
# D5.2 — Drone take-off and landing strategies

## WP5 — Contamination Situation Awareness

October 31, 2021

The PathoCERT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 883484.

## Document Information

| | | | | |
|---|---|---|---|---|
| GRANT AGREEMENT NUMBER | 883484 | ACRONYM | | PathoCERT |
| FULL TITLE | Pathogen Contamination Emergency Response Technologies | | | |
| START DATE | 1st September 2020 | DURATION | | 36 months |
| PROJECT URL | www.pathocert.eu | | | |
| DELIVERABLE | D5.2 — Drone take-off and landing strategies | | | |
| WORK PACKAGE | WP5 — Contamination Situation Awareness | | | |
| DATE OF DELIVERY | CONTRACTUAL | 31/10/21 | ACTUAL | 31/10/21 |
| NATURE | Report | | DISSEMINATION LEVEL | Public |
| LEAD BENEFICIARY | NTUA | | | |
| RESPONSIBLE AUTHOR | Dr. George C. Karras | | | |
| CONTRIBUTIONS FROM | Fotis Panetsos, Panagiotis Rousseas, Dr. George C. Karras, Dr. Charalampos P. Bechlioulis, Prof. Kostas J. Kyriakopoulos (NTUA). Other contributions from CERTH, UCY, CETAQUA, ENG, CSCP, EYATH | | | |
| ABSTRACT | The goal of D5.2, is to increase the aerial platform's autonomy by utilizing feedback from the on-board perception sensors and employ motion control algorithms in order to facilitate safe and autonomous landing in unknown environments as well as efficient detection of water sampling areas. Regarding the autonomous landing operation, vision data acquired from the vehicle on-board sensors, are fed to appropriately designed Artificial Intelligence (AI) algorithms, which are able to detect suitable landing areas and surrounding obstacles. The visual feedback derived from the detection algorithms is incorporated into an efficient motion control scheme responsible to perform the safe landing operation. Regarding the water sampling area selection process, a robust vision-based AI module is designed and developed, which enables the aerial multi-rotor platform to maximize the visible water body from the on-board camera, assuring in this way that the water sampling task will be successful. | | | |

## Document History

| VERSION | ISSUE DATE | STAGE | DESCRIPTION | CONTRIBUTOR |
|---|---|---|---|---|
| 1.0 | 15.10.21 | Draft | A first complete draft of D5.2 is shared with partners for feedback, revision and information update. | NTUA |
| 2.0 | 22.10.21 | Draft | A full review of the completed draft of D5.2 | CERTH |
| 3.0 | 29.10.21 | Final | Final version of D5.2 | NTUA |
| 3.1 | 31.10.21 | Final | Final version for submission | UCY |

## Disclaimer

Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

## Copyright message

# Contents

# List of Figures

# 1 Introduction

In this work, we present the methods, tools and results derived in the context of *Deliverable 5.2: Drone take-off and landing strategies* of the PathoCERT project. This deliverable is a vital part of *Task 5.1: Drone-based situation awareness system with water sampling capabilities*. In particular, Task 5.1 aims at the development of fully autonomous solutions for aiding First Responders (FRs) in precarious critical situations through enhancing their abilities while negating many of the dangers. To this respect, the maximization of the degree of autonomy is considered critical, so as to not only minimize human involvement, and thus human-incurred errors, but also to better allocate resources and the available workforce to tasks that are yet impossible or impractical to automate.

In general, a mission that incorporates an aerial multi-rotor vehicle includes three phases; 1) Take-off, 2) Mission Execution, 3) Landing. While 1 & 3 are not the essential tasks with respect to the goal of the mission, they nevertheless are critical; if phase 1 were to fail, the whole mission would be compromised, while failure of phase 3 could mean either destruction of crucial data, samples and equipment, inability to carry out a subsequent mission, or even injury of the involved FRs. It is thus evident that special care needs to be taken with regard to these mission phases.

The goal of D5.2, is to increase the platform's autonomy by utilizing feedback from the on-board perception sensors and employ motion control algorithms in order to facilitate safe and autonomous landing in unknown environments as well as efficient detection of water sampling areas. Regarding the autonomous landing operation, vision data acquired from the vehicle on-board sensors, are fed to appropriately designed Artificial Intelligence (AI) algorithms, which are able to detect suitable landing areas and surrounding obstacles. The visual feedback derived from the detection algorithms is incorporated into an efficient motion control scheme responsible to perform the safe landing operation. Regarding the water sampling area selection process, a robust vision-based AI module is designed and developed. This module, named herein *visible water maximization algorithm*, enables the aerial multi-rotor platform to maximize the visible water body from the on-board camera, assuring in this way that the water sampling task will be successful.

In Section 2, an overview of the related literature will be presented. In Section 3, the technical problem statement will be delineated, while the relevant supporting infrastructure will be laid out in Section 4. In Section 5 we present the multirotor's kinematic and dynamic models, while the Sections 6-9 are dedicated to the solutions to the aforementioned problems. Finally, the results —both concerning high-fidelity simulations and outdoor experiments— are detailed in Sections 10 and 11. A conclusion surrounding the results of D5.2 is presented in Section 12.

# 2 Related Literature

The problem of landing an autonomous aerial platform has been previously tackled, with a plethora of different approaches as discussed in [18]. The approaches pertaining to outdoor landing, which we will focus upon, can mainly be classified into "known" [12, 17, 5, 9] and "unknown" environments [6]. The landing task can be broken down into two main phases; 1) finding an appropriate landing spot and 2) executing the landing maneuver. In order to execute these steps, both for known and unknown environments, the autonomous platform is equipped with sensing instruments relevant to the task at hand. Notably, during landing operations in known environments less complex sensing instruments and algorithms (e.g., plain RGB cameras) are usually utilized owing to the existence of pre-determined land-marks [18], which are often employed to not only provide with a suitable location for landing, but also to aid in controlling the drone (e.g., through visual control) so as to land safely [15]. In contrast, *a priori* unknown environments are evidently more challenging, as the on-board computing and sensing instruments need to be utilized in order to analyze the environment

and make on-the-fly decisions for choosing a landing location as well as controlling the vehicle to safely land in the appropriate spot.

In this work, we mainly focus on the unknown environments case, since they are relevant to the PathoDRONE missions that involve highly unknown and diverse environments. Additionally, the time to set up a proper landing area with landmarks might be crucial to the success of a mission in an emergency scenario. To deal with such variance of information and environments, advanced sensing capabilities need to be employed. In [7], a LiDAR sensor's information was fused with an RGB camera to obtain a depth map, which was post-processed to obtain a proper landing spot. In [20], an integrated solution with Simultaneous Localization and Mapping (SLAM) was employed to achieve both localization and control of the drone while calculating a safe landing area from a 3-dimensional representation of the platform's environment. However, the commercial drone employed, did not provide the open-source capabilities that have been adopted in the PathoDRONE platform.

Since many viable solutions are already available, we aim at providing an integrated, open-source based solution that is computationally efficient and can be employed with purely on-board sensing and computing capabilities (in contrast to [14] for example, where ground-station sensing instrumentation is incorporated). Finally, quick deployment and fast completion of the mission, were critical requirements taken into consideration during the design of the proposed solutions. Therefore, we formulate and employ custom schemes that differ compared to the relevant literature so as to accomplish the above goals.

# 3 Problem Statement

In this section, the goals of D5.2 are briefly summarized, before diving into the more technical details in the next sections. One of the tasks of D5.2 is to provide the PathoDRONE vehicles with the necessary technology to perform autonomously safe landing in unknown outdoors environments. The take-off procedure can be considered in most cases trivial, since the vehicle is already stationed in a safe obstacle-free area, hence it does not require further autonomous functions than the already included in a standard autopilot system. Furthermore, in D5.2 a part of the sampling process solution is also being tackled, since an algorithm for maximizing the water area beneath the hovering multi-rotor is developed and incorporated in the PathoDRONE system.

## 3.1 Visible Water Maximization Algorithm

In order to maximize the visible water area beneath the drone, so as to ensure the successful execution of the sampling procedure, the problem is formulated as follows: Consider a multi-rotor aerial vehicle, equipped with the traditional auto-pilot and stabilization capabilities of modern platforms, as well as a down-looking RGB camera. The drone initially receives an image from the camera that contains both parts that correspond to land and water. The problem is thus formulated as, i) classifying the pixels in the image received by the camera as "land" and "water" pixels and ii) formulating a control algorithm such that the drone views only "water"-type pixels in the received image upon convergence of the latter algorithm. In essence, these tasks involve a binary classification-image segmentation problem and a visual servoing problem.

## 3.2 Drone Landing in Unknown Environments

For the multi-rotor aerial platform to effectively land in unknown environments, a novel algorithm is formulated. The drone is required to land in an unforeseen, possibly geometrically complicated environment with obstacles and likely in the presence of FRs. The drone is given a waypoint up to

which it can safely and autonomously navigate, and upon reaching the latter, the drone is expected to analyse the data received from its sensors and find an appropriate landing spot, with the task being completed with the execution of the landing maneuver. In order to assure the feasibility of such algorithm, we make the assumption that the candidate landing area indicated by the respective waypoint, indeed includes an appropriate landing spot that is visible from the pre-determined height of the drone's flight path. This, while a strong assumption, is not irrational for the following reasons: i) The drone is expected to land in the vicinity of a group of FRs, as the landing will precede data or sample collection, ii) the drone is expected to be handled by a group of FRs after the completion of a mission. However, if the above assumption does not hold, the vehicle will be forced to perform locally an autonomous exploration task via the methods developed in *D5.4: PathoDRONE coverage algorithms using drones* in order to find a location which includes at least one appropriate landing spot.

We consider a drone equipped with the traditional auto-pilot and stabilization capabilities of modern platforms, along with a dedicated sensor capable to obtain three-dimensional information of the geometry of the ground beneath the vehicle's flight path. Given this information, the algorithm should be able to compose an adequately accurate representation of the environment and subsequently choose an appropriate landing spot and execute a control scheme that allows the vehicle to land safely at the selected location. The appropriateness of the landing spot is characterised by the flatness of a properly sized (given the drone's dimensions) obstacle-free area on the ground. Given the complexity of the ground topology, a simple representation is preferred so as to ensure robust and quick real-time execution of the algorithm. The two modules (selection of landing location and landing execution) will be treated separately.

# 4 PathoDRONE Supporting Infrastructure

## 4.1 Vehicle Description

The *Unmanned Aerial Vehicle* (UAV) is a crucial part of the overall PathoDRONE framework, thus, a vehicle with operational capabilities is necessary for the success of the PathoCERT missions. The *NTUA octorotor* (Figure 1) is a complicated robotic system, composed of multiple parts, which turn it into a powerful and fully autonomous UAV. More precisely, the NTUA octorotor is equipped with the Ardupilot firmware [1], responsible for controlling the aircraft through all regimes of flight. Ardupilot runs on the Cube Pixhawk 2.1 autopilot [3], the heart of the system where all the necessary hardware, e.g., ESCs and sensors, is integrated. The autopilot provides a set of modes which vary from semi-manual control to entirely autonomous, and, hence, the level of the authority given to the human pilot is adjusted correspondingly.

Additionally, the NTUA octorotor is equipped with navigation sensors which provide information about the vehicle position, velocity and angular orientation. Specifically, the following sensors are available:

- A rangefinder which is the primary altitude source,
- A compass or magnetometer providing heading/yaw measurements,
- A GPS which contributes to the estimation of the velocity and the position of the multirotor and
- An IMU which measures the linear accelerations and the body angular rates.

The above sensors are fused using an *Extended Kalman Filter* implemented by the ArduPilot side and, consequently, a proper state estimation is provided during the flight.


Figure 1: NTUA octorotor

The existence of a downward-looking RGB camera is of utmost importance for the detection of water surfaces and the execution of the Visible Water Maximization algorithm. However, a single monocular camera is not adequate for the detection of an appropriate landing area, since depth perception is required in order to extract information about the morphology of the ground surface. Consequently, the ZED 2 stereocamera is used in order to achieve both tasks. Additionally, the execution of computationally expensive algorithms such as image processing, classification using Convolutional Neural Networks or occupancy grid mapping is a necessary prerequisite and, hence, the incorporation of a powerful on board computer is inevitable. Among the various embedded computers, Jetson AGX Xavier [2] can be distinguished owing to its high performance. Beyond this, the Jetson Xavier is suitable for drone applications where size, weight and power consumption play a crucial role. The aforementioned system is appropriately setup in order to interface with the flight controller using the MAVLink protocol. The real-time control of the vehicle is achieved using the Robot Operating System (ROS) [19] and, particularly, through the MAVROS node which provides communication between ROS and ArduPilot vehicles.

## 4.2 Autopilot

In order to effectively control the dynamics that will be presented in Section 5 —and any aerial platform's behavior for that matter— low-level control architectures of varying complexity are employed. These are most commonly referred to as "Autopilots" and serve to compensate for the high-frequency dynamics that a high-level pilot —be it human or autonomous— cannot account for. In the context of the chosen platform, the open source ArduPilot system was chosen in order to provide reliable control of the vehicle. This framework comes with numerous features that are included in the software, and which provide a variety of flight modes from manual to fully autonomous ones, as well as a framework for the execution of fully autonomous missions.

In the case of the autonomous modes, the low-level control of the vehicle is realized by a cascaded PID control structure. More precisely, the desired position ${}^{I}\mathbf{p}_d$, velocity ${}^{I}\mathbf{v}_d$ and heading $\psi_d$ of the vehicle are received by the outer position loop, which is responsible for converting them to a target orientation and thrust. The inner attitude controller translates eventually the commanded thrust and torques to motor Pulse Width Modulation (PWM) values. A useful estimate of the actual state of the multirotor is obtained by fusing sensor measurements, such as data from GPS, compass and IMU, by employing a well-studied and widely adopted Extended Kalman Filter. A brief overview of the control architecture is depicted in Figure 2.

