ensuring full camera coverage of the search area.

Meanwhile, the *Detect Defects* module is enabled. If no defect is detected, the mission proceeds to the next point.

– **Detect Cracks Behavior:** This module encapsulates the defect detection process, which is subdivided into two parts. First, it localizes potential defect areas, and then it runs inference using the neural detector over these regions. Since defects were represented as printed images on paper, the system first segments the paper using classical vision techniques before applying the neural detector. This approach ensures the detector operates only on the relevant area, as the network was trained specifically with printed defect examples ,see Figure 8.21.

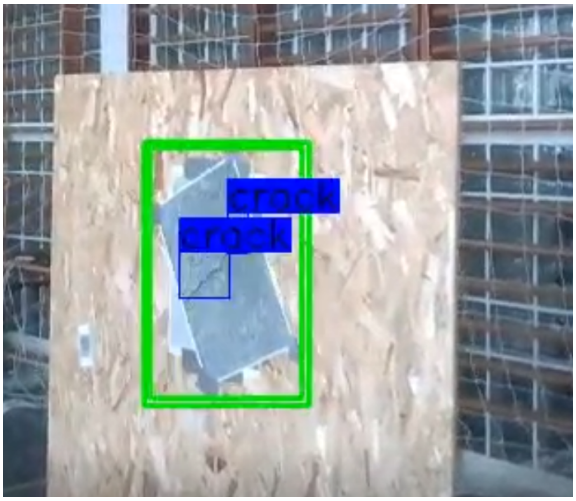


Figure 8.21: Image of the detector during the competition. The green rectangle indicates the region of interest, while detections are marked with annotated blue rectangles.

8.4.3 Results

8.4.3.1 Classification: Simulation Environment

The simulated environment was a highly complex industrial site with multiple pipes and rods that the UAV had to navigate through (see Figure 8.22).

For each run, 10 Points of Interest (POIs) were provided, each of which could either contain a crack or not. The score was computed as follows:

$$\begin{aligned} \text{Score} = {} & 25 \times (\text{percentage of POIs visited}) \\ & + 15 \times (\text{percentage of correct detections}) \\ & - 0.5 \times (\text{number of false positives}) \end{aligned} \tag{8.1}$$

The complexity of the environment led us to achieve only 15 points in this scoring, finishing 5th in this classification phase, which allowed us to advance to the finals.

8.4.3.2 Finals: Real Environment

In the real environment, the scenario was highly simplified, allowing us to consider a 2D navigation problem, similar to the one in the previous ICUAS competition (see Figure 8.23).

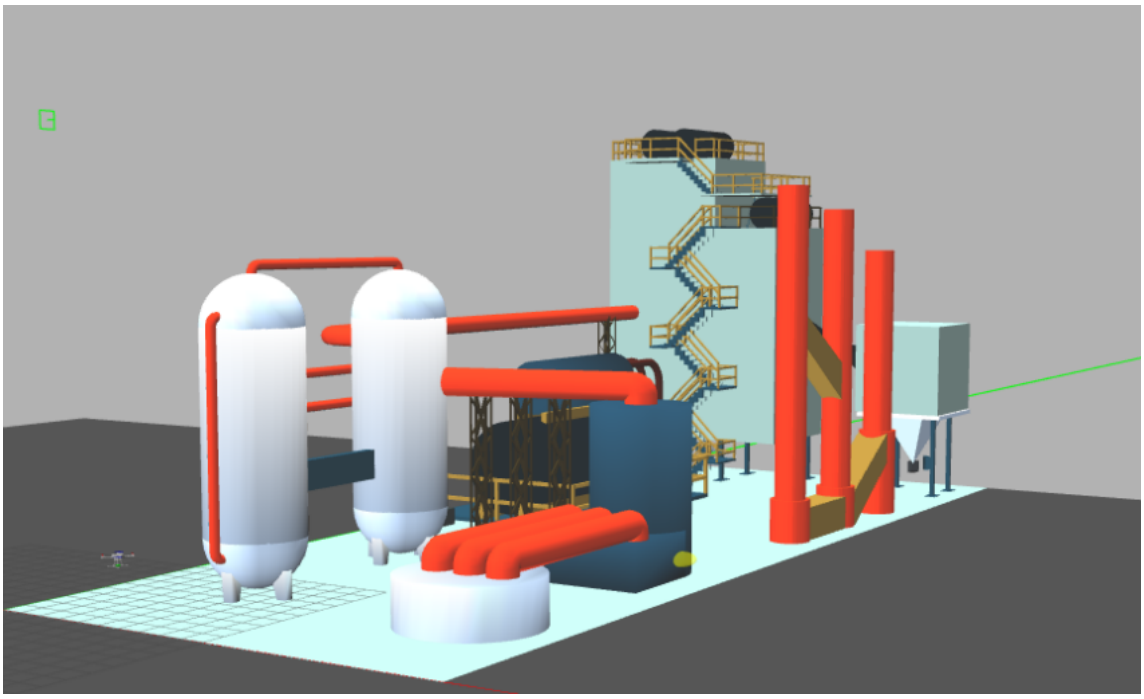


Figure 8.22: Competition arena in the Gazebo simulation environment.

Figure 8.23: Real environment scenario with the drone flying while searching for defects.

In this scenario, we navigated robustly through the arena, searching for defects and detecting them properly, ultimately achieving the best results and securing first place in this benchmark[4].

[4] https://www.uasconferences.com/2023_icuas/uav-competition-final-results/

This success, combined with our strong performance in the other two benchmarks, positioned us as the winners of the ICUAS 2023 Competition (see Figure 8.24).

Figure 8.24: Award of the ICUAS 2023 UAV Competition

8.5 ICUAS 2024: UAVs for Indoor Agriculture Challenge

8.5.1 Challenge Description

The main goal of the ICUAS 2024 UAV Competition is to enhance autonomous capabilities of UAVs in indoor agricultural settings. Teams will develop software solutions to autonomously perform fruit counting tasks within a controlled greenhouse-like environment, represented through Gazebo simulations integrated with ROS. The UAVs must autonomously navigate within the arena, count fruits accurately, and perform the task efficiently with minimal energy usage.

The virtual competition arena simulates an indoor farm environment, featuring multiple raised garden beds containing three different species of plants, each with fruits varying in shape and color. UAVs will receive prior information specifying beds that contain a particular plant variety and must autonomously plan an optimal trajectory for accurate fruit counting.

The arena simulates an indoor farm consisting of multiple garden beds, each with a specific plant species bearing fruits of varying visibility, size, and color, see Figure 8.25. The UAVs must plan their flight paths to achieve thorough inspection and accurate fruit counting.

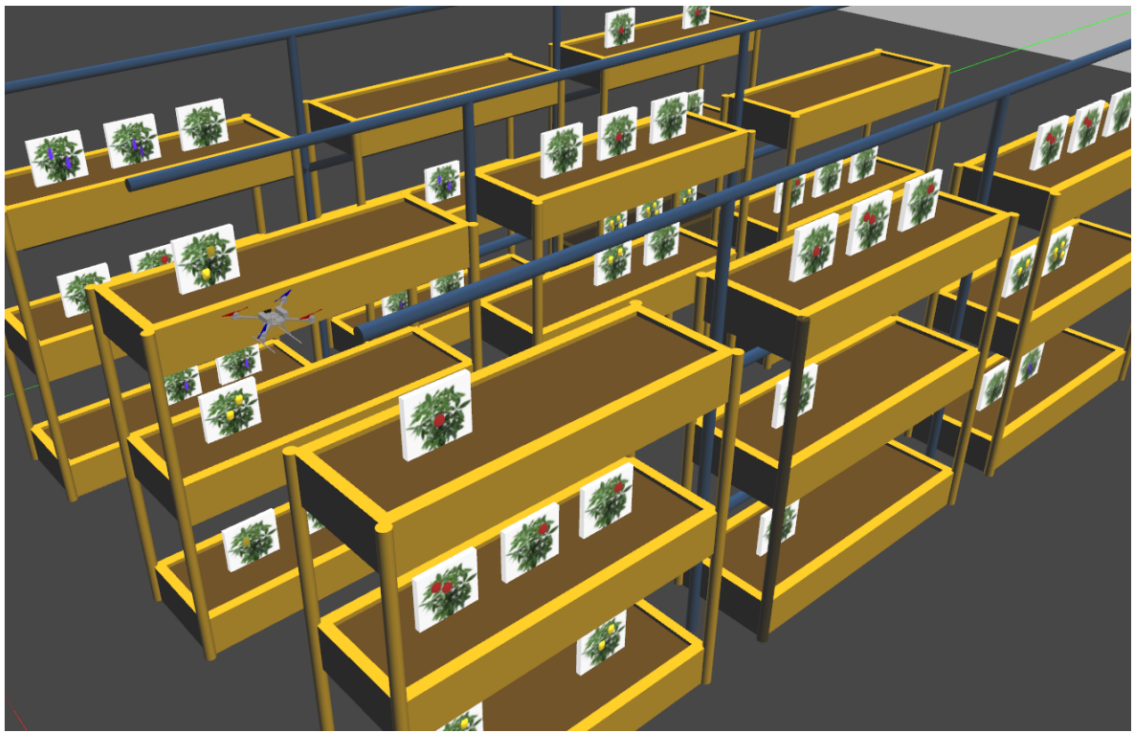


Figure 8.25: Competition Arena in the Gazebo simulated environment

The competition evaluates UAV performance through two benchmarks: Yield Estimation, measuring the UAV's ability to autonomously and accurately count fruits within a virtual greenhouse; and Energy Efficiency, assessing the UAV's capability to minimize energy consumption by optimizing flight trajectories and reducing flight duration. Penalties are imposed for collisions with obstacles, while severe infractions such as UAV crashes, flying over plants, or code execution failures result in disqualification.