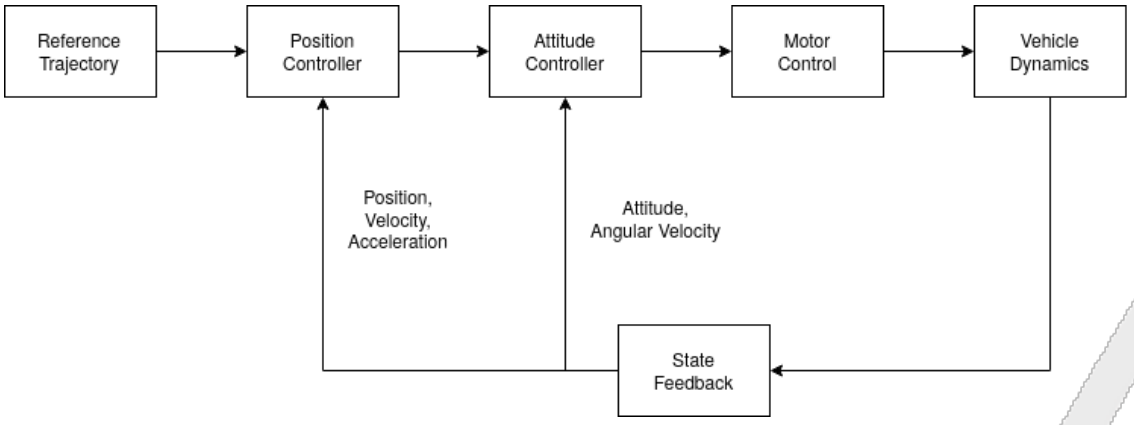



Figure 2: ArduPilot control architecture

## 4.3 Perception Sensor

As aforementioned, an essential prerequisite in order to guarantee the safe landing of the NTUA octorotor is the existence of a sensor, capable of reliably measuring the distance between the vehicle's frame of reference and the surrounding area. Hence, the NTUA octorotor is equipped with the ZED 2 stereocamera [4] (Figure 3), which efficiently estimates depth data. Similar to human binocular vision, the ZED 2 uses two cameras, displaced horizontally from one another, in order to obtain two different images from the same world scene. By comparing the corresponding pixels from these two images, the distance from ZED 2 to objects is estimated.


Figure 3: The ZED 2 stereocamera

The sensor data, collected by the ZED 2, is used in order to build a 2D occupancy grid map, a discrete representation of the robot workspace consisting of fixed-sized cells. An appropriate cost is assigned to each cell according to a criterion which quantifies the flatness of the surface and, consequently, a 2D costmap is generated, as shown in Figure 4. Eventually, safe landing areas are detected in the surrounding environment at each time instant. During the landing procedure, the NTUA octorotor should select the best landing spot in order to avoid collisions and efficiently complete the landing maneuver.

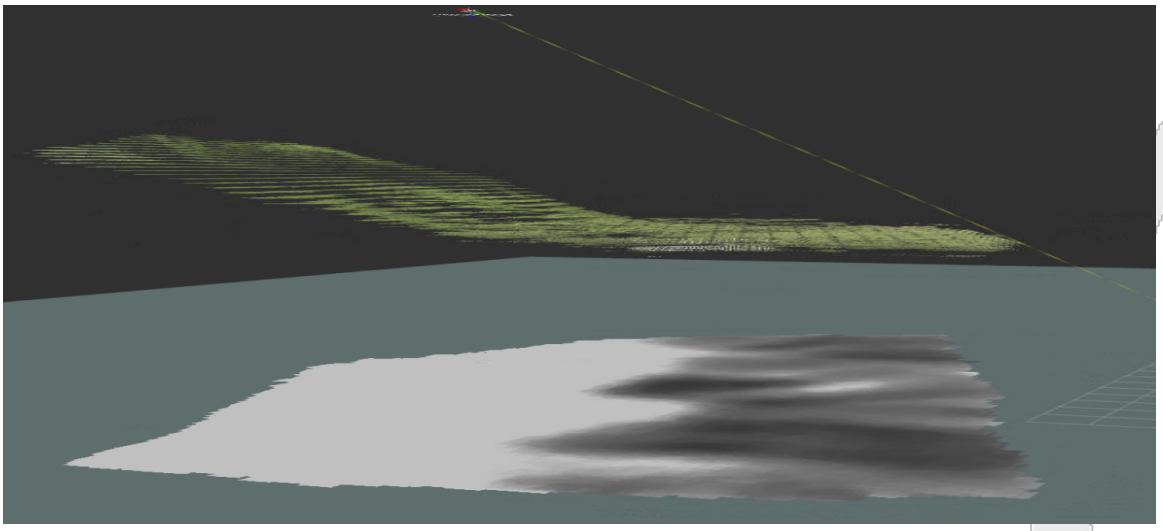


Figure 4: An example of a point-cloud and of the respective Occupancy Grid using the ROS vizualization tool. The gray cells correspond to appropriate landing areas, the white cells to inappropriate ones and the surrounding area is marked as unknown.

## 4.4 Simulator Description

A UAV simulation environment is set-up in order to evaluate custom control algorithms and ensure the smooth and efficient transition to real world experiments. The simulator is based on the well-known Gazebo [13], a powerful tool that provides the ability to simulate robots in complex environments using a comprehensive physics engine and graphics of high quality and real-world fidelity. A number of realistic 3D environments were created, similar to the ones encountered by first responders during their missions, by exploiting real world terrain heightmaps and adding appropriate water visual effects (Figures 5, 7 and 6). Consequently, the acquisition of synthetic data and the testing of image processing algorithms are feasible.

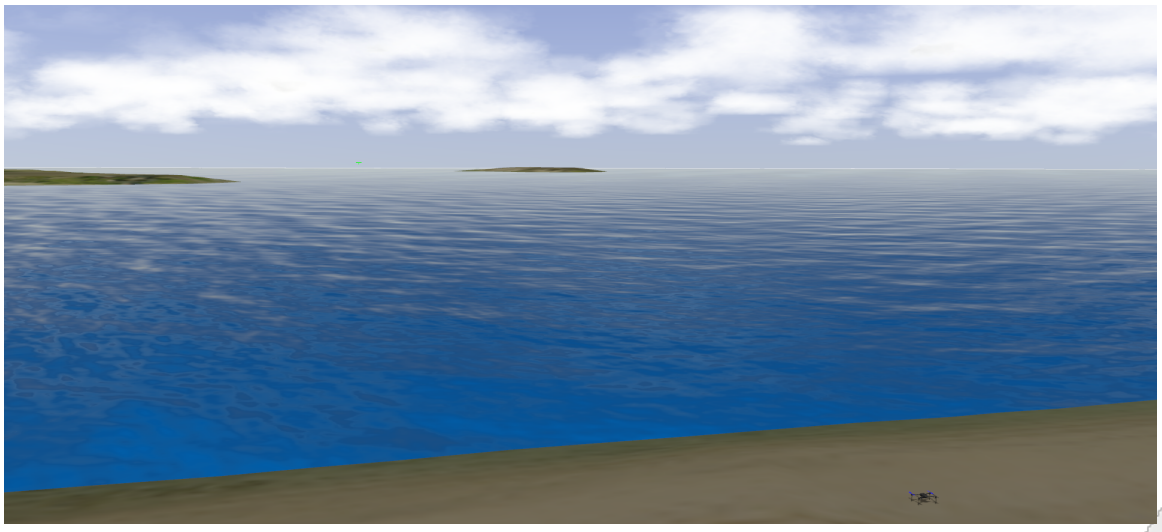


Figure 5: A Gazebo sea world

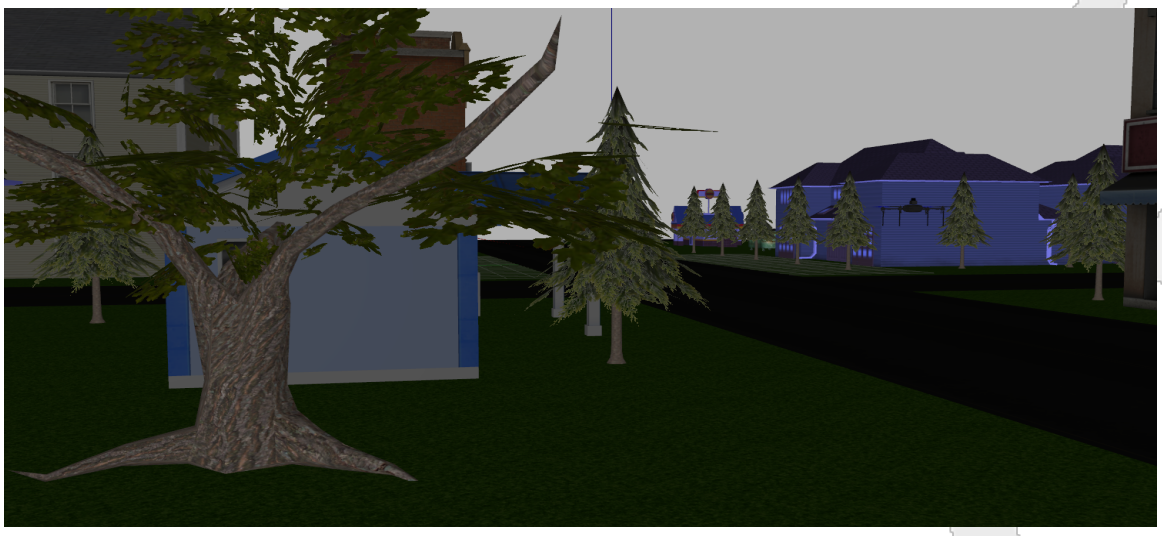


Figure 6: A Gazebo city world

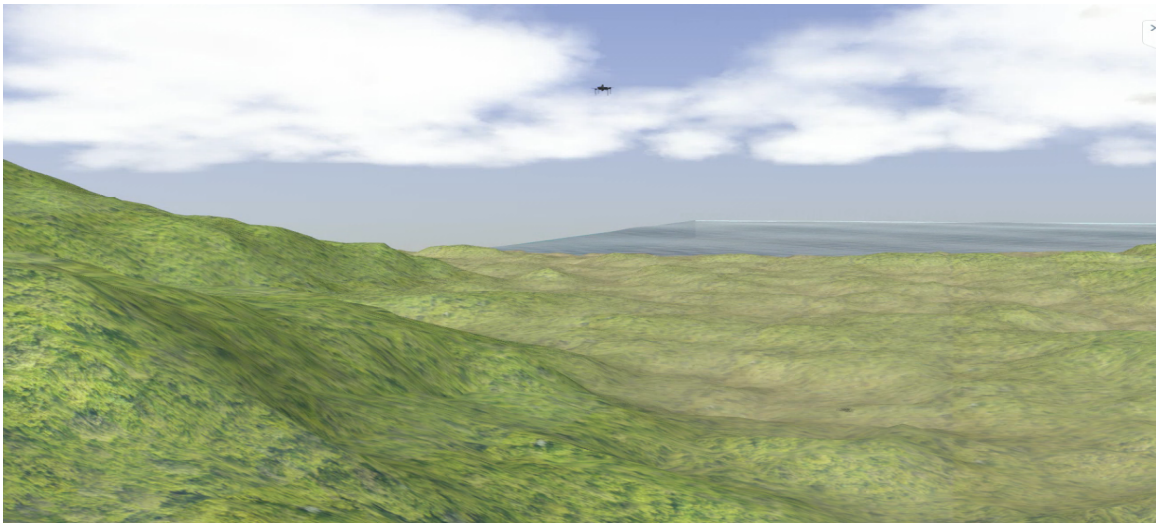


Figure 7: A Gazebo terrain model

A vehicle, integrated with the ArduPilot firmware, is used in all of the simulated scenarios in this report, thus allowing for Software in the Loop (SITL) simulations and testing the behavior of custom software without including actual physical hardware.

# 5 Multirotor Kinematics and Dynamics

In this section we will introduce the well-known kinematic and dynamic models of multirotor robotic platforms that are used for estimation and control in such systems. Consider a multirotor robot as depicted in Figure 8, that consists of a main body structure containing the electronics, the power supply and a sensing suite, along with multiple arms for mounting a number of motors and rotors and which also house the respective motor cables. The robot is further equipped with landing equipment that enables the vehicle to safely land without damaging the on-board sensitive instruments.

Let $\mathbf{B} = \{e_{B_x} \quad e_{B_y} \quad e_{B_z}\}$ denote the body fixed frame, which is attached to the vehicle's center of mass. In addition, an inertial frame $\mathbf{I} = \{e_{I_x} \quad e_{I_y} \quad e_{I_z}\}$, located at a fixed position, is defined, as shown in Figure 8. The Newton-Euler equations are used in order to describe the translational and rotational dynamics of a 6-DoF rigid body subject to external forces and torques [11], [16]:

$$ {}^{I}\dot{\mathbf{p}} = {}^{I}\mathbf{v} \tag{1} $$

$$ m\,{}^{I}\dot{\mathbf{v}} = {}^{I}\mathbf{R}_{B}\mathbf{F} \tag{2} $$

$$ \mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{M} \tag{3} $$

where ${}^{I}\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^{\top}$, ${}^{I}\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^{\top}$ are the position and the linear velocity of the multirotor expressed in $\mathbf{I}$, $m$ is the mass, ${}^{I}\mathbf{R}_{B}$ is the rotation matrix from $\mathbf{B}$ to $\mathbf{I}$, $\mathbf{J}$ is the inertia matrix and $\boldsymbol{\omega}$ is the angular velocity of the vehicle w.r.t the body frame $\mathbf{B}$. It should be noted that the rotation matrix is derived from the Euler roll, pitch, yaw angles or $\phi$,$\theta$,$\psi$ respectively.

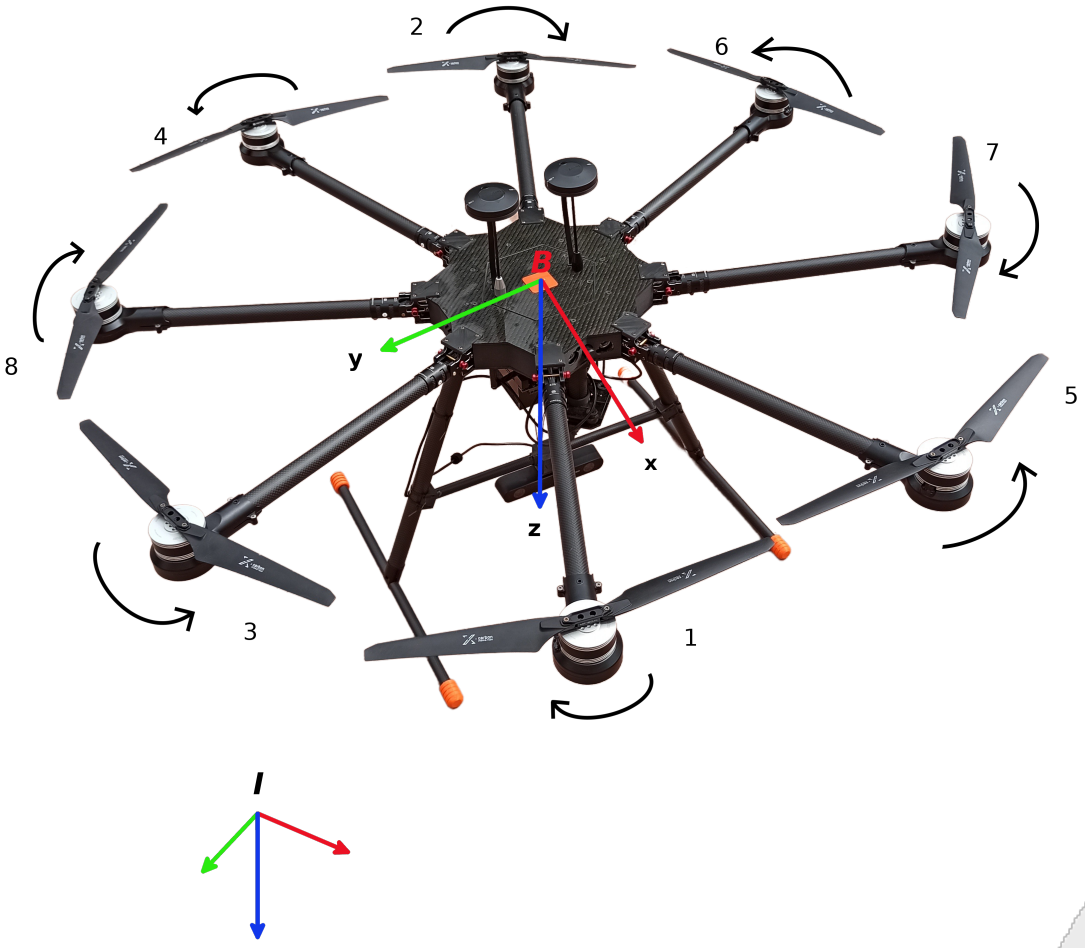



Figure 8: NTUA CSL octorotor's frame's

The external forces and torques applied on the airframe are:

$$\mathbf{F} = \mathbf{F}_M + \mathbf{F}_d + \mathbf{F}_g \tag{4}$$

$$\mathbf{M} = \mathbf{M}_M + \mathbf{M}_d \tag{5}$$

where:

- $\mathbf{F_d} = C_d {}^B\mathbf{R}_I \|{}^I\mathbf{v}\| {}^I\mathbf{v}$ are the drag forces with $C_d$ the drag coefficient matrix;
- $\mathbf{F_g} = m {}^B\mathbf{R}_I \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\top$ is the gravitational force with $g$ denoting the gravitational acceleration;
- $\mathbf{F_M} = \begin{bmatrix} 0 & 0 & -T \end{bmatrix}^\top$ is the total thrust produced by the motors;
- $\mathbf{M_M} = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^\top$ is the torque input vector;

- $\mathbf{M_d} = C_m \|\boldsymbol{\omega}\| \boldsymbol{\omega}$ are the drag moments;

The total thrust and moment applied to the vehicle depend on the number $N$ of motors and the configuration of the airframe. According to momentum theory, both the thrust force $T_i$ and the drag moment $\tau_i$ produced by the propellers is proportional to the square of the motor's angular velocity, i.e.,

$$T_i = C_T {\omega_i}^2 \tag{6}$$

$$\tau_i = C_\tau {\omega_i}^2, \tag{7}$$

where $i = 1, .., N$ and $C_T$, $C_\tau$ are the thrust and drag coefficients correspondingly.

In the specific case of the NTUA octorotor, the propulsion system consists of 8 motors. For an octorotor, the control allocation matrix and subsequently the relationship between the drag moments, total thrust and the angular velocities of the eight motors are defined, as follows:

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T & C_T & C_T & C_T & C_T \\ -C_T l_x & C_T l_x & -C_T l_x & -C_T l_x & C_T l_x & C_T l_x & C_T l_x & -C_T l_x \\ C_T l_y & -C_T l_y & C_T l_y & -C_T l_y & C_T l_y & -C_T l_y & C_T l_y & -C_T l_y \\ -C_\tau & -C_\tau & C_\tau & C_\tau & C_\tau & C_\tau & -C_\tau & -C_\tau \end{bmatrix} \begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \\ {\omega_5}^2 \\ {\omega_6}^2 \\ {\omega_7}^2 \\ {\omega_8}^2 \end{bmatrix} \tag{8}$$

# 6 CNN Water Detection

An important part of the visible water maximization algorithm is the real-time detection and classification of the ground and water surfaces that constitute the environment above which the UAV operates. The platform is equipped with a downward-looking stereoscopic camera. Besides the 3-dimensional information inferred from this sensor, the RGB images taken in real-time can be utilized to extract ground and water pixels.

In order to achieve this classification task, Convolutional Neural Networks were employed. Such Neural Networks have long been used for image analysis and robotic vision related tasks with great success. More specifically an image segmentation-oriented Neural Network was used to classify pixels as ground or water ones. The following procedure was implemented to prepare the dataset for training:

1. Manual labelling of the images,
2. Binary masks creation from labelled images,
3. Augmentation of the dataset through an open-source software [10],
4. Resizing of the frames from $720 \times 480$ pixels to $128 \times 128$ pixels,
5. Classification for 2 classes (Class 0: Ground and Class 1 : Water).

The structure of the proposed CNN is the following: Firstly, the images are passed into the convolutional layer, the normalized output of which is then passed on to the pooling layer. This layer collects data sets from the convolutional layer and samples the output of a result from the selected ones. After a plurality of subsequent convolutional and pooling layers, the final fully connected layers are utilized. The CNN weights are obtained through the back-propagation method. In order to apply the above procedure, the Keras image segmentation framework's vgg_unet CNN was employed [8].

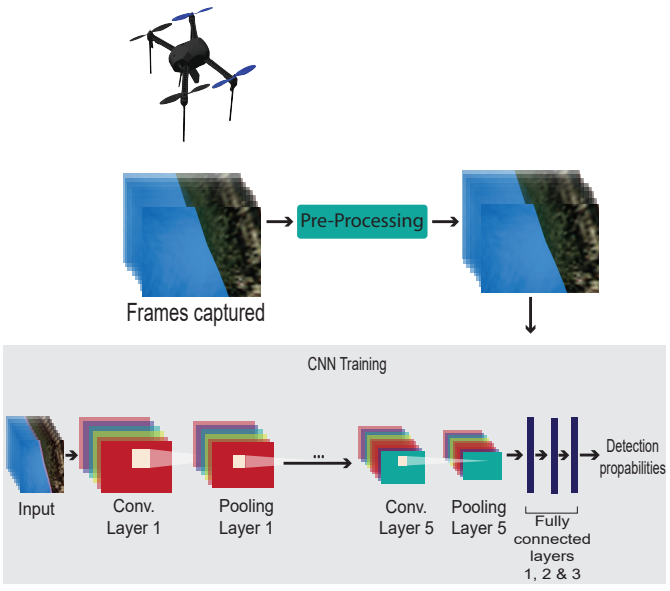



Figure 9: CNN architecture for water-land Detection

The CNN was trained over a sample of initially 1500 manually labeled real world images, augmented to a final dataset of 6000 images. The data were gathered in the form of video frames obtained by manually flying the octorotor above aquatic environments. Approximately 10% of the images were used as a validation set, while the rest were utilized for training. The algorithm converged to over 99% accuracy on the test set in 10 training epochs. An example of the output of the trained CNN, is depicted in Fig. 10.

# 7 Visible Water Maximization

In order to visually survey an area of a water body, a framework for maximizing the visible (to the drone's downwards-looking camera) water is formulated. Let $\mathcal{I} \subset \mathbb{R}^2$ denote the 2D image plane. Then this image consists of two parts, namely $\mathcal{I}_w \subseteq \mathcal{I}$ denoting the "water-part" of the image, and $\mathcal{I}_l \subseteq \mathcal{I}$ denoting the "land-part" of the image. Note that $\mathcal{I}_w \bigcup \mathcal{I}_l = \mathcal{I}$ and $\mathcal{I}_w \bigcap \mathcal{I}_l = \emptyset$. We then consider that the drone follows exactly a setpoint velocity control law, i.e., a single integrator model:

$$\dot{p} = {}^{I}v, \tag{9}$$

where $p = [p_x, p_y]^\top \in \mathbb{R}^2$ denotes the drone's longitudinal and lateral position in an inertial frame of reference (state vector), and ${}^{I}v \in \mathbb{R}^2$ denotes the velocity control (control-input vector). This assumption is very accurate for relatively small velocities. Along with the fact that normally the autopilots of relevant multi-rotor aerial vehicles are designed to take such commands as input, while ensuring stability and safety of the vehicle, such an assumption is in practice not only useful, but also not-limiting in its scope. Furthermore, it is evident that any lateral motion of the drone will be the result of a roll or pitch rotation. The low velocity assumption is also important in this regard, as the following analysis assumes an always downwards-looking camera, which is violated by large rotations in the aforementioned axis'. Alternatively, a gimbal can be used to ensure proper orientation of the camera frame. In order to maximize the water that is inside the frame of the camera, we employ the following control law:

$${}^{I}v = s_w(p) \triangleq \frac{1}{A}\left[S_x(p), S_y(p)\right]^\top, \tag{10}$$

where

$$S_y(p) = \iint_{\mathcal{I}_w(p)} (y - p_y)\, dA, \quad S_x(p) = \iint_{\mathcal{I}_w(p)} (x - p_x)\, dA, \tag{11}$$

where $A$ denotes the area in the image frame that is occupied by water. The above expression essentially means that we input the position vector of the centroid of the water area $\mathcal{I}_w$ w.r.t. the position of the drone. This means intuitively that the drone will tend to move towards the centroid of the water part of the image, which, under mild assumptions, will result in minimizing the area of land that is visible to the drone evidently thus maximizing the respective area of visible water. We will prove this assertion in Theorem 1.

Figure 10: Images from the on-board camera combined with the CNN output. The red pixels represent the pixels which the CNN classifies as water-pixels. The raw output of the NN consists of a binary mask in which the respective water pixels are highlighted.

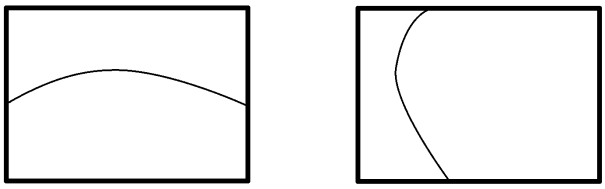


Figure 11: Example of a properly defined boundary-function (left) and an improperly defined one that can be fixed through a simple rotation (right).

For the purposes of the following proof, let the water-land boundary be denoted by a function, which is assumed to be a one-to-one mapping from the x to the y coordinates, i.e., $S_p(x) : \mathbb{R} \rightarrow \mathbb{R} \in C^1$ expressed w.r.t. an inertial frame of reference. The proof follows along the same lines even for functions that do not satisfy this assumption, by performing strategic cuts on the function on its critical points and breaking up the relevant integrals[2]. This process is left out for the sake of brevity, however, it would be imperative to execute such "cuts" in order to properly define the piece-wise inverse functions of the monotonic sections of the respective function for calculating the double integrals of Eq. (11). We thus limit ourselves to monotonic ones. One final assumption is that some water is visible in the camera frame upon the execution of the control law (10). The following theorem (1) proves that the control scheme 10 stabilizes the drone in a position where the visible water to the on-board camera is maximized:

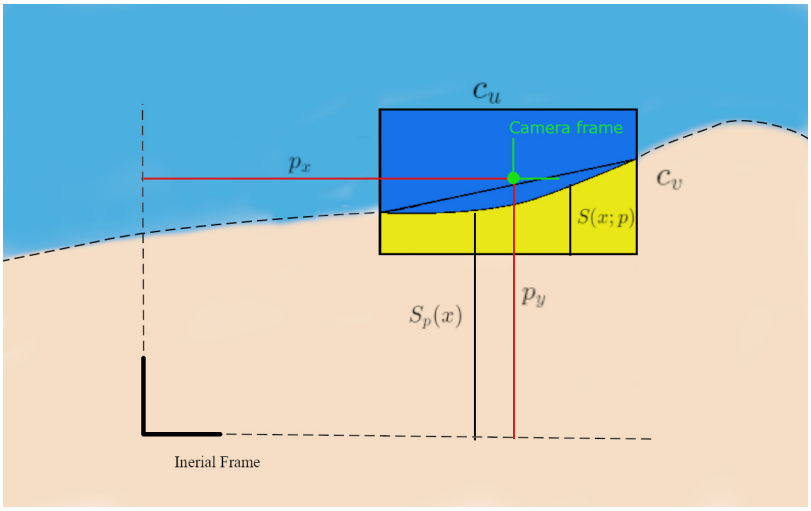



Figure 12: The Figure of Theorem 1. The relevant frames are depicted, along with the function that represents the water-land boundary. The components of the drone's position vector are also depicted.

[1]It is evident how the function of the land-water boundary depends on the position of the drone.

[2]This analytical definition can be applied even if the boundary is not a function (i.e., not right-unique), but can be expressed as such via a rotation of the image plane in the xy plane —see Fig. 11—.

**Theorem 1.** *Consider a coordinate system centered at the camera frame, which is assumed to be identical to the drone's position, namely* $p \in \mathbb{R}^2$. *Then let* $S(x) \triangleq S(x;p) : [-c_u, c_u] \to [-c_v, c_v] \in C^1$ *denote a function that describes the land-water boundary w.r.t. the above frame, where* $c_u, c_v$ *are the halved dimensions of the image plane in* $[x, y]$ *respectively -appropriately scaled w.r.t. the height of the drone to reflect physical, real-world dimensions-. We further assume that the characteristics of the shore are such that the body of water is larger than the respective land mass, such that no land encircles a part of water*[3]. *Then the dynamical system 9 under the control law 10 is asymptotically stable.*

*Proof.* Consider as a Lyapunov candidate the following function:

$$\mathcal{L}(p) = A(p) = \iint_{\mathcal{I}_l(p)} dA. \tag{12}$$

This function essentially expresses the measure of area of the land-part of the ground that the drone observes and is always positive. It is furthermore zero only when the drone observes nothing but water. Thus, it is a valid Lyapunov candidate for the goal in mind, i.e., that the drone only observes water upon convergence. We will now show how the time derivative of the above Lyapunov candidate is negative, except for point(s) of convergence. We have:

$$\begin{aligned} \dot{\mathcal{L}} &= \nabla_p A^\top \dot{p} \\ &= \nabla_p A^\top s_w(p). \end{aligned} \tag{13}$$

We will prove that

$$\dot{\mathcal{L}} < 0, \tag{14}$$

for points that do not result in the drone observing solely water, by showing that the vectors $\nabla_p A$ and $s_w(p)$ are pointing towards opposite directions. Firstly, note that the centroid of the land area, denoted by $s_l(p)$ is contradirectional to $s_w(p)$, owing to the centroid of the whole image being at the origin of the chosen frame of reference, and the fact that the two areas combined form the image. To see this, note that given the centroids of two shapes $s_1, s_2 \in \mathbb{R}^2$ in a given coordinate frame, then the centroid $s_{1,2}$ of the shape resulting from combining the two former ones is given by:

$$s_{1,2} = \frac{A_1 s_1 + A_2 s_2}{A_{1,2}}, \tag{15}$$

where $A_1, A_2, A_{1,2}$ denote the areas of the respective shapes. Since in our case, the coordinate frame is centered at the center of the camera frame, which coincides with the centroid of the rectangular image, then:

$$\begin{aligned} s_{l,w} = \frac{A_w s_w(p) + A_l s_l(p)}{A_{l,w}} = \vec{0} \Rightarrow \\ s_w(p) = -\frac{A_l}{A_w} s_l(p), \end{aligned} \tag{16}$$

which shows that the two centroids are contradirectional (see Fig. 13).