8.5.2 Proposed Solution

Similarly to previous ICUAS competitions, we opted to continue with the existing architecture, maintaining a *Mission Planner* responsible for orchestrating the activation and deactivation of different components (see Figure 8.26).

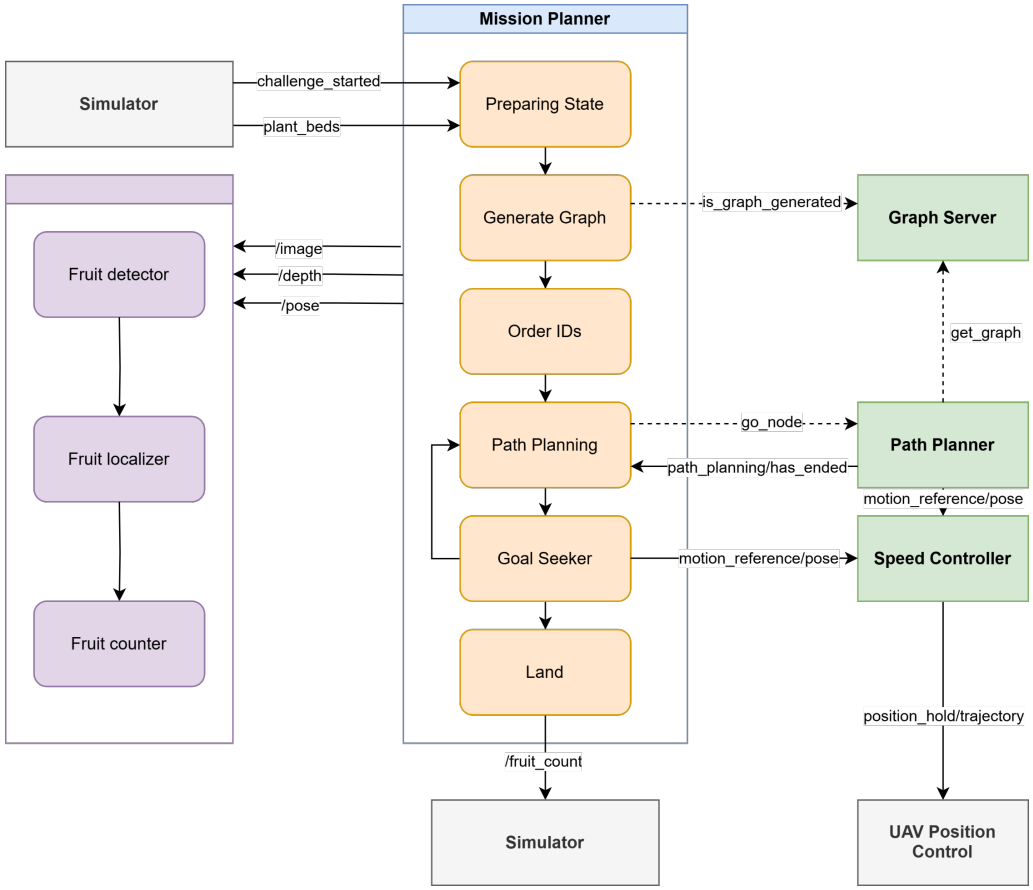


Figure 8.26: Schema of the software architecture for the ICUAS24 competition. The square boxes represent processes, while the rounded ones indicate logical components of each process.

The mission planner operates as follows:

1. *Preparing State:* In this phase, the system waits to receive the list of plant beds to be inspected—identified by an ID for each bed—and the type of fruit to be counted.

2. *Generate Graph:* A 3D graph of valid UAV positions is generated. This graph includes waypoints at the center of each shelf and all necessary pass-through points for collision-free navigation. Additionally, shelves selected for inspection are annotated.

3. *Order IDs:* Using the generated graph, a simplified Travelling Salesman Problem (TSP) problem is solved to determine the optimal traversal order for the garden beds. This results in an ordered list of positions to be inspected.

4. *Path Planning:* While there are remaining inspection points, the algorithm iterates. The drone navigates to the required position via a collision-free path, activates the *Fruit Detector Node*, waits to ensure proper image capture, and then proceeds to the next point.

5. *Land:* Once all beds have been inspected, the drone lands and transmits the final count.

To accomplish these tasks, several auxiliary nodes are required:

- **Vision Pipeline:** This pipeline is divided into three sequential stages:
  1. *Fruit Detector:* A detection algorithm is executed based on the fruit type. Once detected, the center of the fruit is identified.
  2. *Fruit Localizer:* The detected fruit center is combined with depth map data and the drone's pose to estimate its position within a 3D map.
  3. *Fruit Counter:* Using the 3D map, a filtering process prevents duplicate counting when the same bed is inspected from both the front and back.
- **Navigation Modules:** These modules enable effective drone navigation:
  - *Graph Server:* Stores the graph of traversable positions and maintains information on which beds are visible from each position.
  - *Path Planner:* Runs an A* algorithm on the *Graph Server* to compute the optimal route from the drone's current position to the required fruit bed and sends pose references to the speed controller.
  - *Speed Controller:* Converts pose references into UAV position control commands, interfacing with the position controller provided by the competition organizers.

Figure 8.27 shows the system's visualization of the different beds, the generated graph for this arrangement, and the planned path that the drone will follow to pass through the specified beds.

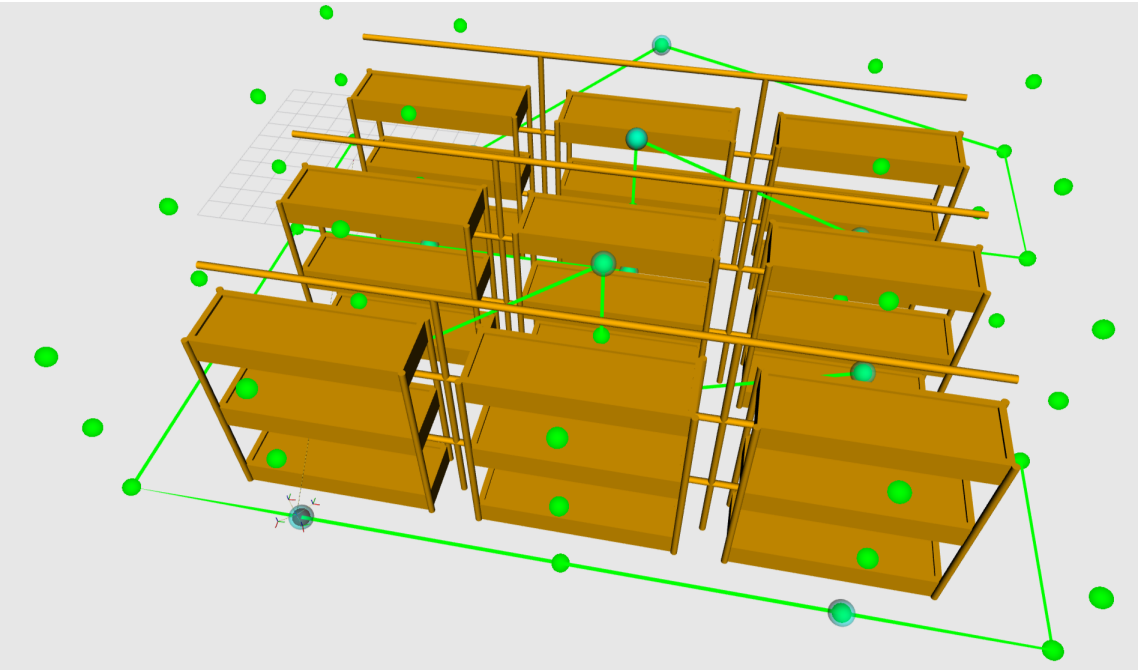


Figure 8.27: Visualization of the generated graph (green spheres) and the computed path (green lines).

8.5.3 Results

The system was tested in simulation with a variety of inspection requests, as shown in Table 8.6. For inspecting six different plants, the average completion time was approximately 225 seconds, covering a total distance of about 140 meters, resulting in an average inspection speed of 0.6 m/s.

Plants IDs	**Completion time (s)**	**Path length(m)**
Eggplant 1 6 15 17 19 25	233.0	143.7
Pepper 5 13 14 21 23 24	226.0	141.7
Banana 4 14 13 22 24 25	223.0	140.6
Pepper 10 11 17 20 21 23	222.0	140.5

Table 8.6: Evaluation results comparing different plants to detect in different beds.

Our proposed approach consistently and robustly traverse the beds, and count the fruits precisely. However, we ended up the 9th team of the competition due to the speed of our proposal. Our approach was the one that better punctuation recevied without failing in any scenario, nevertheless the organizers only considered the best of four results [5].

[5] https://github.com/larics/icuas24_competition/discussions/52

Chapter 9

Results of Aerostack2 as an Open-Source Project

One of the principal contributions of this thesis is the *Aerostack2 framework* as an open-source project aimed at easing the development of aerial systems for a broader audience while providing a set of well-tested and validated components that accelerate the development of real systems or research in specific fields. In this chapter, we analyze the results obtained during this period and validate whether the proposed objectives have been successfully achieved.

9.1 Open-Source Project Approach

From its inception, Aerostack2 was intended to be "the Navigation2 for Aerial Robotics." This means serving as the reference framework for the development of aerial robotic systems with ROS 2.

Aside from the scientific contributions this framework offers, such as self-supervision, reconfiguration, and metacontrol capabilities, it also provides technical contributions that facilitate its deployment by the broader scientific community.