[3] This assumption is rather mild, the applications at which this framework is targeted would most likely involve large bodies of water, with the drone operating at such heights that even a small lake would satisfy the above assumption. If however, this assumption does not hold, then the drone would still converge with the water body at its center, thus the monitoring process could still be considered successful.

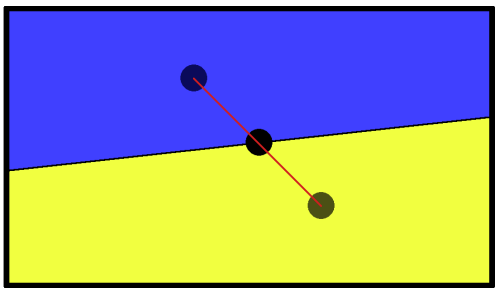


Figure 13: The Centroids of the water-land areas.

We now only have to prove that $\nabla_p A$ and $s_l(p)$ are codirectional. Assume that the function $S(x)$ is monotonically increasing —see Fig. 12—. The proof for a monotonically decreasing function follows the same procedure. We have:

$$A(p) = \iint_{\mathcal{I}_l} dA = \int_{p_x - c_u}^{p_x + c_u} S_p(x) - (p_y - c_v) dx, \tag{17}$$

where $S_p(x)$ is the function $S(x)$ expressed w.r.t. the inertial frame of reference. Thus:

$$\nabla_p A = [S_p(p_x + c_u) - S_p(p_x - c_u), -2c_u]^\top = [S(c_u) - S(-c_u), -2c_u]^\top. \tag{18}$$

Furthermore, expressed at the new frame of reference (camera frame center), the coordinates of $s_l(p) = [s_{l,x}, s_{l,y}]^\top$ are:

$$s_{l,y}(p) = \tfrac{1}{2} \int_{-c_u}^{c_u} S^2(x) - c_v^2 dx, \tag{19}$$

which by applying the mean value theorem becomes:

$$s_{l,y}(p) = \tfrac{1}{2} \left[ \left( S^2(\chi) - c_v^2 \right) \right] 2c_u, \tag{20}$$

where $\chi \in [-c_u, c_u]$, and

$$s_{l,x}(p) = \tfrac{1}{2} \int_{S(-c_u)}^{S(c_u)} c_u^2 - \left( S^{-1}(y) \right)^2 dy, \tag{21}$$

which by applying the mean value theorem becomes:

$$s_{l,x}(p) = \tfrac{1}{2} \left[ c_u^2 - \left( S^{-1}(\xi) \right)^2 \right] \left[ S(c_u) - S(-c_u) \right], \tag{22}$$

where $\xi \in [S(-c_u), S(c_u)]$. Putting all of the above together, we get:

$$\nabla_p A^\top s_l(p) = \tfrac{1}{2} \left[ S(c_u) - S(-c_u) \right]^2 \left[ c_u^2 - \left( S^{-1}(\xi) \right)^2 \right] + \tfrac{1}{2} \left[ c_v^2 - S^2(\chi) \right] 4c_u^2, \tag{23}$$

which, since $S^{-1} : [-c_v, c_v] \to [-c_u, c_u]$ and $S : [-c_u, c_u] \to [-c_v, c_v]$ is positive as a sum of positive terms. Furthermore, this quantity is zero only when the volume defined by the bounds $[-c_u, c_u]$, $[-c_v, c_v]$ and $S(x)$ is zero. This means that the vectors $\nabla_p A$ and $\dot{p}$ are contradirectional, since $\nabla_p A$ and $s_l(p)$ are codirectional. Thus, $\dot{\mathcal{L}} < 0$ and the system 9 under the control law 10 is asymptotically stable under the proposed assumptions. This concludes the proof. □

An example of a Lyapunov function used in the above proof is depicted in Fig. 14.

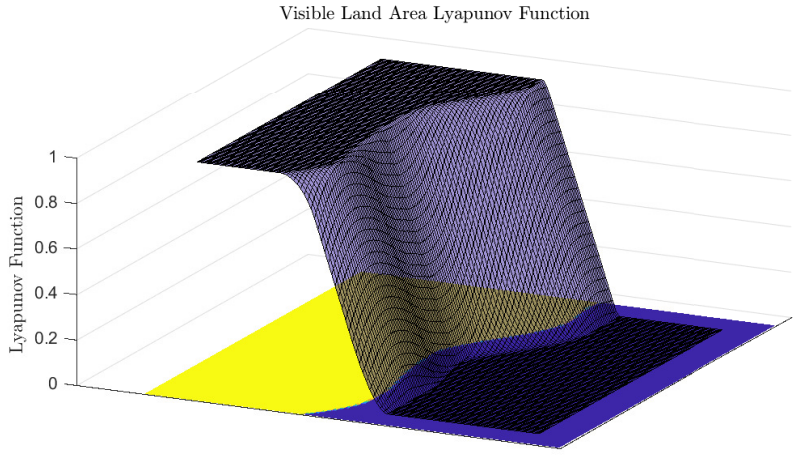
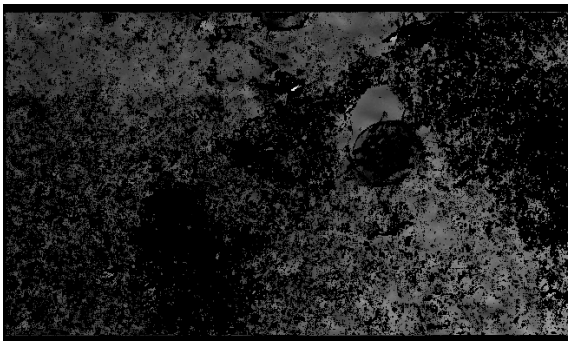


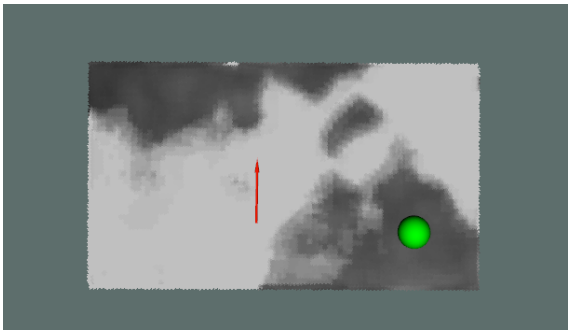

Figure 14: Example of a Lyapunov function of Theorem 1.

Evidently, the drone will converge to any part of the state space where the Lyapunov function is equal to zero, which might lead to it drifting indefinitely away from the shore due to external disturbances. This will be avoided through the high-level controller which will stop the execution. One can visualize the drone's motion under the proposed control law as rolling down the hill of the Lyapunov function. It becomes thus evident that the visible water will be maximized. Note that this analogy is not exact as the drone will not follow exactly the negated gradient of the Lyapunov function, however, we have shown that it will always move towards the same direction as the aforementioned gradient.



(a) Depth Image

(b) Score Image



(c) Occupancy Grid

Figure 15: Landing Area Detection Algorithm

# 8 Landing Area Detection

Having presented the parts of the framework that address the execution of the main mission, we now address the landing problem. For a UAV to land autonomously, while a 2D-point might be given by a pilot, the vehicle should examine the topology of the area surrounding the aforementioned point, and decide where to land based on the given topology.

In order to accomplish the above, first an area of interest around the provided waypoint is selected. Then, the latter is discretized into a grid of cells corresponding to points on the ground. In order to determine how fit a specific area is for landing, an appropriate score is formulated, where the cells with higher scores are considered more fit for landing. In this way, an appropriately landing area (according to the drone's footprint) is defined on the fly.

To accomplish this, height measurements need to be acquired in real-time. This is achieved by exploiting the Depth Image (Figure 15a) provided by the on-board stereo-camera, but can be obtained through any other method. Once a Depth Image $D_n(x, y)$ -where the tuple $(x, y)$ denotes the drone's position in an inertial frame of reference and the index $n$ an instant of measurements - is obtained, a procedure of post-processing is followed.

More precisely, the Depth Image is a 2D matrix with dimensions $W \times H$, where $W, H$ are the width and the height of the image. At each pixel $(u, v)$, where $u = 0, \cdots, W-1$ and $v = 0, \cdots, H-1$, a distance value $z(u, v)$, expressed in meters, is stored. It should be noted that some elements of the matrix may be characterized as $+\infty$ or $-\infty$ in case that the objects of the corresponding pixels are respectively too far or too close to the camera. Additionally, due to visual occlusions, the estimation of depth may be infeasible or highly inaccurate and, hence, the values of pixels with low confidence

are marked as *Not a Number* (NaN).

In order to evaluate the appropriateness of a pixel for landing, the neighbourhood around the query pixel $(u, v)$ is examined. Specifically, a window of size $(K+1) \times (K+1)$ is utilized so as to determine the region of interest (ROI) around each pixel $(u, v)$. It is mentioned that the size of the window is highly related to the size of the UAV and is therefore chosen such that the cost of each pixel reflects the ability of the drone to land in the respective surrounding area. The suitability of the surface is quantified by computing the standard deviation $\sigma(u, v)$ of the z-coordinates of the $(K+1) \cdot (K+1)$ pixels which constitute the aforementioned region of interest:

$$\overline{z}(u,v) = \frac{1}{(K+1)\cdot(K+1)} \sum_{i=u-\frac{K}{2}}^{u+\frac{K}{2}} \sum_{j=v-\frac{K}{2}}^{v+\frac{K}{2}} z(i,j) \tag{24}$$

$$\sigma(u,v) = \sqrt{\frac{1}{(K+1)\cdot(K+1)} \sum_{i=u-\frac{K}{2}}^{u+\frac{K}{2}} \sum_{j=v-\frac{K}{2}}^{v+\frac{K}{2}} (z(i,j) - \overline{z}(u,v))^2} \tag{25}$$

As for the pixels which lie on the borders of the image, a window of smaller size is considered. Additionally, the percentage $\pi(u, v)$ of finite distance values, i.e., values which are not marked as $+\infty$, $-\infty$ or NaNs, is computed inside the region of interest in order to check the validity of the depth information. Regions of interest with percentage $\pi(u, v)$ less than a threshold value $\pi_{min}$ are discarded.

The related score of the region of interest is eventually normalized to $[0, 1]$, where score 0 indicates inappropriate areas for landing while 1 appropriate ones, according to the following equation:

$$C(u,v) = \begin{cases} e^{-\sigma(u,v)} & \pi(u,v) > \pi_{min} \\ \\ 0 & \pi(u,v) \leq \pi_{min} \end{cases} \tag{26}$$

Consequently, a new image, namely Score Image (Figure 15b), $C_n(x, y)$ is constructed, where the score values are stored at each time instant $n$, and a bilateral filter is then applied so as to smooth the Score Image while preserving edges. In order to globally store the associated information, the above scores are matched to the corresponding cells of the Occupancy Grid (Figure 15c) which is expressed with respect to the Inertial frame. The matching is performed by exploiting the intrinsic camera parameters, particularly the principal point $c_u$,$c_v$ and the focal lengths $f_x$, $f_y$ in the $u$ and $v$ directions respectively, and the current position ${}^{I}\mathbf{p}_n$ and orientation $\phi_n, \theta_n, \psi_n$ of the UAV. The aforementioned scores are ultimately averaged over the whole set of measurements for each cell:

$$C_n^{avg}(c_x, c_y) = C_{n-1}^{avg}(c_x, c_y) \frac{N_n(c_x, c_y) - 1}{N_n(c_x, c_y)} + C_n(c_x, c_y) \frac{1}{N_n(c_x, c_y)} \tag{27}$$

where the tuple $c_x, c_y$ denotes the respective cell of the pixel $u, v$, $N_n(c_x, c_y)$ the number of costs computed through the respective observations up to the $n$-th measurement instant for the above cell and $C_n(c_x, c_y)$ denote the score for the respective cell obtained at the $n$-th measurement instant.

Finally, the drone selects the best area to land by finding the cell of the grid that has the maximum score, and matching it to a physical 2D position:

$$p_{landing} = p(c_x^l, c_y^l), \tag{28}$$

where:

$$(c_x^l, c_y^l) = \arg\{\max_{c_x, c_y}\{C(c_x, c_y)\}\} \tag{29}$$

# 9 Autonomous Landing

As far as the autonomous landing is concerned, a model predictive controller (MPC) is formulated in order to complete safely the landing procedure. More precisely, the objective of the control scheme is to minimize the error in position between the vehicle and the aforementioned detected landing location, while, simultaneously, satisfying the constraints imposed by the vehicle's low-level velocity controller. The translational kinematics of the multirotor are described by the equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \Rightarrow {}^{I}\dot{\mathbf{p}} = \mathbf{R}_z(\psi)\,{}^{B}\mathbf{v} \tag{30}$$

where ${}^{I}\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^{\top} \in \mathbb{R}^3$ is the position of the vehicle, ${}^{B}\mathbf{v} = \begin{bmatrix} u & v & w \end{bmatrix}^{\top} \in \mathbb{R}^3$ is the velocity control input with respect to the body frame of the vehicle, $\psi$ is the yaw angle and $\mathbf{R}_z(\psi)$ is the rotation matrix around the z-axis of the inertial frame.

At each time instant $t$, a constrained optimization problem is solved by the MPC over a finite horizon of $N$ steps and, consequently, an optimal sequence of feasible control inputs $\left({}^{B}\mathbf{v}_t^*, \cdots, {}^{B}\mathbf{v}_{t+N}^*\right)$ is derived, which minimizes the following weighted sum of accumulative and terminal costs:

$$\min_{{}^{B}\mathbf{v}_t, \cdots, {}^{B}\mathbf{v}_{t+N}} \sum_{k=0}^{N-1} (\|{}^{I}\mathbf{p}_{t+k} - {}^{I}\mathbf{p}_{des}\|_Q^2 + \|{}^{B}\mathbf{v}_{t+k}\|_R^2) + \|{}^{I}\mathbf{p}_{t+N} - {}^{I}\mathbf{p}_{des}\|_P^2$$

$$\text{subject to : } {}^{I}\mathbf{p}_{t+k+1} = {}^{I}\mathbf{p}_{t+k} + \mathbf{R}_z(\psi_{t+k})\,{}^{B}\mathbf{v}_{t+k} \cdot dt, k = 0, \cdots, N-1$$

$${}^{I}\mathbf{p}_t = {}^{I}\mathbf{p}(t)$$

$${}^{B}\mathbf{v}_t \in U = \left\{ \forall {}^{B}\mathbf{v} \in \mathbb{R}^3 : \begin{bmatrix} u_{min} \\ v_{min} \\ w_{min} \end{bmatrix} \leq {}^{B}\mathbf{v} \leq \begin{bmatrix} u_{max} \\ v_{max} \\ w_{max} \end{bmatrix} \right\}$$

where $dt$ is the sample time, ${}^{I}\mathbf{p}_{des}$ is the detected landing position and $Q$, $R$ and $P$ are positive definite matrices which penalize the state error, the input and the terminal state error respectively. According to receding horizon control principle, only the first element ${}^{B}\mathbf{v}_t^*$ of the optimal control sequence is applied to the vehicle and the optimization procedure is repeated at the next time instant $t+1$, given the measured position ${}^{I}\mathbf{p}_{t+1}$, until the successful landing of the vehicle.

# 10 Simulation Results

In order to evaluate the proposed algorithms on a first level, integrated in a mission framework, all the herein presented schemes were employed in the simulation environment (see Section 4.4). This enables assessing the framework in its entirety in order to ensure that each module works appropriately both independently and in conjunction with other modules. The mission consists of waypoint-to-waypoint navigation (which is the subject of D5.1), subsequent visible water maximization and the completion of the mission through the execution of the landing procedure. An overview of the simulator framework is depicted in Fig. 16. The evolution of the x,y and z components of the vehicle are depicted in Fig. 17. Concerning the landing task, an overall view is depicted in Fig. 18 and the cost is presented in Fig. 20, with the final 3D landing trajectory depicted in Fig. 19. The performance of the overall framework is better illustrated in the relevant video (`https://youtu.be/5H2HhRz6Oqg`).

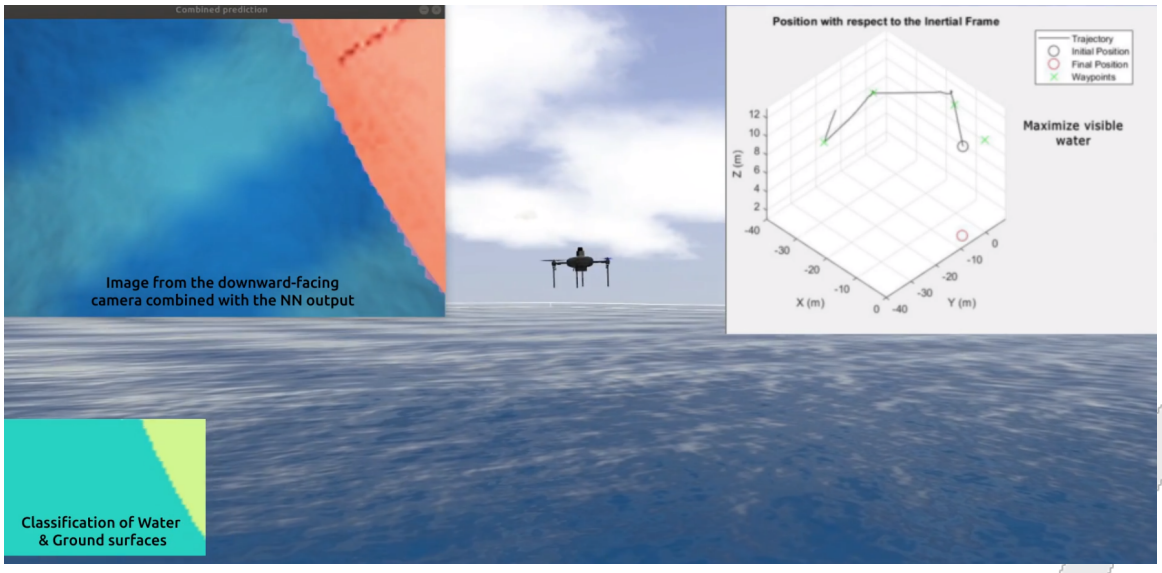
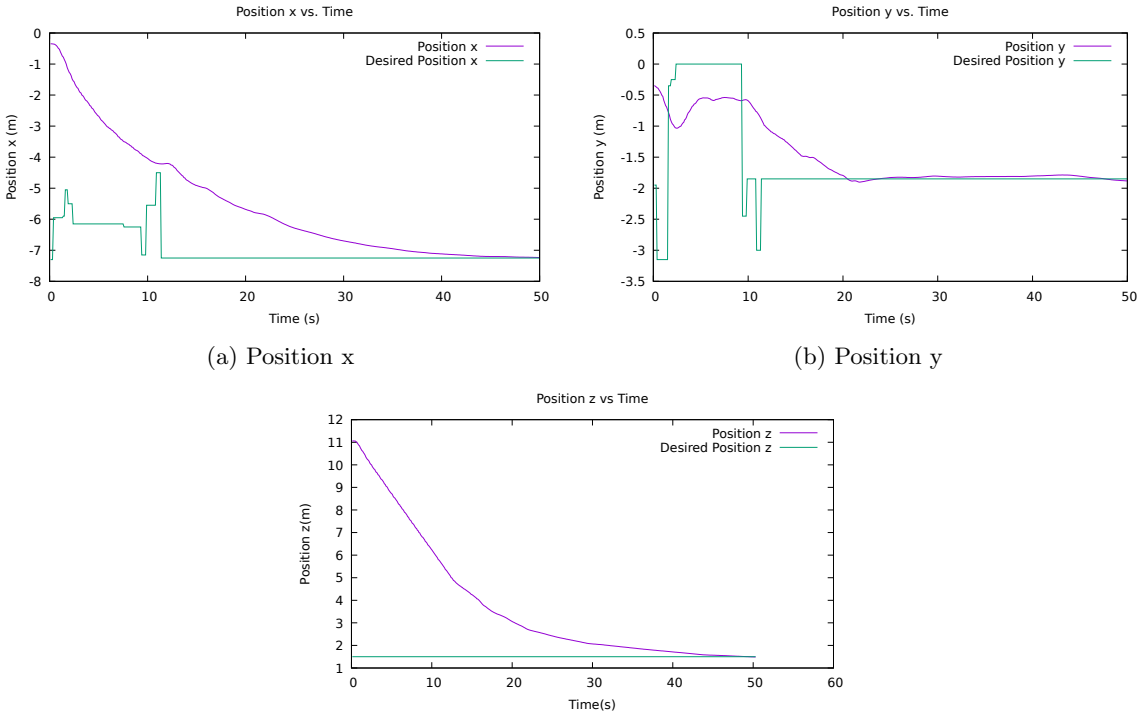



Figure 16: Overview of the Simulator Environment, along with the NN output (pure and overlayed on the camera view) and a 3D plot of the waypoints and multi-rotor's trajectory.




(a) Position x

(b) Position y

(c) Position z

Figure 17: The position ${}^{I}\mathbf{p}$ of the vehicle with respect to the map frame $\mathbf{I}$ (purple curve) along with the desired (reference for the MPC) position as output of the respective algorithm (green curve). The desired position is continually updated as the algorithm gathers more data and the cost map is accordingly updated.


Figure 18: Overview of the Simulator Environment during the Landing process, along with the NN output (pure output and overlayed on the camera view) and the point-cloud from the stereo-camera overlayed onto the resulting cost map. The fitness of each position (cell) for landing is depicted through the grayscale value (black: unfit, white: fit for landing).

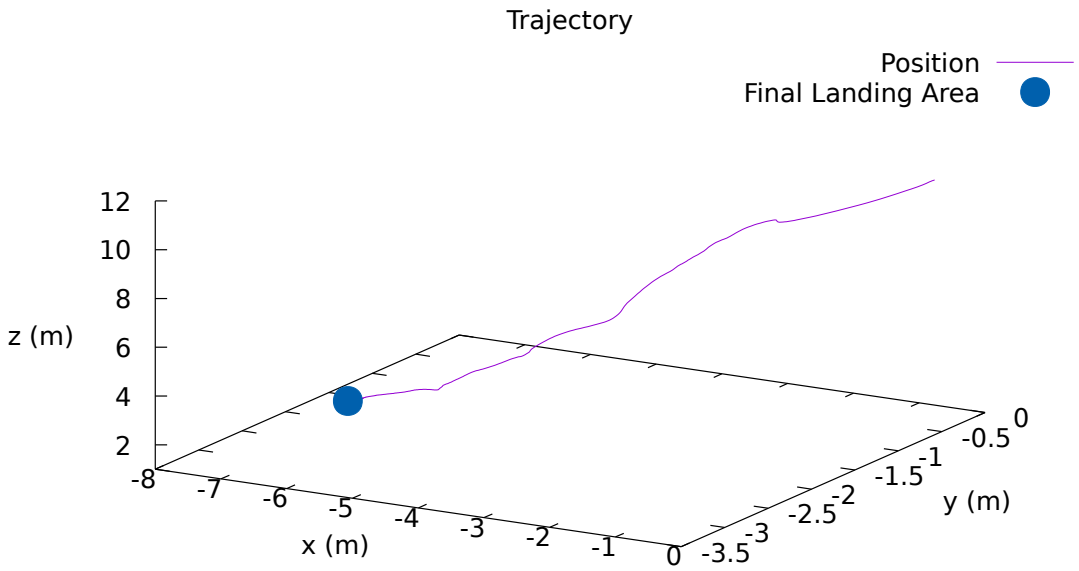



Figure 19: Executed 3-dimensional trajectory under the MPC control law during the landing process.

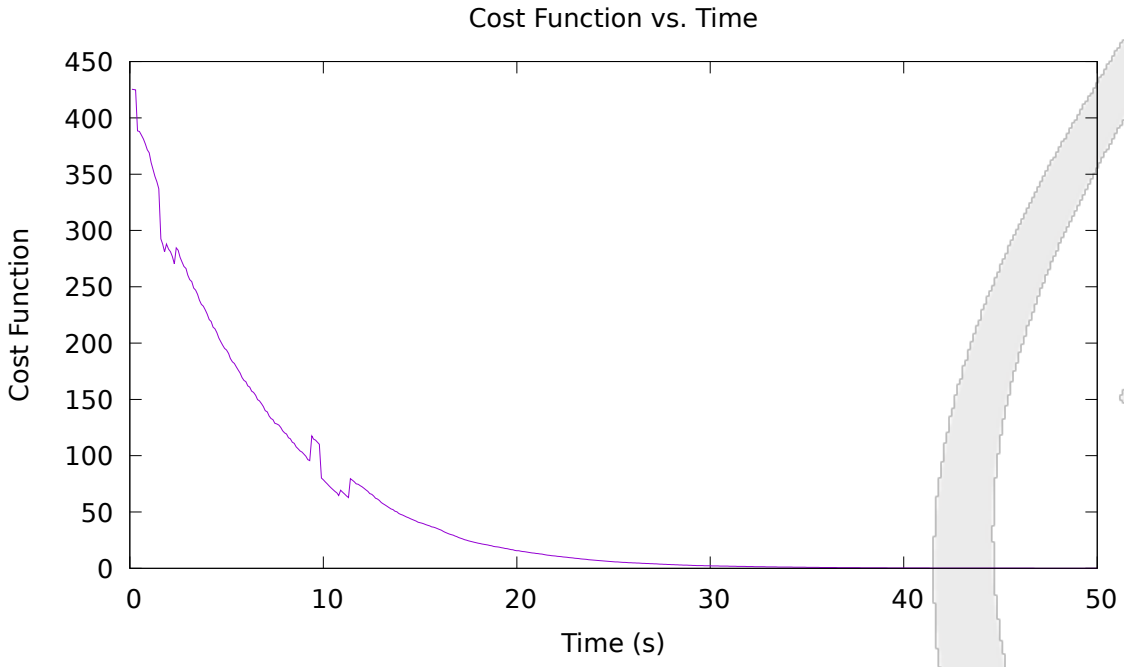



Figure 20: The time evolution of the MPC cost function during the landing process.

# 11 Experimental Results

In order to evaluate the efficacy, robustness and applicability of the proposed control schemes, the aforementioned algorithms are employed in the robotic platform discussed in Section 4, for carrying out real-world experiments in environments that are similar to in-field conditions. In this manner, the algorithms are evaluated in highly varying conditions of the robot's environment which not only influence the tasks directly (i.e., rough landing conditions) but also indirectly through variance in the sensing process (i.e., different lighting conditions). The following sub-sections include landing experiments 11.1 and visible water maximization experiments 11.2.

## 11.1 Landing Experiments

**Experiment I**

The first experiment was carried out in an outdoor area where the multi-rotor platform was positioned above an area for which the on-board camera feed consisted of practically two types of terrain, namely an inappropriate part that included dense foliage and an appropriate part that consisted of asphalt. A frame from the respective position is depicted in Fig. 21. The respective depth map is depicted in Fig. 22, while the resulting cost map is depicted in Fig. 23 along with the appropriate landing area as determined by the algorithm, depicted in a green ball. It is evident that the algorithm with no *a priori* knowledge of the surrounding area properly identifies an appropriate landing spot.

Following proper recognition of a safe landing area, the landing procedure is initiated. The response of the system under the formulated MPC scheme is depicted in the diagrams of Fig. 24, while the evolution of the MPC cost w.r.t. time is depicted in Fig. 25. It is evident that the controller exhibits satisfactory performance w.r.t. to task execution and trajectory response. An overview of the experiment is presented in Fig. 26 and in the corresponding video `https://youtu.be/LjZvSn45vx4`.

Figure 21: Image of the outdoor environment of Landing Experiment I captured by the on-board camera.

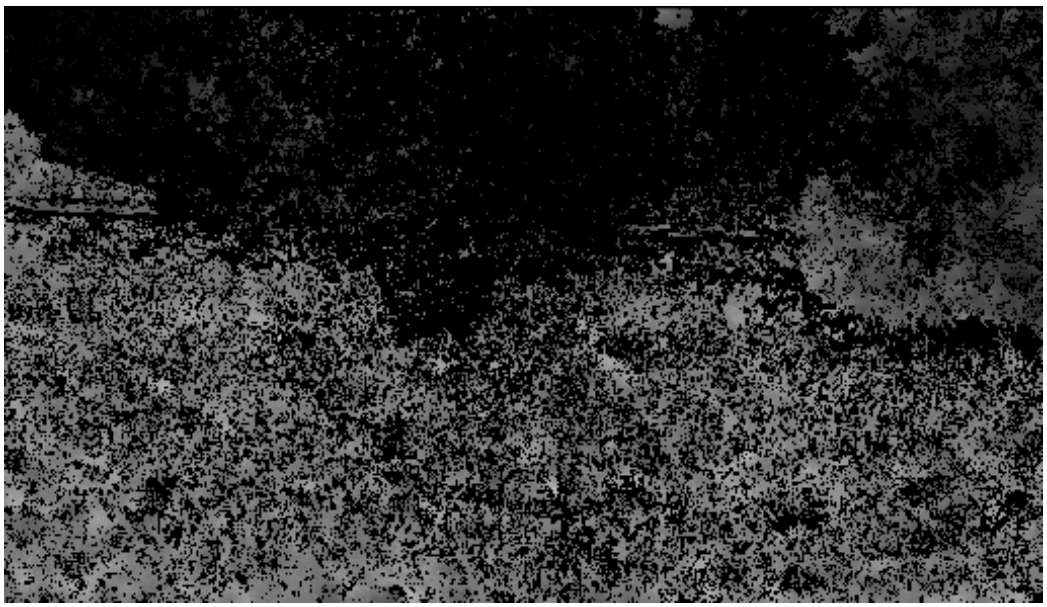


Figure 22: Depth Image of the outdoor environment obtained by the downward-looking ZED2 during the Landing Experiment I.