The first and most important aspect of an open-source project is comprehensive documentation that aids in onboarding new users. Aerostack2 includes public documentation[1] that guides users from the initial installation steps to setting up their first operational project. It also presents the different layers of the framework and provides tutorials on developing new modules based on it.

To simplify installation, Aerostack2 can be installed in multiple ways:

- **Source Install:** Based on ROS community standards, the framework can be downloaded and installed by compiling the source code directly, allowing users to modify any of its components.

[1] `https://aerostack2.github.io/`

- **Container-Based Deployment:** To facilitate testing and usage across different machines, Aerostack2 can be containerized using Docker. We provide nightly images that include the latest modifications to the framework, as well as release images with stable versions.

- **Ubuntu Packages:** Finally, the easiest way to install *Aerostack2* is through binary packages generated by the official ROS 2 build farm, significantly simplifying its use for all users.

Its also worth mentioning that this framework has a Continous Integration (CI) pipeline which has two main purposes:

1. **Quality Assurance**: All the modifications to this framework have to pass through quality tests, about style and code smells. In addition, each change has to be reviewed to avoid decreasing the quality of the overall code.

2. **Automate deployment**: The updated container images are automatically generated and uploaded when a modification is approved, easing the use of containers even for the latest updates or in development time.

Currently, all framework-related code can be found on the Aerostack2 organization's GitHub page[2], which hosts multiple repositories. Focusing on the main repository[3], as of March 2025, there have been more than 11 public releases and over 1,700 contributions from multiple contributors, including individuals from around the world.

To the best of our knowledge, Aerostack2 is currently the largest framework in ROS 2 that enables the development of complete aerial robotic applications from scratch on different platforms in both industrial and scientific fields, while covering all the relevant features analyzed in the state of the art (see Table 2.2).

9.2 Software Package Structure

The Aerostack2 framework is structured into more than 30 different software packages, distributed as follows:

- **Core Libraries:** Libraries that include basic templates for generating specific components such as aerial platforms. They also include all the specific messages and interfaces used for communication between different modules.

- **Framework Utilities:** A set of utilities that facilitate the use of various framework functionalities. A relevant example is the `MotionControllerHandlers` package, which encapsulates a set of components to simplify sending commands to the controller without handling all controller mode configurations.

- **Aerial Platforms Packages:** Implementations of hardware interfaces for different platforms, both simulated and real. Aerostack2 supports more than nine different platforms—two for simulation and the rest for real-world aerial platforms.

- **Motion Control Package:** Contains all motion control-related components, including the Motion Controller Manager and all plugin implementations. Currently, three controllers are implemented: PID, Differential-Based, and MPC.

[2] `https://github.com/aerostack2`
[3] `https://github.com/aerostack2/aerostack2`

- **State Estimation Package:** Includes all state estimation-related components, such as the state estimation manager and its plugins. Currently, three publicly available plugins exist: External Estimation, Ground Truth, and Mocap. A Visual Odometry plugin is under development.
- **Behavior Packages:** Provides templates for developing behaviors. It also includes the implementation of different types of behaviors, categorized into Motion Behaviors and Perception Behaviors.
- **Python API Module:** Contains all the necessary functions for controlling and scheduling the execution of different behaviors, as well as controlling the aircraft when required.
- **Simulation Assets:** A collection of assets that aid in developing simulation environments. This includes environmental assets as well as robotic components such as aerial platforms and custom sensors.
- **User Interface Packages:** Includes tools and utilities for both commanding the drone and supervising its execution. It also provides visualization tools to assist with debugging.
- **Demonstrative Projects:** A set of *Projects* with predefined configurations for specific applications. A *Project* contains all necessary configurations and launch files for a given application, serving as a starting point for new developments.

9.3 System Requirements

The presented framework has been developed on Ubuntu Linux 20.04 and 22.04 and has been tested with the ROS 2 Galactic (EOL) and ROS 2 Humble distributions. The preferred programming language for the components is C++, except for the plan execution and mission execution modules, where Python has been used predominantly. This is because components that need to run in real-time benefit more from using C++, whereas for higher-level layers, real-time constraints are less strict, making Python a convenient option.

Aerostack2 has been run on a wide variety of computers, ranging from high-end laptops to very small and resource-constrained computers, aiming to be suitable for onboard computations. Some of the single-board computers on which the full Aerostack2 framework has been successfully executed include (listed from least to most powerful):

- Raspberry Pi 3 Model B+: 1.2 GHz quad-core ARM Cortex-A53 CPU and 1 GB of RAM
- Nvidia Jetson Nano: 1.43 GHz quad-core ARM Cortex-A57 CPU, 128-core Maxwell GPU, and 4 GB of LPDDR4 RAM
- Nvidia Jetson Xavier NX: 6-core Nvidia Carmel ARMv8.2 CPU, 384-core Volta GPU with 48 Tensor Cores, and 16 GB of LPDDR4x RAM
- Nvidia Jetson Orin NX: 8-core ARM Cortex-A78AE v8.2 CPU, 1024-core Ampere GPU with 32 Tensor Cores, and 16 GB of LPDDR5 RAM
- Nvidia Jetson AGX Xavier: 8-core Nvidia Carmel ARMv8.2 CPU, 512-core Volta GPU with 64 Tensor Cores, and 32 GB of LPDDR4x RAM
- Nvidia Jetson AGX Orin: 12-core ARM Cortex-A78AE v8.2 CPU, 2048-core Ampere GPU with 64 Tensor Cores, and 64 GB of LPDDR5 RAM

Depending on the specific application and the payload capacity of each drone, different computers have been used. For the most resource-limited systems, the execution frequency of certain modules has been adjusted to achieve optimal overall performance.

9.4 Scientific Research Built with Aerostack2

Since the first beta release of the Aerostack2 framework in 2021, it has been used for various research projects. This section presents a selection of these works, demonstrating the .

Specific applications include studies by Luna et al. (Luna et al., 2022), Arias-Perez et al. (Arias-Perez et al., 2024), and Mejias et al. (Mejias et al., 2024), which focused on indoor environments using homogeneous swarms of nanodrones with offboard processing. These works investigated swarm control and real-time replanning with motion capture-based localization. Outdoor industrial scenarios were explored by Perez-Segui et al. (Perez-Segui et al., 2023) and Melero-Deza et al. (Melero-Deza et al., 2024), who demonstrated heterogeneous UAV swarms with onboard processing for vision-based inspections controlled via a graphical user interface (GUI). Readers are encouraged to consult these studies for detailed insights and quantitative evaluations of the framework's performance.

Table 9.1 summarizes each study, detailing the number of drones used, the platforms employed, and the specific testing conditions. This overview highlights Aerostack2's scalability, demonstrated through simulations involving up to seven drones and real-world tests with up to four drones, as well as its adaptability across diverse mission specifications and aerial platforms. Through these studies, the Aerostack2 framework has been validated across diverse research applications, providing quantitative evidence of its capabilities and effectiveness.

9.5 Communication and Dissemination Activities

To promote the use of this framework within the robotics community, several communication and dissemination activities have been carried out. The most relevant ones are:

1. **Presentation within the ROS Aerial Robotics Workgroup:** This ROS workgroup consists of various individuals involved in the use of ROS 2 for aerial robotics, both from industry and research institutions. It serves as an excellent platform for connecting with potential new users and contributors.[4]

2. **Talk at ROSCon 2023 Madrid:** ROSCon Madrid was a national-level conference that brought together robotics developers across Spain, increasing the framework's visibility at the national level.[5]

3. **Talk at ROSCon 2023 New Orleans:** ROSCon is the official conference of the international ROS community. Having the opportunity to present our work there significantly contributed to global outreach.[6]

4. **Aerial Swarm Tools and Applications Workshop at RSS 2024:** Beyond the ROS community, the Aerostack2 framework was presented in a workshop at the well-known Robotics: Science and Systems (RSS) International Conference. This provided an opportunity to introduce the framework to a broader audience not necessarily familiar with the ROS ecosystem.[7]

[4] https://www.youtube.com/watch?v=cVAzlew_hnI&ab_channel=Bitcraze
[5] https://www.youtube.com/watch?v=9YS7so52e7U&ab_channel=ROSConEspa
[6] https://vimeo.com/879000655
[7] https://imrclab.github.io/workshop-aerial-swarms-rss2024/

Application Domain	Tested In	# Drones	Platform	Basic Robotic Functions	Mission Control
(Luna et al., 2024) Multi-UAV area coverage with online replanning	S, RL	3 (5)	CF (OFB)	MOCAP – PID	Mission Interpreter
(Arias-Perez et al., 2024) Multi-UAV exploration of unknown cluttered environments	S, RL	3 (7)	CF (OFB)	MOCAP – PID	Python API
(Perez-Segui et al., 2023) Sim2Real approach for autonomous UAV industrial inspections	S, RO	1 (1)	DJI PSDK (OB)	GPS – PID	GUI, Mission Interpreter
(Melero-Deza et al., 2024) On-demand monitoring of a PV plant using UAVs and IoT	S, RO	2	DJI PSDK (OB)	GPS – PID	Mission Interpreter
(Mejias et al., 2024) Swarm formation control with decentralized collision avoidance	RL	4	CF (OFB)	MOCAP – PID	Python API
(Rutinowski et al., 2023) UAV for re-identification of entities in an industrial setting	RL	1	CF (OFB)	MOCAP – PID	Python API

S: Simulation; RL: Real experiments in the laboratory; RO: Real experiments outside the laboratory; CF: Crazyflie; OFB: Offboard computation; OB: Onboard computation; MOCAP: Motion Capture System;

Table 9.1: Collection of applications and use cases where Aerostack2 is utilized. All simulations were performed in Gazebo. The platform used is followed by the computation method in brackets. The number of drones used in simulation appears in brackets. Both GPS and VIO estimation use an external localization plugin that adapts sensor measurements into Aerostack2.

Attending these events has helped increase visibility and provided opportunities to exchange insights on the strengths and weaknesses of the software with other colleagues, ultimately contributing to further improvements to the framework.

Chapter 10

Discussion and Conclusions

10.1 Discussion of the obtained results

Throughout this doctoral work, several results have been presented. In this section, a discussion of these results is provided. Although some of the results obtained from the proposed algorithm have already been discussed in their corresponding chapters, a brief summary of them has also been included here for completeness.

Discussion of Architectural Results: A Reference Architecture and a Software Framework for Robotic Systems

In Chapter 3, a layered architecture for developing autonomous robotic systems was presented. The proposed architecture categorizes different types of systems based on how the decision-making process occurs and what information is used for it. It introduces definitions that establish the foundation for developing robotic systems that leverage *Understanding* and *Awareness* to enhance their robustness.

This reference architecture has guided the layered design of Aerostack2, a software framework for developing autonomous aerial robotic systems. Although the concepts of *Understanding* and *Awareness* have been introduced in the extended version of this framework, these new capabilities have barely been tested within it. However, this establishes the foundation for further improving these processes and testing them in more complex scenarios to effectively evaluate their impact.

Collective Understanding capabilities for MRS have been explored through the development of the *Multi S-Graph* system, which exploits the semantic relational model built by each agent to generate an emergent collective map of the environment, outperforming other state-of-the-art algorithms. However, the implications of this emergent model in terms of mission completion have not been considered in this work.

Regarding the *Aerostack2* framework presented in Chapter 4, numerous applications have been developed and described throughout this doctoral work. Over the course of this research, more than

ten different systems have been successfully developed. The wide range of applications demonstrates the versatility of the proposed framework. In the scientific community, Aerostack2 has been used to develop flocking algorithms, multi-robot exploration strategies, and logistic applications by incorporating new components while relying on pre-existing robotic functions. This approach allows researchers to focus on developing novel algorithms, demonstrating the framework's utility in facilitating research in aerial robotics.

Beyond these applications, Aerostack2 has also been employed in international competitions and industrial scenarios. By combining these diverse use cases, valuable insights have been gathered to validate the framework. Specifically, the majority of systems relied on the Python API to design their missions, highlighting its flexibility in adapting missions to specific applications. Additionally, the sim-to-real capabilities have been validated multiple times, confirming that the level of abstraction across different layers effectively minimizes the sim-to-real gap.

The maximum number of drones tested simultaneously was seven in simulation and four in real-world applications, though these tests were conducted with small drones. In industrial applications, only two drones have been tested flying simultaneously due to increased risk, complexity, and legal constraints associated with deploying these systems in real-world scenarios.

In terms of reusability, several components—such as basic motion behaviors and the PID motion controller—have been used across multiple applications without modification. This was possible even across different platforms, thanks to the adaptation and metacontrol capabilities developed within the framework.

Discussion of Fast Trajectory Modification

In Chapter 5, a novel and efficient method for dynamically recomputing trajectories for quadrotors was proposed, based on the use of an innovative concept: the Local Gaussian Modifiers (LGMs). To validate this approach, isolated and performance tests were conducted before testing the proposed algorithm in both simulated and real flight environments. From the results obtained, we observe the following:

In the first testbed, modifying the trajectory with LGMs enabled smooth waypoint transitions with continuous position, velocity, and acceleration references (Figures 5.5 and 5.6). However, it introduced significant acceleration peaks, which could result in unfeasible references.

Profiling tests showed that modifying a trajectory with LGMs is over three orders of magnitude faster than generating a new polynomial trajectory. The time reduction ranged from 10^5 to 10^3 times for a short trajectory (six points) and from 10^6 to 10^4 times for a longer one (26 points), consistent across both a laptop and an onboard computer. From this, we observe that the time reduction is significant and that the proportion remains consistent when comparing computers with different computing capabilities. This suggests that a similar reduction ratio could be maintained even with less capable microprocessors.

Simulated experiments confirmed that with short computation times (e.g., on a laptop), the system could continuously regenerate trajectories upon waypoint changes and stitch them smoothly. However, when simulating lower-performance hardware—by artificially increasing trajectory computation times—real-time trajectory generation was no longer feasible, necessitating the use of LGMs for modifications. This became critical at higher speeds, allowing the drone to exceed 14 m/s while consistently passing through moving gates. Our approach allowed the drone to successfully navigate gates, performing comparably to full trajectory generation up to 15 m/s when the trajectory references remained within its actuation limits. However, near maximum actuation limits, reference speeds exceeded acceleration constraints, reducing success rates.

In real-world tests, the proposed algorithm enabled successful navigation at peak speeds of 4 m/s and an average of 1 m/s. However, inaccuracies in gate positioning and localization er-

rors—worsening with speed—affected accuracy. Unlike in simulations, the smaller real-world flight area limited the time available for gate position estimation, leading to less precise measurements. Odometry shifts, particularly with the rear-facing RealSense T265, caused abrupt image capture changes as the drone passed through gates. Higher speeds were unattainable in real-world tests due to accumulating inaccuracies, which introduced noise in trajectory tracking and led to crashes.

Discussion of Collaborative Semantic-Relational SLAM

In Chapter 6, a novel LiDAR-based distributed SLAM system designed for structured environments is presented. It leverages a factor graph that integrates a pose graph with a 3D scene graph containing semantic-relational concepts. This graph encodes essential knowledge for extracting key information required for inter-robot communication and integrates data from other agents, forming a Multi-Robot Situational Graph (*Multi S-Graphs*).

In simulations, the *Multi S-Graph* algorithm significantly outperformed decentralized CSLAM methods, reducing trajectory errors by over 90% compared to DCL-SLAM, Disco-SLAM, and Swarm-SLAM, and by 72.9% compared to LAMP 2.0, despite the latter requiring precise initial robot poses. While measurement errors arose due to sparse environmental features, extracting semantic information helped mitigate their impact. The method also drastically reduced inter-agent data exchange, lowering communication overhead while maintaining accuracy.

Real-world experiments confirmed these advantages, with substantial improvements in map matching accuracy over existing approaches. Although semantic feature extraction was slightly less accurate due to added environmental complexity, leveraging room-wall structures and point clouds ensured superior performance. The average reconstruction error remained low, with minor variations across different conditions.

An ablation study highlighted the impact of hierarchical optimization, which establishes spatial relationships between rooms and walls. This improved accuracy by 25.6% but required a 77% increase in data exchange. Fine alignment by using ICP, further enhanced performance by up to 74.7%, though at the cost of significantly higher bandwidth usage. While fine alignment techniques provided the best accuracy, their high data demands made them less practical in constrained communication scenarios. In such cases, using semantic information combined with the Scan Context, without transmitting point clouds, offered a viable alternative with minimal accuracy loss.

Despite the benefits of graph-based information, point cloud transmission for improving the aligment at a room level remained the primary contributor to increased data volume. While the growth in graph data during exploration was manageable, optimizing the balance between accuracy and communication efficiency still crucial for scalable deployment.

10.2 Conclusions

The rapid evolution of robotic systems has led to their widespread adoption across various fields, from industrial automation to specialized applications such as autonomous industrial inspection. Among these, aerial robots have played a crucial role, offering high mobility and versatility for tasks ranging from infrastructure inspection to search and rescue. However, despite advancements in AI and sensor technologies, many robotic systems still face challenges in achieving full autonomy, particularly in complex and dynamic environments.

Although open-source frameworks for aerial robotics have already been developed, they still exhibit certain weaknesses. These limitations primarily stem from their specificity to particular platforms or use cases, restricting their applicability to other domains. Additionally, existing

frameworks often present challenges in terms of extensibility and scalability and lack the capability to acquire and utilize knowledge to enhance decision-making processes.

This doctoral research addresses these challenges in the development of autonomous robotic systems by advancing system architectures and software frameworks, with the goal of improving the autonomy, adaptability, and collaboration of robotic agents, particularly in aerial robotics.

Beyond architectural research, this thesis also elaborates on the design of novel algorithms aimed at tackling a key issue: scalability. The research explores this challenge from two perspectives. One approach focuses on miniaturizing robots by developing computationally efficient trajectory planning methods for UAVs. The other aims to increase the size of robot fleets by designing a robust and efficient method for generating collaborative maps among multiple robots.

The contributions of this doctoral work align with the research on these scientific and technological challenges and can be categorized into four blocks: (1) advances in the development of novel robotic architectures and frameworks, (2) the improvement in the efficiency of two relevant algorithms for robotic systems, providing novel approaches, (3) the validation of the results of this thesis in both particular cases (for the evaluation of the concrete algorithms) and general cases (for the evaluation of the architectural research), and (4) the dissemination of this work, with special relevance to the Aerostack2 Open-Source Project.

Advances in Robotic Architectures

The main topic of this thesis has been to contribute to the development of robotic systems for different situations, accounting for both simple and highly complex scenarios. This work has contributed with:

C.1 A Reference Architecture for Autonomous Robotic Systems: A novel layered reference architecture was developed, providing an incremental framework that structures robotic system capabilities from basic control layers to complex cognitive and multi-agent decision-making. The concepts of **Understanding and Awareness** were formally introduced, emphasizing their role in improving situational adaptability and decision-making in both single and multi-robot systems.

C.2 A Software Framework for the Development of Robotic Aerial Systems: A novel, comprehensive, modular, and scalable software framework was developed to support the deployment of aerial robotic systems with various configurations and mission requirements. This framework excels in enabling the development of aerial systems for a wide range of application domains and different aerial platforms. Recent advancements in this framework lay the foundation for extending it into a cognitive system, supporting knowledge acquisition and exploitation at runtime.