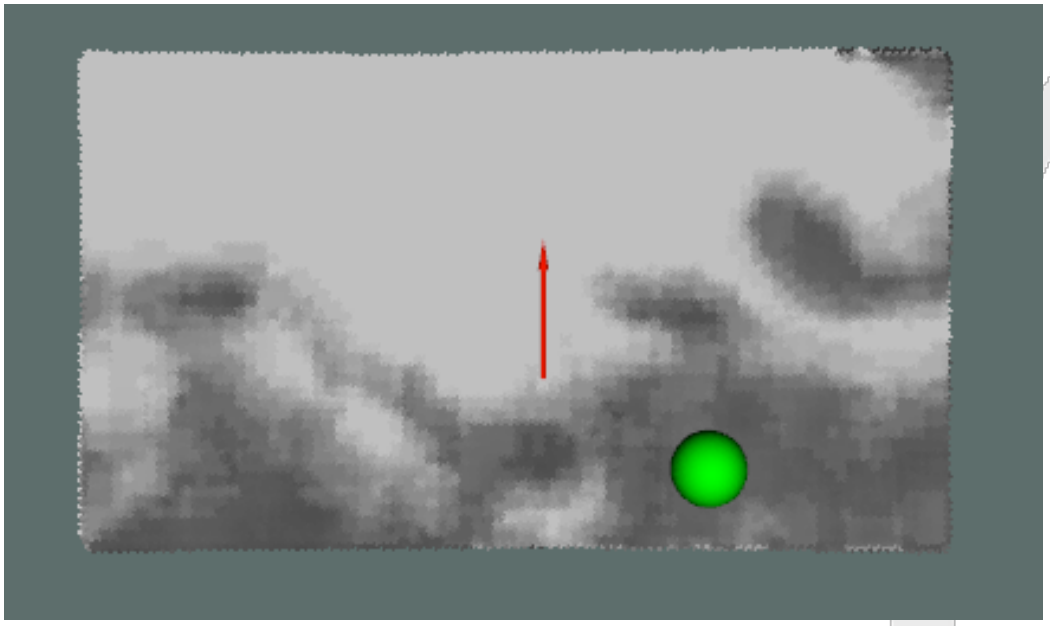
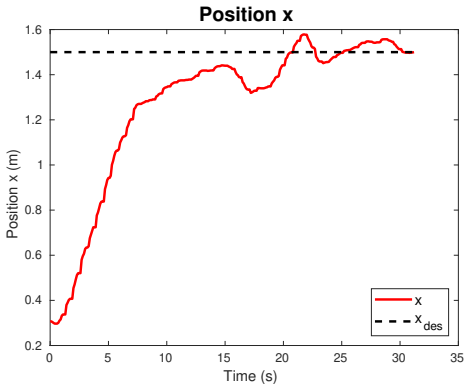
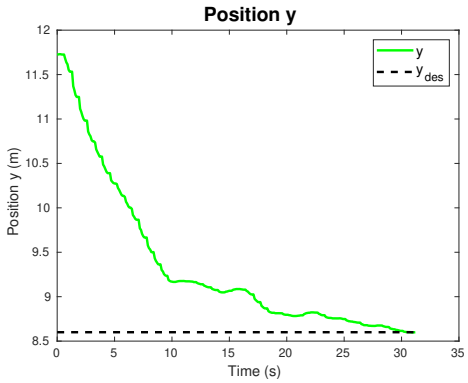


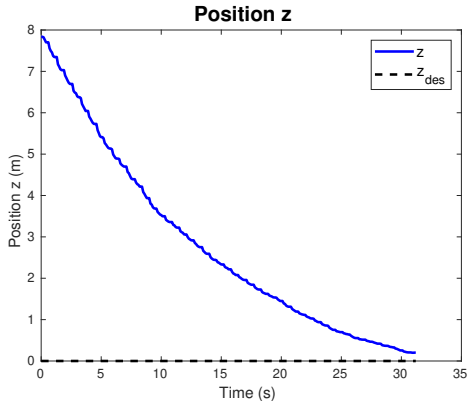
Figure 23: Occupancy Grid, built in real time according to Section 8, and the best landing spot (marked as a green sphere) during the Landing Experiment I.




(a) Position x




(b) Position y




(c) Position z

Figure 24: The position ${}^{I}\mathbf{p}$ of the vehicle with respect to the map frame $\mathbf{I}$, compared to the desired landing spot ${}^{I}\mathbf{p}_{des}$ during the Landing Experiment I.

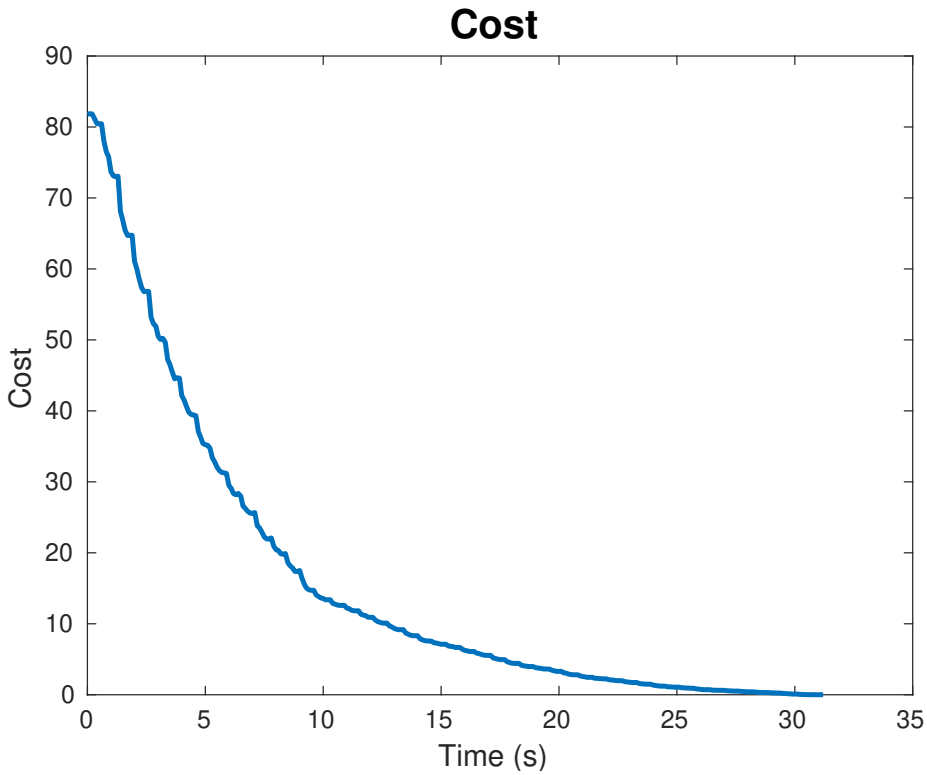



Figure 25: Time evolution of MPC cost function during the Landing Experiment I.

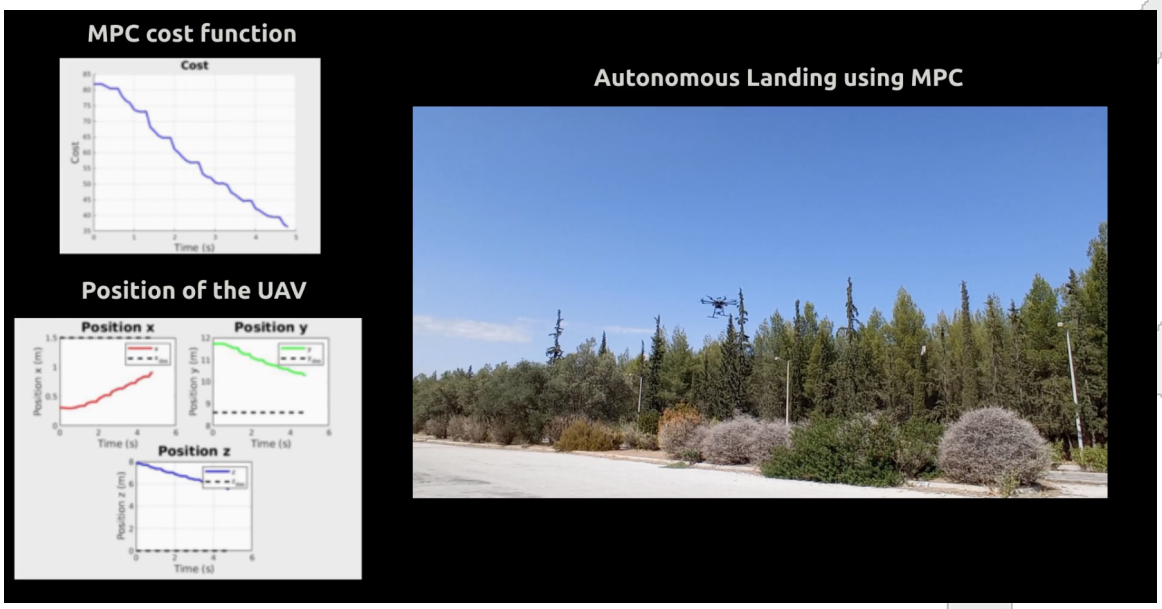



Figure 26: An overall view of the Landing Experiment I.

**Experiment II**

The second experiment was carried out in a similar manner in outdoor environment, where the multi-rotor platform was positioned above an area for which the on-board camera feed included a more complex morphology, with heightened obstacles and patches of rough dirt-like terrain. A frame from the respective position is depicted in Fig. 27. The respective depth map is depicted in Fig. 28, while the resulting cost map is depicted in Fig. 29 along with the appropriate landing area as determined by the algorithm, depicted in a green ball. As expected, the algorithm identifies a suitable landing spot.

The landing process is also carried out in the same manner as in Experiment I. The response of the system is depicted in the diagrams of Fig. 30, while the evolution of the MPC cost w.r.t. time is depicted in Fig. 31. It is evident that the controller exhibits satisfactory performance w.r.t. to task execution and trajectory response. An overview of the experiment is presented in Fig. 32 and in the following video `https://youtu.be/ZVU_mZ6rZYY`.

Figure 27: Image of the outdoor environment of Landing Experiment II captured by the on-board camera.

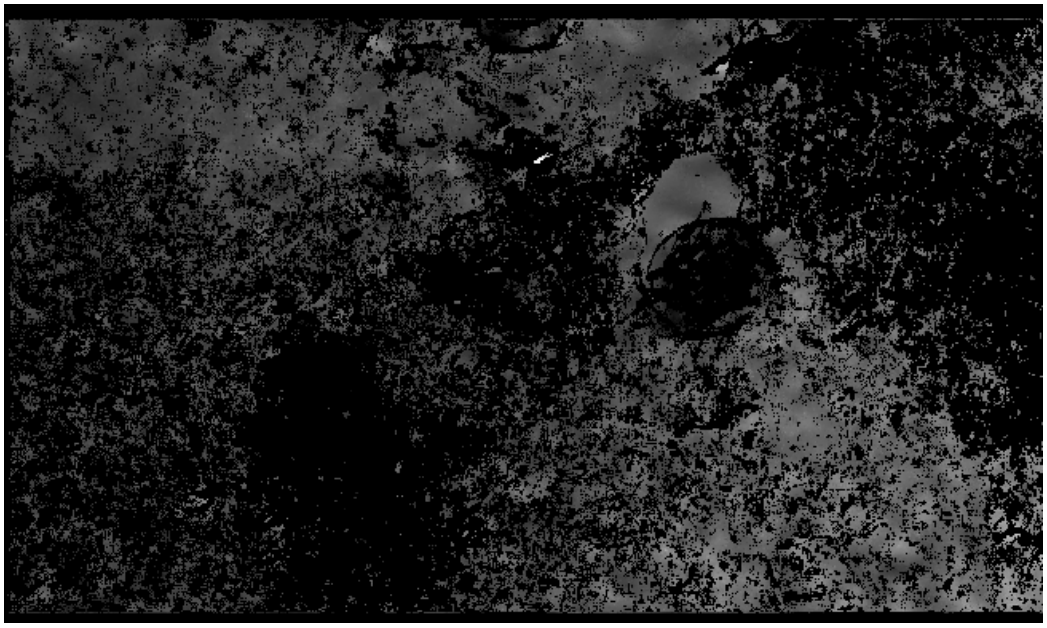


Figure 28: Depth Image of the outdoor environment obtained by the downward-looking ZED2 during the Landing Experiment II.

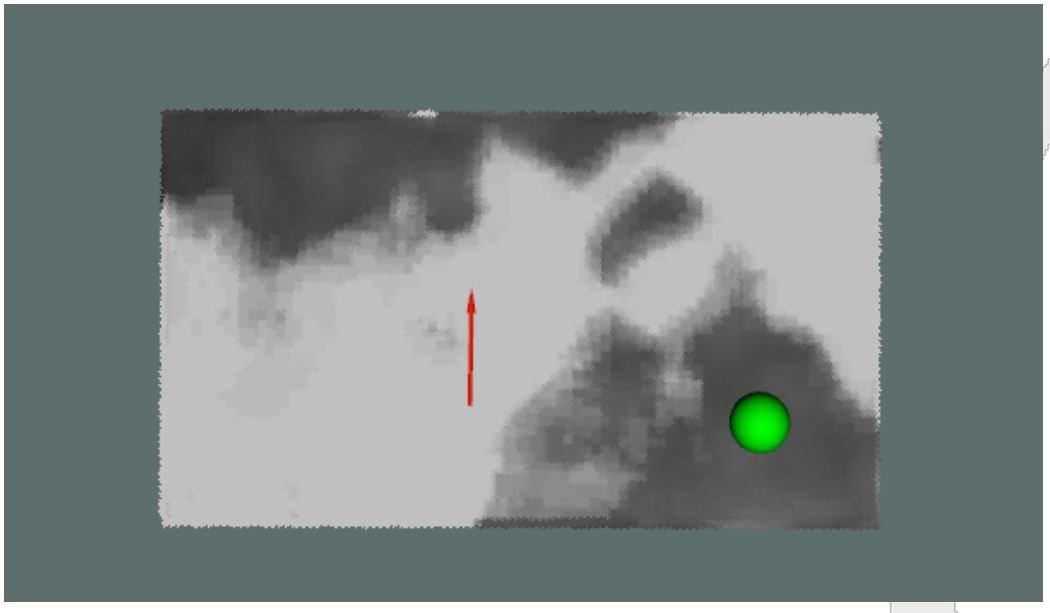
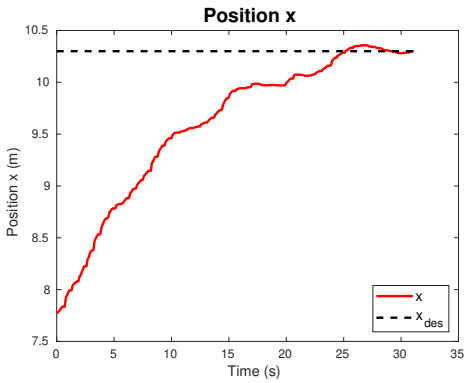
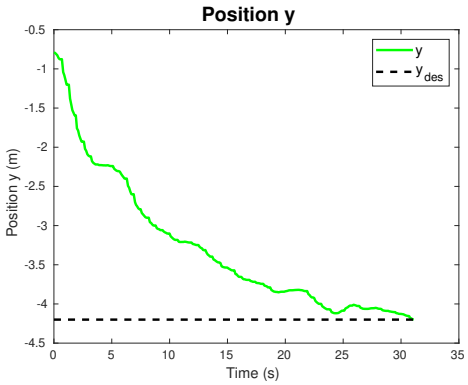


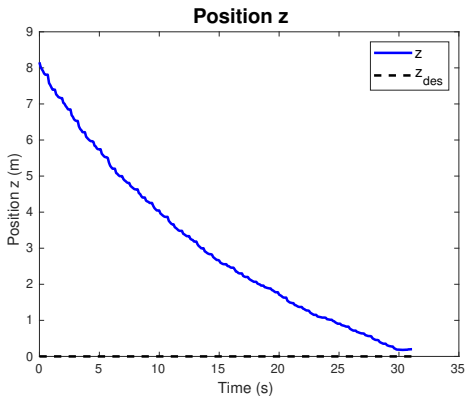
Figure 29: Occupancy Grid, built in real time according to Section 8, and the best landing spot (marked as a green sphere) during the Landing Experiment II.