Algorithmic Research

Algorithmic research presents improvements for some of the algorithms that drive robotic behavior, making them more robust and efficient in two key areas: trajectory generation and Collaborative SLAM.

C.3 Dynamic Trajectory Generation Algorithm: A fast and efficient algorithm for modifying UAV trajectories in real-time was proposed, enabling agile maneuvering in dynamic environments while maintaining computational efficiency. The algorithm was extensively tested in both simulated and real-world drone racing scenarios, proving its capability for high-speed trajectory adjustments.

C.4 Collaborative SLAM for Multi-Robot Systems: A novel distributed SLAM approach was developed based on situational graphs, integrating semantic information to improve inter-robot map merging and localization accuracy. The method demonstrated improved robustness and efficiency in real-world multi-robot experiments, reducing communication overhead while enhancing localization performance. Although the experiments were carried out using quadrupeds, the approach is compatible with any robot equipped with a LiDAR and an odometry source, such as an UAV.

Validation and Open-Source Contributions

Validation and open-source contributions ensure that the developed methods are thoroughly tested and made accessible to the research community for further advancements.

C.5 Extensive Real-World Validation: The proposed frameworks and algorithms were evaluated in international competitions, demonstrating their performance in real-world and challenging robotic scenarios. Additionally, some outcomes of this thesis were applied to industrial projects, showcasing the effectiveness of the developed methods in UAV-based inspection tasks, such as photovoltaic panel and windmill inspections.

C.6 Open-Source Contributions: Open-source releases of the developed implementations foster further research collaboration and real-world adoption of the proposed solutions. In particular, Aerostack2 has been conceived as an open-source project aimed at connecting and supporting researchers worldwide. Dissemination and communication efforts have been carried out to achieve this objective, yielding promising results.

Through these contributions, this thesis has advanced the understanding and implementation of **autonomous robotic systems**, particularly in the domains of multi-robot collaboration, trajectory optimization, and aerial robotic system architectures. The integration of perception, deliberation, and control into a cohesive system design has proven to be an essential step toward achieving reliable autonomy in real-world applications.

10.3 Future Work

While this research has made significant strides in enhancing autonomous robotic systems, several open challenges and future directions remain. One key area for further development is enhancing awareness in multi-robot systems. Future research should expand the awareness model developed in this work by incorporating cognitive architectures that enable robots to reason about past experiences and anticipate future events. The integration of hybrid AI models, which combine symbolic reasoning with deep learning, could further improve decision-making in autonomous robotic systems. Additionally, investigating explainable AI techniques for robotic decision processes could enhance interpretability and trust in autonomous systems, fostering more reliable interactions between autonomous agents and human operators.

Building upon these advancements in cognitive capabilities, it is essential to strengthen the software frameworks that support robotic autonomy. The presented software framework still consists of a limited set of capabilities, and its evolution requires incorporating more predefined behaviors and algorithms to improve mission planning flexibility and adaptability. Enhancing knowledge integration will allow UAVs to better utilize prior information and learned experiences for more efficient decision-making. Furthermore, integrating state-of-the-art algorithms for localization, SLAM, and control is crucial for ensuring that Aerostack2 evolves into a predominant framework

in robotic system development, enabling more robust and intelligent multi-robot coordination.

A critical aspect of improving robotic adaptability lies in trajectory optimization, particularly for high-speed applications. Developing computationally efficient methods for real-time trajectory adaptation will be a game-changer in agile drone operations. The use of neural models can facilitate the generation of horizon-limited trajectories that dynamically adjust to environmental conditions without the need for explicit trajectory recomputation. Additionally, the proposed algorithm shall be tested on highly computationally restricted hardware, such as the Crazyflie microprocessor, to truly validate its potential. By integrating these advancements within the Aerostack2 framework, UAVs could achieve greater autonomy in high-speed and unpredictable scenarios.

At the same time, improving scalability in Collaborative SLAM is vital for enabling large-scale multi-robot deployments. The current approach relies on specific semantic-relational structures, such as four-walled rooms in building-like environments, which may not always be easy to extract. Room matching depends on these structures and the detection of static features, which limits performance in feature-sparse or highly dynamic environments. To overcome these challenges, future research should focus on developing more generalized representations that enhance SLAM scalability across diverse operational settings. Additionally, integrating edge computing and cloud-based processing could enhance real-time performance, reducing the computational burden on individual robots.

Ultimately, advancing the Technology Readiness Level (TRL) of these research outcomes is a crucial step toward real-world deployment. Transitioning these developments into high-TRL applications requires close collaboration with industry partners to implement UAV-based solutions in infrastructure inspection, agriculture, and logistics. Conducting extensive field tests in unstructured and unpredictable environments will help refine and improve the robustness of the proposed approaches, ensuring their effectiveness in real operational conditions. By addressing these interconnected challenges, future research can push autonomous robotic systems closer to widespread, reliable, and intelligent deployment in complex real-world scenarios.

Appendices

Appendix A

Scientific Dissemination

A.1 Journal Publications

2024

P. Arias-Perez, A. Gautam, **M. Fernandez-Cortizas**, D. Perez-Saura, S. Saripalli, and P. Campoy, "Exploring Unstructured Environments Using Minimal Sensing on Cooperative Nano-Drones," IEEE Robotics and Automation Letters, 9(12), 11202-11209. https://doi.org/10.1109/LRA.2024.3486212. JCR (2024): 4.6

M. Fernandez-Cortizas, H. Bavle, D. Perez-Saura, J. L. Sanchez-Lopez, P. Campoy, and H. Voos, "Multi S-Graphs: An Efficient Distributed Semantic-Relational Collaborative SLAM," IEEE Robotics and Automation Letters, 9(6), 6004-6011. https://doi.org/10.1109/LRA.2024.3399997. JCR (2024): 4.6

J. Rodriguez-Vazquez, I. Prieto-Centeno, **M. Fernandez-Cortizas**, D. Perez-Saura, M. Molina, and P. Campoy, (2024). "Real-Time Object Detection for Autonomous Solar Farm Inspection via UAVs." Sensors, 24(3), 777. https://doi.org/10.3390/s24030777. JCR (2024): 3.4

2023

R. Perez-Segui, P. Arias-Perez, J. Melero-Deza, **M. Fernandez-Cortizas**, D. Perez-Saura, and P. Campoy, (2023). "Bridging the Gap between Simulation and Real Autonomous UAV Flights in Industrial Applications." Aerospace, 10(9), 814. https://doi.org/10.3390/aerospace10090814. JCR (2023): 2.1

D. Perez-Saura, **M. Fernandez-Cortizas**, R. Perez-Segui, P. Arias-Perez, and P. Campoy, (2023). "Urban Firefighting Drones: Precise Throwing from UAV." Journal of Intelligent & Robotic Systems, 108(4), 66. https://doi.org/10.1007/s10846-023-01883-6. JCR (2023): 3.1

J. Rodriguez-Vazquez, **M. Fernandez-Cortizas**, D. Perez-Saura, M. Molina, and P. Campoy, (2023). "Overcoming Domain Shift in Neural Networks for Accurate Plant Counting in Aerial Images." Remote Sensing, 15(6), 1700. https://doi.org/10.3390/rs15061700. JCR (2023): 4.2

S. Awasthi, **M. Fernandez-Cortizas**, C. Reining, P. Arias-Perez, M. A. Luna, D. Perez-Saura, M. Roidl, N. Gramse, P. Klokowski, and P. Campoy, (2023). "Micro UAV Swarm for Industrial Applications in Indoor Environment: A Systematic Literature Review." Logistics Research, 16(1), 1-43. https://doi.org/10.23773/2023_11 SJR (2023): 0.368

2022

A. Rodriguez-Ramos, A. Alvarez-Fernandez, H. Bavle, J. Rodriguez-Vazquez, L. Lu, **M. Fernandez-Cortizas**, R. A. S. Fernandez, A. Rodelgo, C. Santos, M. Molina, L. Merino, F. Caballero, and P. Campoy, (2022). "Autonomous Aerial Robot for High-Speed Search and Intercept Applications." Field Robotics, 2, 1320-1350. https://doi.org/10.55417/fr.2022044

M. Fernandez-Cortizas, D. Perez-Saura, P. Santamaría, J. Rodriguez-Vazquez, M. Molina, and P. Campoy, (2022). "Framework and Evaluation Methodology for Autonomous Drone Racing." Unmanned Systems, 10(4), 355-367. https://doi.org/10.1142/S2301385022410035

A.2 Peer-reviewed conference publications

2024

J. Melero-Deza, R. Perez-Segui, P. Arias-Perez, **M. Fernandez-Cortizas**, D. Perez-Saura, G. GP-Lenza, M. Tradacete-Agreda, C. S. Pérez, F. J. R. Sánchez, and P. Campoy, "Photovoltaic Plant Monitoring and Inspection through Synergic Integration of UAVs and IoT." 15th Annual International Micro Air Vehicle Conference and Competition, Bristol, United Kingdom, 2024, pp. 185–192.

G. GP-Lenza, C. DR. Pita-Romero, **M. Fernandez-Cortizas**, and P. Campoy, "A Methodology for Designing Knowledge-Driven Missions for Robots." 2024 7th Iberian Robotics Conference (ROBOT), Madrid, Spain, 2024, pp. 1-6. https://doi.org/10.1109/ROBOT61475.2024.10796896.

M. Fernandez-Cortizas, D. Perez-Saura, R. Perez-Segui, J. Rodriguez-Vazquez, J. S. Cely, and P. Campoy, "Local Gaussian Modifiers (LGMs): UAV Dynamic Trajectory Generation for Onboard Computation." 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania - Crete, Greece, 2024, pp. 1101-1108. https://doi.org/10.1109/ICUAS60882.2024.10556985.

M. Fernandez-Cortizas, D. Perez-Saura, R. Sanz, M. Molina, and P. Campoy, (2024). "The Landscape of Collective Awareness in Multi-Robot Systems." European Robotics Forum 2024, Springer, Cham. https://doi.org/10.1007/978-3-031-76424-0_17.

2023

M. A. Luna, M. S. A. Isaac, **M. Fernandez-Cortizas**, C. Santos, A. R. Ragab, M. Molina, and P. Campoy, (2023). "Spiral Coverage Path Planning for Multi-UAV Photovoltaic Panel Inspection Applications." 2023 International Conference on Unmanned Aircraft Systems (ICUAS), Warsaw, Poland, 2023, pp. 679-686. https://doi.org/10.1109/ICUAS57906.2023.10156085.

2021

M. Fernandez-Cortizas, P. Santamaría, D. Perez-Saura, J. Rodriguez-Vazquez, M. Molina, and P. Campoy, (2021). "Framework and Evaluation Methodology for Autonomous Drone Racing." 12th International Micro Air Vehicle Conference, Puebla, México, 2021, pp. 50–56.

A.3 Preprints and Under-Review Manuscripts

M. Fernandez-Cortizas, M. Molina, P. Arias-Perez, R. Perez-Segui,D. Perez-Saura, and P. Campoy (2023). "Aerostack2: A software framework for developing multi-robot aerial systems". arXiv preprint arXiv:2303.18237.

Appendix B

Aerial Platforms Used in This Thesis

Throughout this thesis, multiple real-world experiments have been conducted using a variety of aerial platforms. In this appendix, a brief description of each platform is provided, ordered from the smallest and lightest to the largest and most powerful.

B.1 Crazyflie 2.1

The Crazyflie 2.1 is a nano quadcopter developed by Bitcraze, a Swedish company focused on creating open-source robotics platforms for research and education. Originally designed as a tool for prototyping and academic research, the Crazyflie 2.1 is particularly useful due to its compact size, modular expandability, and strong community support. It enables researchers to experiment with swarming behaviors, vision-based navigation, and autonomous flight in constrained indoor environments.

Figure B.1: Crazyflie 2.1

Bitcraze Crazyflie 2.1	
Dimensions	7.3 x 7.3 x 2.9 cm
Weight	27 g
MTOW	42 g
Max Speed	2.5 m/s
Estimated Flight Time	7 min
Battery	LiPo 1S 250 mAh
Autopilot	N/A
Sensors	AI Deck – RGB 256×256 @10Hz Camera Flow Deck – VO Sensor + Altimeter Laser

Table B.1: Specifications of the Bitcraze Crazyflie 2.1

B.2 Ryze Tello

The Ryze Tello is a compact, lightweight drone developed by Ryze Robotics in collaboration with DJI and Intel. Designed as an educational and recreational drone, it serves as an accessible platform for beginners and researchers alike. Its integration of DJI's flight technology and Intel's vision processing capabilities makes it an excellent tool for learning programming, experimenting with computer vision, and exploring autonomous flight applications.

Figure B.2: Ryze Tello

Ryze Tello	
Dimensions	17.1 x 17.1 x 4.1 cm
Weight	80 g
MTOW	100g
Max Speed	1 m/s
Estimated Flight Time	13 min
Battery	LiPo (3.7 V, 1100 mAh)
Autopilot	Integrated flight controller
Sensors	Vision positioning system, IMU, Barometer, 5MP camera

Table B.2: Specifications of the Ryze Tello

B.3 F330 Custom quadcopter

The F330 Custom Quadcopter is a custom-built quadrotor designed for research applications, offering flexibility in both onboard computing and payload customization. Equipped with an NVIDIA Jetson Xavier NX and a Pixhawk 4 Mini flight controller, this platform supports real-time processing and autonomous navigation. Its modular design allows for the integration of various sensors and cameras, making it an ideal choice for experiments.

Figure B.3: Custom F330 based quadrotor equipped with an NVIDIA Jetson Xavier NX, a Pixhawk 4 Mini, an Intel RealSense T265, and a USB fisheye camera.

F330 Platform	
Dimensions	35x35x20 cm
Weight	1.42 Kg
MTOW	2 Kg
Max Speed	10 m/s
Estimated Flight Time	10-13 min
Battery	LiPo 4S 5000 MAh
Autopilot	Pixhawk 4 Mini
Sensors	Intel RealSense T265 VIO Sensor USB Fisheye Camera 1080p@30Hz
Offboard Computer	Nvidia Jetson NX

Table B.3: Specifications of the Custom F330 Platform

B.4 DJI Matrice 210 V2 RTK

The DJI Matrice 210 V2 RTK is a rugged industrial-grade quadcopter designed for high-precision mapping, surveying, and inspection tasks. Equipped with an RTK (Real-Time Kinematic) module, it provides centimeter-level positioning accuracy, making it ideal for applications requiring high-precision navigation. This platform features an integrated flight controller, multiple sensor redundancies, and a weather-resistant design, allowing for reliable operation in harsh environments. For onboard computing capabilities, we used a Nvidia Jetson AGX Xavier Single Board Computer (SBC).

Figure B.4: DJI Matrice 210 V2 Equipped with the Nvidia Jetson AGX XavierSBC

DJI Matrice 210 V2 RTK	
Dimensions	59.5 × 59.5 × 24.5 cm
Weight	4.6 kg
MTOW	6.3 kg
Max Speed	18.1 m/s
Estimated Flight Time	38 min
Battery	2x TB55 (174.6 Wh)
Autopilot	Integrated DJI flight controller with RTK
Sensors	GPS, RTK module, IMU, Barometer, Ultrasonic sensor

Table B.4: Specifications of the DJI Matrice 210 V2 RTK

B.5 DJI Matrice 300 RTK

The DJI Matrice 300 RTK is a next-generation industrial drone designed for long-endurance operations and advanced autonomous capabilities. Compared to the Matrice 210 V2 RTK, it features an extended flight time, improved weather resistance, and an upgraded obstacle avoidance system. This platform supports multiple payload configurations, including thermal cameras, LiDAR, and multi-spectral sensors, making it suitable for applications in infrastructure inspection, precision agriculture, and public safety. Similarly to the Matrice 210 , for onboard computing capabilities, we used a Nvidia Jetson AGX Xavier Single Board Computer (SBC).

Figure B.5: DJI Matrice 300 RTK Equipped with the Nvidia Jetson AGX XavierSBC

DJI Matrice 300 RTK	
Dimensions	81 × 67 × 43 cm
Weight	6.3 kg (airframe only)
MTOW	9 kg
Max Speed	23 m/s
Estimated Flight Time	55 min
Battery	2 × TB60 intelligent flight batteries (52.8 V x 5935 mAh)
Autopilot	Integrated DJI flight controller with RTK
Sensors	RTK module, GPS, IMU, Barometer, Obstacle Avoidance sensors

Table B.5: Specifications of the DJI Matrice 300 RTK

B.6 DJI Matrice 350 RTK

The DJI Matrice 350 RTK is the latest iteration in the Matrice series, offering improved durability, flight performance, and operational efficiency compared to the Matrice 300 RTK. With increased payload capacity and enhanced obstacle avoidance, it is well-suited for high-risk and high-precision missions, including search and rescue, industrial inspections, and environmental monitoring. It also integrates an advanced vision-based obstacle avoidance system, further increasing safety during autonomous operations.

DJI Matrice 350 RTK	
Dimensions	89 × 75 × 45 cm
Weight	8.5 kg (airframe only)
MTOW	9.2 kg
Max Speed	26 m/s
Estimated Flight Time	55 min
Battery	2 × TB60 intelligent flight batteries (52.8 V x 5935 mAh)
Autopilot	Integrated DJI flight controller with RTK
Sensors	RTK module, GPS, IMU, Barometer, Vision-based Obstacle Avoidance

Table B.6: Specifications of the DJI Matrice 350 RTK

Bibliography

Progress on competitiveness of clean energy technologies. Technical report, European Comission, 2023.

ALBUS, J. S. Outline for a theory of intelligence. *IEEE transactions on systems, man, and cybernetics*, vol. 21(3), pages 473–509, 1991.

ANGLE, C. *Genghis, a six legged autonomous walking robot*. PhD Thesis, Massachusetts Institute of Technology, 1989.

ARANDJELOVIĆ, R., GRONAT, P., TORII, A., PAJDLA, T. and SIVIC, J. Netvlad: Cnn architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40(6), pages 1437–1451, 2018.

ARIAS-PEREZ, P., GAUTAM, A., FERNANDEZ-CORTIZAS, M., PEREZ-SAURA, D., SARIPALLI, S. and CAMPOY, P. Exploring unstructured environments using minimal sensing on cooperative nano-drones. *IEEE Robotics and Automation Letters*, vol. 9(12), pages 11202–11209, 2024.

AWASTHI, S., FERNANDEZ-CORTIZAS, M., REINING, C., ARIAS-PEREZ, P., LUNA, M. A., PEREZ-SAURA, D., ROIDL, M., GRAMSE, N., KLOKOWSKI, P. and CAMPOY, P. Micro uav swarm for industrial applications in indoor environment: a systematic literature review. *Logistics Research*, vol. 16(1), pages 1–43, 2023.

AZZAM, R., TAHA, T., HUANG, S. and ZWEIRI, Y. Feature-based visual simultaneous localization and mapping: A survey. *SN Applied Sciences*, vol. 2, pages 1–24, 2020.

BACA, T., PETRLIK, M., VRBA, M., SPURNY, V., PENICKA, R., HERT, D. and SASKA, M. The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, vol. 102(1), page 26, 2021.