(a) Position x




(b) Position y




(c) Position z

Figure 30: The position ${}^{I}\mathbf{p}$ of the vehicle with respect to the map frame $\mathbf{I}$, compared to the desired landing spot ${}^{I}\mathbf{p}_{des}$ during the Landing Experiment II.

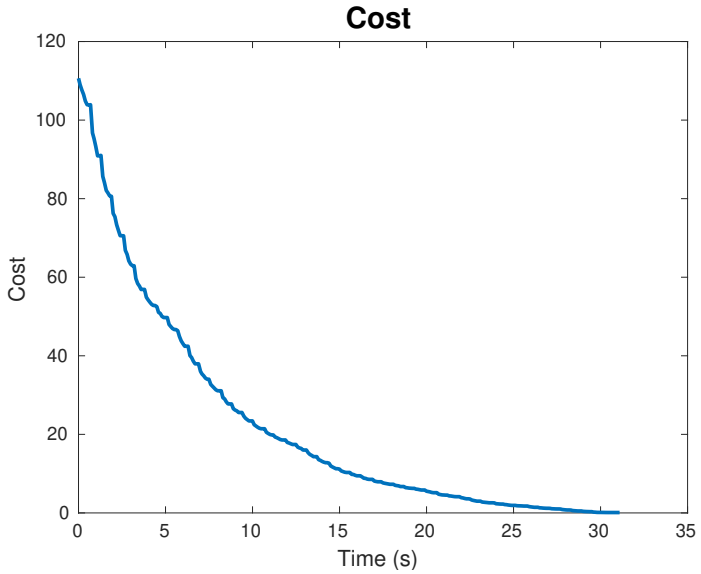



Figure 31: Time evolution of MPC cost function during the Landing Experiment II.

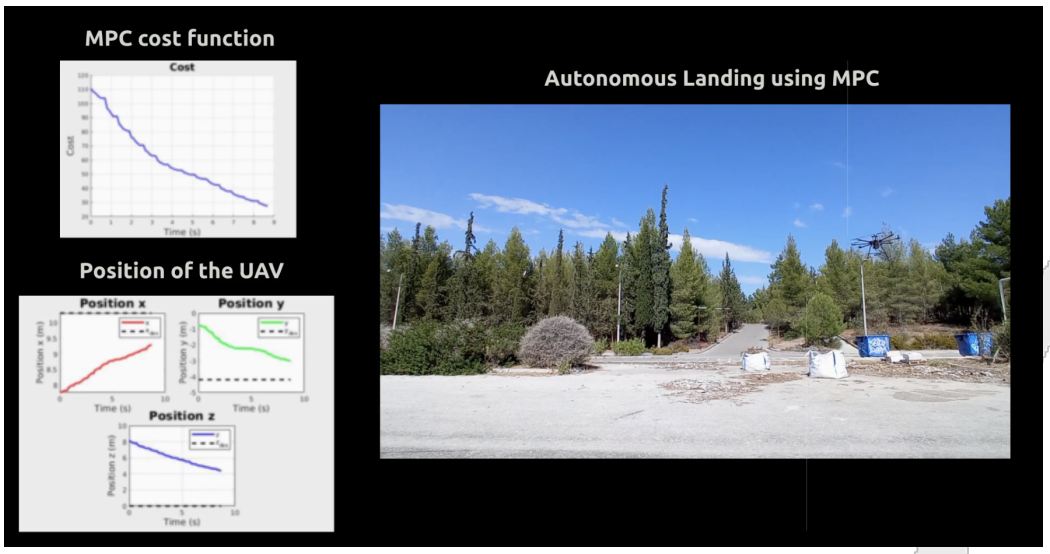



Figure 32: An overall view of the Landing Experiment II.

**Experiment III**

The third experiment was carried out in a coastal outdoor environment, in a similar manner to the previous experiments, although the conditions (wind, lighting) were more challenging. A frame from the respective position is depicted in Fig. 33. The respective depth map is depicted in Fig. 34, while the resulting cost map is depicted in Fig. 35 along with the appropriate landing area as determined by the algorithm, depicted in a green ball. As expected, the algorithm identifies a suitable landing spot.

The landing process is also carried out in the same manner as in Experiment I. The response of the system is depicted in the diagrams of Fig. 36, while the evolution of the MPC cost w.r.t. time is depicted in Fig. 37. It is evident that the controller exhibits satisfactory performance w.r.t. to task execution and trajectory response. An overview of the experiment is presented in Fig. 38 and in the following video `https://youtu.be/_d4rWiSVFug`.

Figure 33: Image of the outdoor environment of Landing Experiment III captured by the on-board camera.

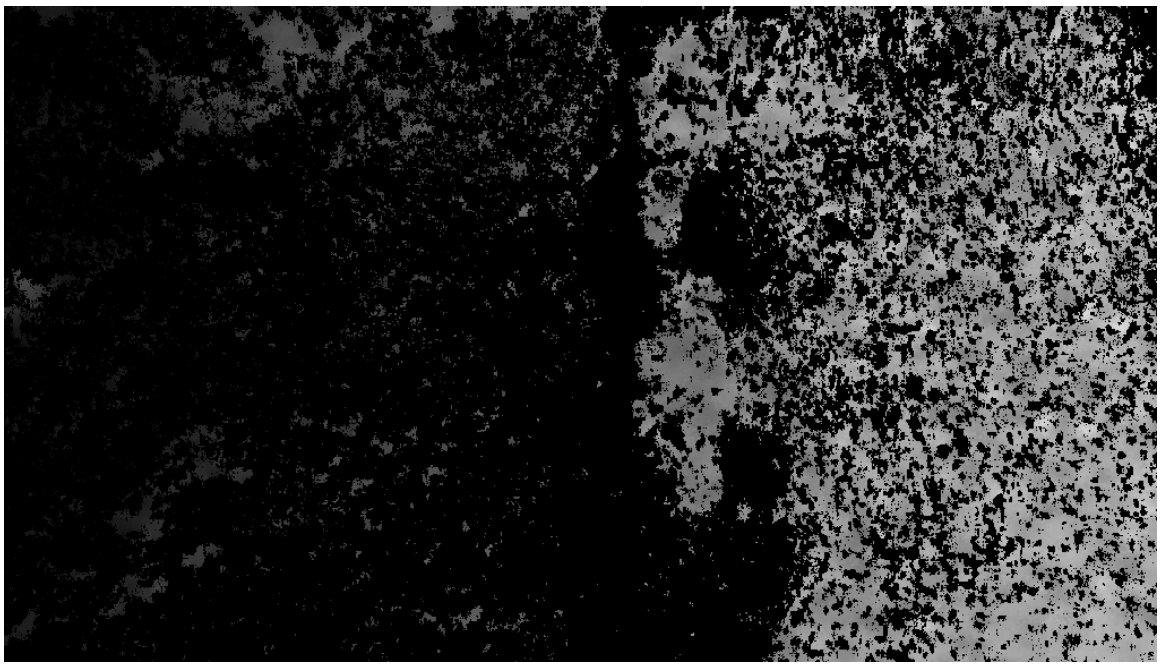


Figure 34: Depth Image of the outdoor environment obtained by the downward-looking ZED2 during the Landing Experiment III.

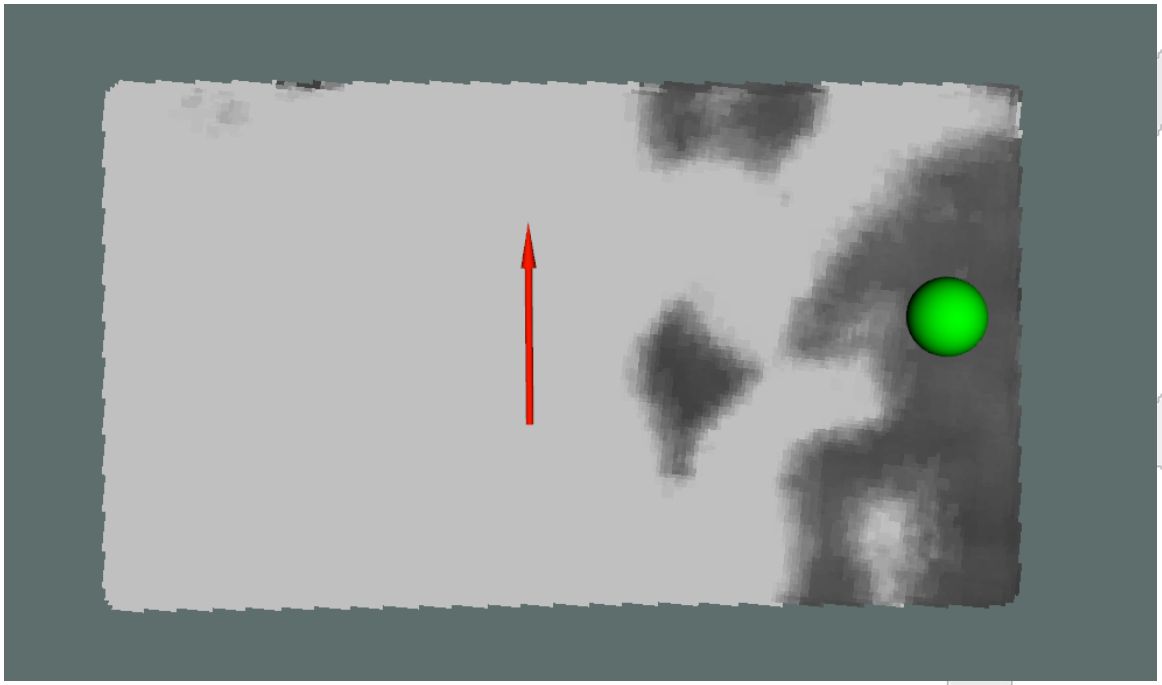
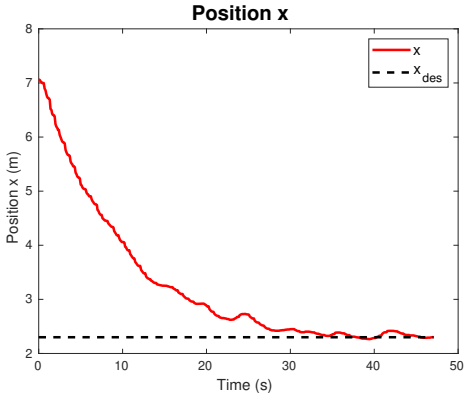
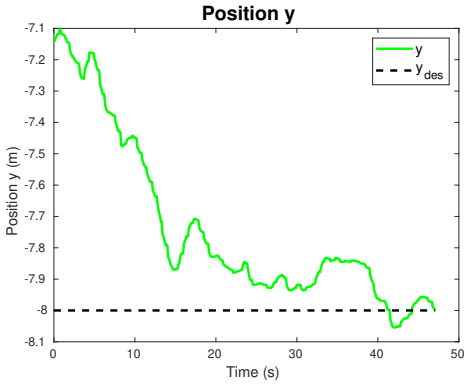


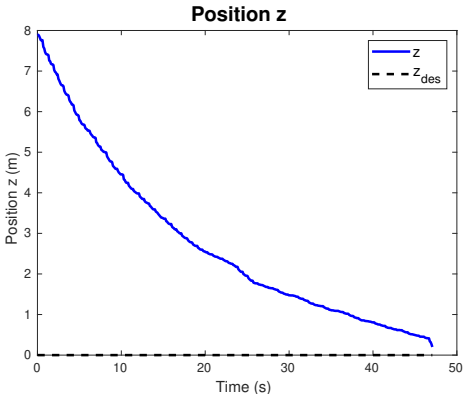
Figure 35: Occupancy Grid, built in real time according to Section 8, and the best landing spot (marked as a green sphere) during the Landing Experiment III.




(a) Position x




(b) Position y




(c) Position z

Figure 36: The position ${}^{I}\mathbf{p}$ of the vehicle with respect to the map frame $\mathbf{I}$, compared to the desired landing spot ${}^{I}\mathbf{p}_{des}$ during the Landing Experiment III.

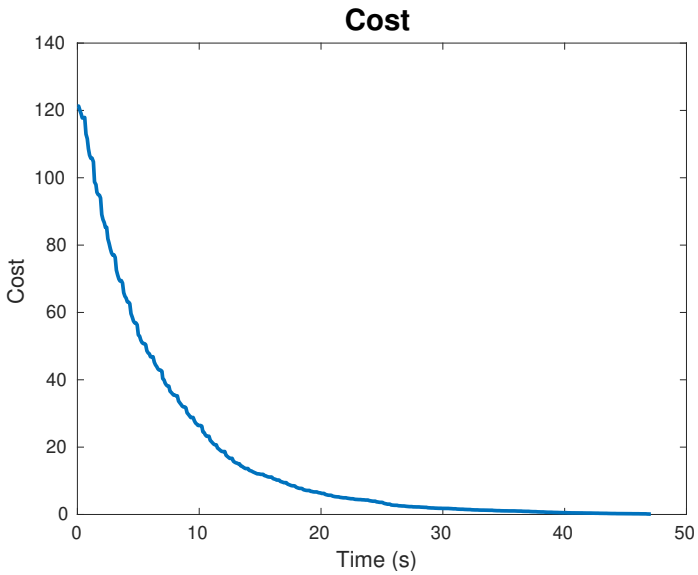



Figure 37: Time evolution of MPC cost function during the Landing Experiment III.