BAVLE, H., SANCHEZ-LOPEZ, J. L., SHAHEER, M., CIVERA, J. and VOOS, H. Situational graphs for robot navigation in structured indoor environments. *IEEE Robotics and Automation Letters*, vol. 7(4), pages 9107–9114, 2022.

BAVLE, H., SANCHEZ-LOPEZ, J. L., SHAHEER, M., CIVERA, J. and VOOS, H. S-graphs+: Real-time localization and mapping leveraging hierarchical representations. *IEEE Robotics and Automation Letters*, vol. 8(8), pages 4927–4934, 2023.

BERTON, G., MASONE, C. and CAPUTO, B. Rethinking visual geo-localization for large-scale applications. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

BOCHKOVSKIY, A., WANG, C.-Y. and LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

BOUABDALLAH, S. and SIEGWART, R. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158. 2007.

BROOKS, R. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, vol. 2(1), pages 14–23, 1986.

BUEHLER, M., IAGNEMMA, K. and SINGH, S. *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56. Springer Science & Business Media, 2009.

BURRI, M., OLEYNIKOVA, H., , ACHTELIK, M. W. and SIEGWART, R. Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments. In *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*. 2015.

BÄNNINGER, P., ALZUGARAY, I., KARRER, M. and CHLI, M. Cross-agent relocalization for decentralized collaborative slam. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5551–5557. 2023.

CAMPILHO, R. D. S. G. and SILVA, F. J. G. Industrial process improvement by automation and robotics. *Machines*, vol. 11(11), 2023. ISSN 2075-1702.

CAO, Y. and LEE, C. G. Behavior-tree embeddings for robot task-level knowledge. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12074–12080. IEEE, 2022.

CHANG, Y., EBADI, K., DENNISTON, C. E., GINTING, M. F., ROSINOL, A., REINKE, A., PALIERI, M., SHI, J., CHATTERJEE, A., MORRELL, B. ET AL. Lamp 2.0: A robust multi-robot slam system for operation in challenging large-scale underground environments. *IEEE Robotics and Automation Letters*, vol. 7(4), pages 9175–9182, 2022.

CHANG, Y., HUGHES, N., RAY, A. and CARLONE, L. Hydra-multi: Collaborative online construction of 3d scene graphs with multi-robot teams. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10995–11002. IEEE, 2023.

CHEN, H., TAO, J., ZHOU, B. and LIU, X. Research on an autonomous uav search and rescue system based on the improved ego-planner algorithm. In *2024 5th International Conference on Computer Engineering and Application (ICCEA)*, pages 1170–1175. IEEE, 2024.

CHEN, S. and CHEN, H. Mpc-based path tracking with pid speed control for autonomous vehicles. In *IOP Conference Series: Materials Science and Engineering*, vol. 892, page 012034. IOP Publishing, 2020.

COLAPRICO, M., DE RUVO, M. F., LEOTTA, G., BIZZARRI, F., VERGURA, S. and MARINO, F. Dubio: a fully automatic drones & cloud based infrared monitoring system for large-scale pv plants. In *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–5. IEEE, 2018.

COLEMAN, D., SUCAN, I., CHITTA, S. and CORRELL, N. Reducing the barrier to entry of complex robotic software: a moveit! case study. 2014.

COLLEDANCHISE, M. and ÖGREN, P. *Behavior Trees in Robotics and AI*. CRC Press, 2018.

COMMUNITY, B. O. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

CRAMARIUC, A., BERNREITER, L., TSCHOPP, F., FEHR, M., REIJGWART, V., NIETO, J., SIEGWART, R. and CADENA, C. maplab 2.0 – a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, vol. 8(2), page 520–527, 2023. ISSN 2377-3774.

DEUTSCH, I., LIU, M. and SIEGWART, R. A framework for multi-robot pose graph slam. *2016 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2016*, pages 567–572, 2016.

DUBÉ, R., CRAMARIUC, A., DUGAS, D., NIETO, J., SIEGWART, R. and CADENA, C. Segmap: 3d segment mapping using data-driven descriptors. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 2018.

EBEID, E., SKRIVER, M., TERKILDSEN, K. H., JENSEN, K. and SCHULTZ, U. P. A survey of open-source uav flight controllers and flight simulators. *Microprocessors and Microsystems*, vol. 61, pages 11–20, 2018.

ELMOKADEM, T. and SAVKIN, A. V. Towards fully autonomous uavs: A survey. *Sensors*, vol. 21(18), page 6223, 2021.

ENDSLEY, M. R. Situation awareness global assessment technique (sagat). In *Proceedings of the IEEE 1988 national aerospace and electronics conference*, pages 789–795. IEEE, 1988.

ENDSLEY, M. R. Toward a theory of situation awareness in dynamic systems. *Human factors*, vol. 37(1), pages 32–64, 1995.

FOEHN, P., KAUFMANN, E., ROMERO, A., PENICKA, R., SUN, S., BAUERSFELD, L., LAENGLE, T., CIOFFI, G., SONG, Y., LOQUERCIO, A. ET AL. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, vol. 7(67), page eabl6259, 2022.

FOEHN, P., ROMERO, A. and SCARAMUZZA, D. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, vol. 6(56), page eabh1221, 2021.

FOOTE, T. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6. 2013. ISSN 2325-0526.

FREY, K. M., STEINER, T. J. and HOW, J. P. Efficient constellation-based map-merging for semantic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1302–1308. 2019.

FURRER, F., BURRI, M., ACHTELIK, M. and SIEGWART, R. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26054-9.

GALVANE, Q., FLEUREAU, J., TARIOLLE, F.-L. and GUILLOTEL, P. Automated cinematography with unmanned aerial vehicles. *arXiv preprint arXiv:1712.04353*, 2017.

GARRIDO-JURADO, S., MUÑOZ-SALINAS, R., MADRID-CUEVAS, F. and MEDINA-CARNICER, R. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, vol. 51, 2015.

GREVE, E., BÜCHNER, M., VÖDISCH, N., BURGARD, W. and VALADA, A. Collaborative dynamic 3d scene graphs for automated driving. *arXiv preprint arXiv:2309.06635*, 2023.

HUANG, J., TIAN, G., ZHANG, J. and CHEN, Y. On unmanned aerial vehicles light show systems: Algorithms, software and hardware. *Applied Sciences*, vol. 11(16), 2021. ISSN 2076-3417.

HUANG, Y., SHAN, T., CHEN, F. and ENGLOT, B. Disco-slam: Distributed scan context-enabled multi-robot lidar slam with two-stage global-local graph optimization. *IEEE Robotics and Automation Letters*, vol. 7, pages 1150–1157, 2022. ISSN 23773766.

HUBER, F. *A logical introduction to probability and induction*. Oxford University Press, 2018.

HUGHES, N., CHANG, Y. and CARLONE, L. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*. 2022.

INDUSTRIYARC. Unmanned aircraft systems market - by type , by size , by range , by endurance , by energy source , by application , by geography - global opportunity analysis & industry forecast, 2024 - 2030. 2024.

ISO. Iso/iec/ieee international standard - systems and software engineering–system life cycle processes. *ISO/IEC/IEEE 15288:2023(E)*, pages 1–128, 2023.

KARAMAN, S. and FRAZZOLI, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, vol. 30(7), pages 846–894, 2011.

KIM, G. and KIM, A. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018.

KOENIG, N. and HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pages 2149–2154 vol.3. 2004.

KOIDE, K., YOKOZUKA, M., OISHI, S. and BANNO, A. Voxelized gicp for fast and accurate 3d point cloud registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11054–11059. IEEE, 2021.

KUIPERS, B., FEIGENBAUM, E. A., HART, P. E. and NILSSON, N. J. Shakey: from conception to history. *Ai Magazine*, vol. 38(1), pages 88–103, 2017.

LAIRD, J. E., NEWELL, A. and ROSENBLOOM, P. S. Soar: An architecture for general intelligence. *Artificial intelligence*, vol. 33(1), pages 1–64, 1987.

LAJOIE, P.-Y. and BELTRAME, G. Swarm-slam : Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. 2023.

LAJOIE, P. Y., RAMTOULA, B., CHANG, Y., CARLONE, L. and BELTRAME, G. Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, vol. 5, pages 1656–1663, 2020. ISSN 23773766.

LUNA, M. A., ALE ISAAC, M. S., RAGAB, A. R., CAMPOY, P., FLORES PEÑA, P. and MOLINA, M. Fast multi-uav path planning for optimal area coverage in aerial sensing applications. *Sensors*, vol. 22(6), page 2297, 2022.

LUNA, M. A., MOLINA, M., DA-SILVA-GOMEZ, R., MELERO-DEZA, J., ARIAS-PEREZ, P. and CAMPOY, P. A multi-uav system for coverage path planning applications with in-flight re-planning capabilities. *Journal of Field Robotics*, vol. 41(5), pages 1480–1497, 2024.

MACENSKI, S., MARTÍN, F., WHITE, R. and GINÉS CLAVERO, J. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

MAHONY, R., KUMAR, V. and CORKE, P. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, vol. 19(3), pages 20–32, 2012.

MARKOVIC, L., PETRIC, F., IVANOVIC, A., GORICANEC, J., CAR, M., ORSAG, M. and BOGDAN, S. Towards a standardized aerial platform: Icuas'22 firefighting competition. *Journal of intelligent & robotic systems*, vol. 108(3), page 52, 2023.

MEJIAS, L., ARIAS-PEREZ, P., MELERO-DEZA, J. and CAMPOY, P. A virtual spring-damper approach for uav swarm formation and decentralised collision avoidance. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2024.

MELERO-DEZA, J., PEREZ-SEGUI, R., ARIAS-PEREZ, P., FERNANDEZ-CORTIZAS, M., PEREZ-SAURA, D., GUILLERMO, G.-L., TRADACETE-AGREDA, M., PÉREZ, C. S., SÁNCHEZ, F. J. R. and CAMPOY, P. Photovoltaic plant monitoring and inspection through synergic integration of uavs and iot. In *15th annual International Micro Air Vehicle Conference and Competition*, pages 185–192. Bristol, United Kingdom, 2024.

MELLINGER, D. and KUMAR, V. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

MOHTA, K., WATTERSON, M., MULGAONKAR, Y., LIU, S., QU, C., MAKINENI, A., SAULNIER, K., SUN, K., ZHU, A., DELMERICO, J., THAKUR, D., KARYDIS, K., ATANASOV, N., LOIANNO, G., SCARAMUZZA, D., DANIILIDIS, K., TAYLOR, C. J. and KUMAR, V. Fast, autonomous flight in gps-denied and cluttered environments. *Journal of Field Robotics*, vol. 35(1), pages 101–120, 2018.

MUR-ARTAL, R., MONTIEL, J. M. M. and TARDOS, J. D. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, vol. 31(5), pages 1147–1163, 2015.

NOORALISHAHI, P., IBARRA-CASTANEDO, C., DEANE, S., LÓPEZ, F., PANT, S., GENEST, M., AVDELIDIS, N. P. and MALDAGUE, X. P. Drone-based non-destructive inspection of industrial sites: A review and case studies. *Drones*, vol. 5(4), page 106, 2021.

PATEL, M., KARRER, M., BÄNNINGER, P. and CHLI, M. Covins-g: A generic back-end for collaborative visual-inertial slam. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

PATZ, B. J., PAPELIS, Y., PILLAT, R., STEIN, G. and HARPER, D. A practical approach to robotic design for the darpa urban challenge. *Journal of Field Robotics*, vol. 25(8), pages 528–566, 2008.

PENICKA, R., SONG, Y., KAUFMANN, E. and SCARAMUZZA, D. Learning minimum-time flight in cluttered environments. *IEEE Robotics and Automation Letters*, vol. 7(3), pages 7209–7216, 2022.

PENIN, B., SPICA, R., GIORDANO, P. R. and CHAUMETTE, F. Vision-based minimum-time trajectory generation for a quadrotor uav. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6199–6206. IEEE, 2017.

PEREZ-SEGUI, R., ARIAS-PEREZ, P., MELERO-DEZA, J., FERNANDEZ-CORTIZAS, M., PEREZ-SAURA, D. and CAMPOY, P. Bridging the gap between simulation and real autonomous uav flights in industrial applications. *Aerospace*, vol. 10(9), 2023. ISSN 2226-4310.

PICHIERRI, L., TESTA, A. and NOTARSTEFANO, G. Crazychoir: Flying swarms of crazyflie quadrotors in ros 2. *IEEE Robotics and Automation Letters*, vol. 8(8), pages 4713–4720, 2023.

PRABHU, A., LIU, X., SPASOJEVIC, I., WU, Y., SHAO, Y., ONG, D., LEI, J., GREEN, P. C., CHAUDHARI, P. and KUMAR, V. Uavs for forestry: Metric-semantic mapping and diameter estimation with autonomous aerial robots. *Mechanical Systems and Signal Processing*, vol. 208, page 111050, 2024. ISSN 0888-3270.

PRECEDENDE, R. Unmanned aerial vehicle (uav) drones market size, share, and trends 2025 to 2034. 2025.

PREISS, J. A., HONIG, W., SUKHATME, G. S. and AYANIAN, N. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304. IEEE, 2017.

QUIGLEY, M., CONLEY, K., GERKEY, B., FAUST, J., FOOTE, T., LEIBS, J., WHEELER, R., NG, A. Y. ET AL. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, vol. 3, page 5. Kobe, 2009.

RAJ, R. and KOS, A. A comprehensive study of mobile robot: History, developments, applications, and future research perspectives. *Applied Sciences*, vol. 12(14), 2022. ISSN 2076-3417.

REAL, F., TORRES-GONZÁLEZ, A., SORIA, P. R., CAPITÁN, J. and OLLERO, A. Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, vol. 17(4), pages 1–13, 2020.

RICHTER, C., BRY, A. and ROY, N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.

RITTER, F. E., TEHRANCHI, F. and OURY, J. D. Act-r: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 10(3), page e1488, 2019.

ROMERO, A., SUN, S., FOEHN, P. and SCARAMUZZA, D. Model predictive contouring control for near-time-optimal quadrotor flight. *arXiv preprint arXiv:2108.13205*, 2021.

ROMERO-RAMIREZ, F., MUÑOZ-SALINAS, R. and MEDINA-CARNICER, R. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, vol. 76, 2018.

ROSINOL, A., ABATE, M., CHANG, Y. and CARLONE, L. Kimera: an open-source library for real-time metric-semantic localization and mapping. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

RUTINOWSKI, J., SCHUSTER, D., KAUFFMANN, S., ARIAS-PÉREZ, P., POLACHOWSKI, F., GRANERO, M., ROIDL, M. and CAMPOY, P. Exploring the re-identification of industrial entities on autonomous guided vehicles. *Logistics Journal: referierte Veröffentlichungen*, vol. 2023(1), 2023.

SANCHEZ-LOPEZ, J. L., MOLINA, M., BAVLE, H., SAMPEDRO, C., SUÁREZ FERNÁNDEZ, R. A. and CAMPOY, P. A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework. *Journal of Intelligent & Robotic Systems*, vol. 88, pages 683–709, 2017.

SANZ, R., RODRIGUEZ, M., MOLINA, M., ZAMANI, F., AGUADO, E., FERNANDEZ, M., PEREZ, D., GABRIEL, A. and HERNANDEZ, C. State of the art and needs for a theory of understanding and awareness. Deliverable d1.1, The CORESENSE Project, 2023.

SAUNDERS, J., SAEEDI, S. and LI, W. Autonomous aerial robotics for package delivery: A technical review. *Journal of Field Robotics*, vol. 41(1), pages 3–49, 2024.

SCHMUCK, P., ZIEGLER, T., KARRER, M., PERRAUDIN, J. and CHLI, M. Covins: Visual-inertial slam for centralized collaboration. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 171–176. 2021.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, vol. 27(3), pages 379–423, 1948.

SONG, Y., NAJI, S., KAUFMANN, E., LOQUERCIO, A. and SCARAMUZZA, D. Flightmare: A flexible quadrotor simulator. In *Conference on Robot Learning*. 2020.

SONG, Y., STEINWEG, M., KAUFMANN, E. and SCARAMUZZA, D. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212. IEEE, 2021.

SUN, R. The clarion cognitive architecture: Extending cognitive modeling to social simulation. *Cognition and multi-agent interaction*, pages 79–99, 2006.

TANG, Q., XU, Z., YU, F., ZHANG, Z. and ZHANG, J. Dynamic target searching and tracking with swarm robots based on stigmergy mechanism. *Robotics and Autonomous Systems*, vol. 120, page 103251, 2019. ISSN 0921-8890.

TANIGUCHI, T., MURATA, S., SUZUKI, M., OGNIBENE, D., LANILLOS, P., UGUR, E., JAMONE, L., NAKAMURA, T., CIRIA, A., LARA, B. ET AL. World models and predictive coding for cognitive and developmental robotics: frontiers and challenges. *Advanced Robotics*, vol. 37(13), pages 780–806, 2023.

TAO, Y., LIU, X., SPASOJEVIC, I., AGARWAL, S. and KUMAR, V. 3d active metric-semantic slam. *IEEE Robotics and Automation Letters*, vol. 9(3), pages 2989–2996, 2024.

THRUN, S., MONTEMERLO, M., DAHLKAMP, H., STAVENS, D., ARON, A., DIEBEL, J., FONG, P., GALE, J., HALPENNY, M., HOFFMANN, G. ET AL. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, vol. 23(9), pages 661–692, 2006.

TIAN, Y., CHANG, Y., HERRERA ARIAS, F., NIETO-GRANDA, C., HOW, J. P. and CARLONE, L. Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE Transactions on Robotics*, vol. 38(4), page 2022–2038, 2022. ISSN 1941-0468.

VAŠČÁK, J. and HERICH, D. Map merging for multi-robotic applications. In *2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000021–000026. 2023.

WANG, Y., SUN, Z., XU, C.-Z., SARMA, S. E., YANG, J. and KONG, H. Lidar iris for loop-closure detection. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

WU, Y. and LOW, K. H. An adaptive path replanning method for coordinated operations of drone in dynamic urban environments. *IEEE Systems Journal*, vol. 15(3), pages 4600–4611, 2020.

XIAO, K., TAN, S., WANG, G., AN, X., WANG, X. and WANG, X. Xtdrone: A customizable multi-rotor uavs simulation platform. In *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pages 55–61. IEEE, 2020.

YU, J. and SHEN, S. Semanticloop: loop closure with 3d semantic graph matching. *IEEE Robotics and Automation Letters*, vol. 8(2), pages 568–575, 2022.

YU, S., FU, C., GOSTAR, A. K. and HU, M. A review on map-merging methods for typical map types in multiple-ground-robot slam solutions. *Sensors*, vol. 20(23), 2020. ISSN 1424-8220.

ZHONG, S., QI, Y., CHEN, Z., WU, J., CHEN, H. and LIU, M. Dcl-slam: A distributed collaborative lidar slam framework for a robotic swarm. *IEEE Sensors Journal*, 2023.

ZOLLARS, M. D., COBB, R. G. and GRYMIN, D. J. Optimal suas path planning in three-dimensional constrained environments. *Unmanned Systems*, vol. 07(02), pages 105–118, 2019.