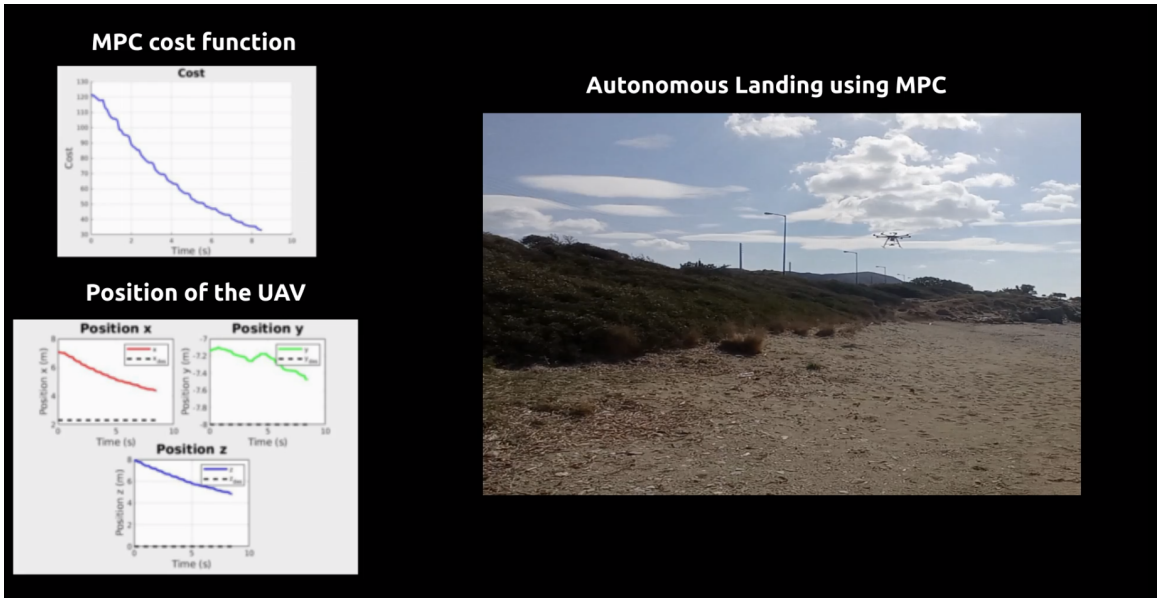



Figure 38: An overall view of the Landing Experiment III.

## 11.2 Water Maximization Experiments

In this subsection, the water maximization experiments are demonstrated. The goal of the latter is to demonstrate the ability of the controller to lead the drone to a position where the on-board camera detects only water, so as to facilitate the sampling process. At this point, we should highlight that the actual water sampling process is part of *D5.3: Drone robust control for water sampling*, however, the identification of appropriate sampling locations is a topic covered by D5.2. Both experiments took place in a remote beach where wind and lighting conditions were highly varying, making both the control and detection tasks more challenging. The two experiments were carried out to demonstrate the efficacy of the proposed schemes with different drone orientations, even in the presence of noisy sensing (i.e., waves, light reflections, etc), as the orientation might be relevant to the sampling task, or any other process that is part of the platform's mission, and is thus not restricted by the proposed scheme.

### Experiment I

In the first experiment the multi-rotor was placed in a position facing normal w.r.t. the wave-front. An overview of the experiment is depicted in Fig, 40, while the visible water metric, in the form of percentage of water pixels, is depicted in Fig. 39. The non-monotonic evolution of the metric, which is not to be expected from the Lyapunov analysis carried out in the relevant technical section -owing to asymptotic convergence- is due to the un-modeled wave oscillations. Nevertheless, the algorithm is robust and quick enough so as to properly stabilize the system in the desired position. The algorithm is successful in maximizing the water that is visible to the camera, while the components of the algorithm (i.e., sensing, control) were highly satisfactory during the entire execution of the algorithm. The performance of this scheme is better illustrated in the relevant video (`https://youtu.be/xn2X1tm9yKk`).

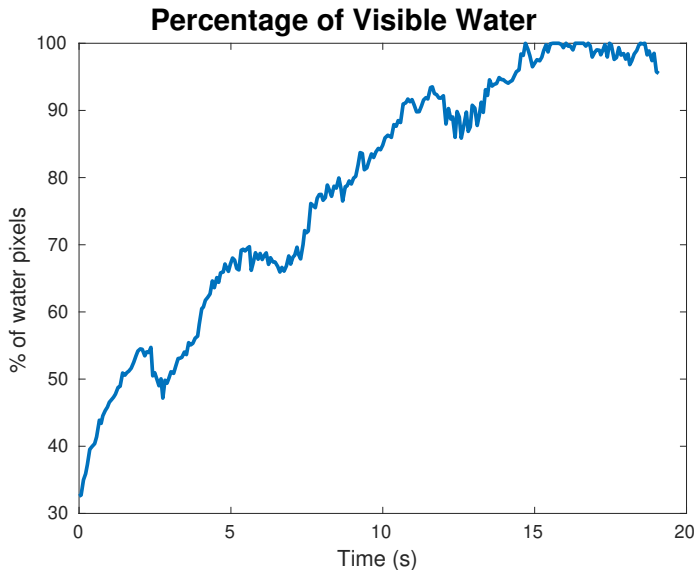



Figure 39: Time evolution of the percentage of visible "water" pixels during the Water Maximization Experiment I.

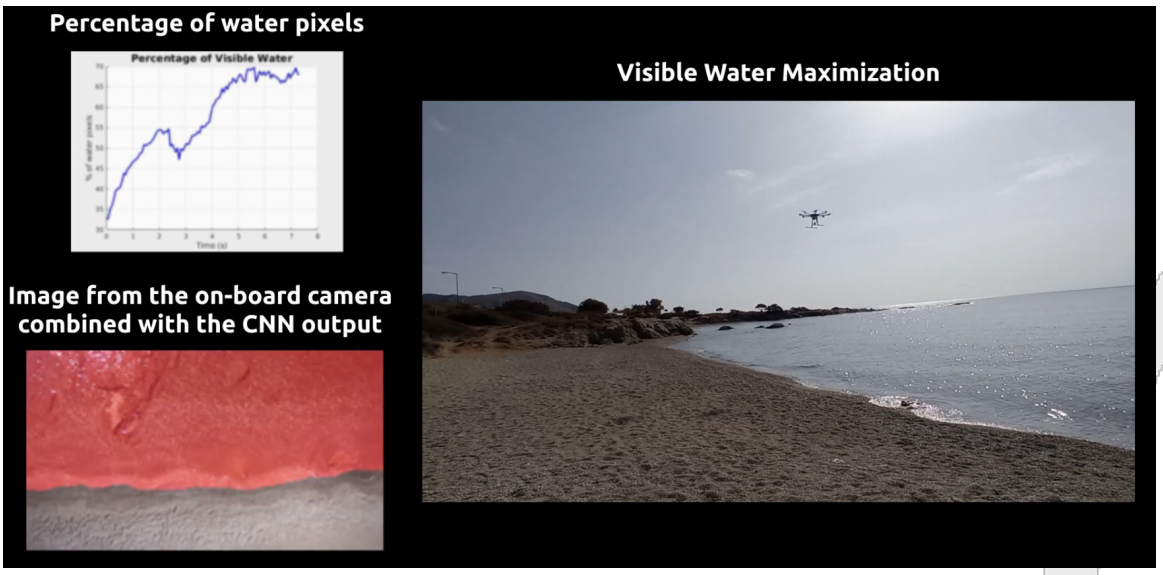



Figure 40: An overall view of the Water Maximization Experiment I.

### Experiment II

In the second experiment the multi-rotor is setup in an identical manner with the sole difference that it was placed in a position facing at an angle w.r.t. the wave-front. An overview of the experiment is also depicted in Fig, 42, with the visible water metric, in the form of percentage of water pixels, depicted in Fig. 41. The algorithm is again successful in maximizing the water that

is visible to the camera. The performance of this scheme is better depicted in the following video (`https://youtu.be/zPOg5yJulTg`).

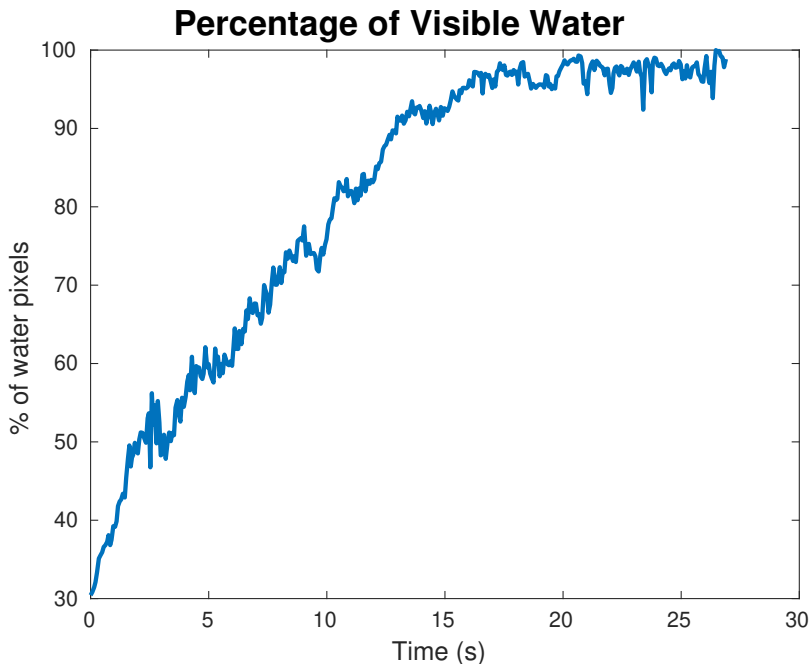



Figure 41: Time evolution of the percentage of visible "water" pixels during the Water Maximization Experiment II.

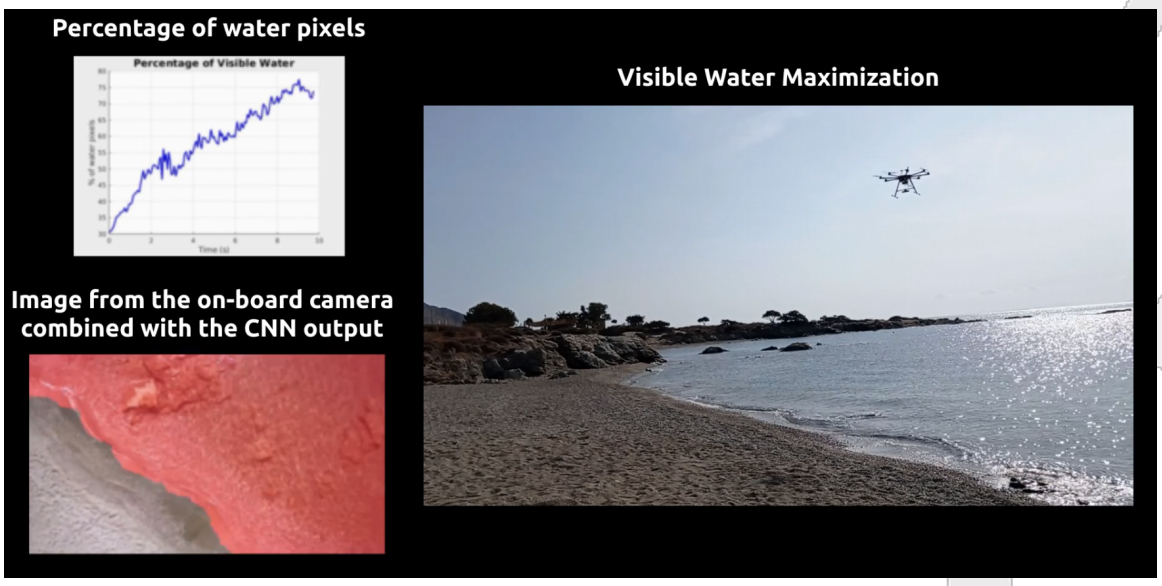



Figure 42: An overall view of the Water Maximization Experiment II.

# 12 Conclusion

In D5.2 a comprehensive set of algorithms for tackling the aspects that are pivotal to a First Responder's mission execution were developed. A novel algorithm for autonomous landing in *a priori* unknown environments utilizing on-board sensing and computing was formulated. Additionally, a provably correct scheme for maximizing the water area below the drone was formulated that asymptotically maximizes the visible water area to the multi-rotor's on-board camera field of view. Both schemes were built up by freely distributed open source software that is readily available and easy to integrate with any state-of-the-art platform.

Both frameworks were demonstrated to work both in extensive simulations and in real-world field experiments, where the robustness and efficiency of the proposed framework was put to the test. Given the promising results, these modules can be employed in a modular fashion to support any mission that is relevant to the scope of deliverable D5.2 as well as to the rest of the PathoDRONE functionalities related to Task 5.1.

# Bibliography


[1] Ardupilot. http://www.ardupilot.org/.

[2] Jetson xavier. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.

[3] Pixhawk. https://pixhawk.org/.

[4] Zed stereo camera. https://www.stereolabs.com/zed/.

[5] Roman Barták, Andrej Hrasko, and David Obdrzalek. A controller for autonomous landing of ar.drone. pages 329–334, 05 2014.

[6] Andrea Cesetti, Emanuele Frontoni, Adriano Mancini, Primo Zingaretti, and Sauro Longhi. A vision-based guidance system for uav navigation and safe landing using natural landmarks. *Journal of Intelligent and Robotic Systems*, 57:233–257, 01 2010.

[7] Lyujie Chen, Xiaming Yuan, Yao Xiao, Yiding Zhang, and Jihong Zhu. Robust autonomous landing of uav in non-cooperative environments based on dynamic time camera-lidar fusion, 2020.

[8] Divam Gupta. A beginner's guide to deep learning based semantic segmentation using keras, 2019.

[9] Yuan Haiwen, Changshi Xiao, Supu Xiu, Wenqiang Zhan, Zhenyi Ye, Fan Zhang, Chunhui Zhou, Yuanqiao Wen, and Qiliang Li. A hierarchical vision-based localization of rotor unmanned aerial vehicles for autonomous landing. *International Journal of Distributed Sensor Networks*, 14:155014771880065, 09 2018.

[10] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[11] G. C. Karras, C. P. Bechlioulis, G. K. Fourlas, and K. J. Kyriakopoulos. Target tracking with multi-rotor aerial vehicles based on a robust visual servo controller with prescribed performance. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 480–487, 2020.

[12] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29:315 – 378, 03 2012.

[13] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[14] Weiwei Kong, Daibing Zhang, Xun Wang, Zhiwen Xian, and Jianwei Zhang. Autonomous landing of an uav with a ground-based actuated infrared stereo vision system. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2963–2970, 2013.

[15] Sven Lange, Niko Sünderhauf, and Peter Protzel. Autonomous landing for a multirotor uav using vision. *In Workshop Proceedings of SIMPAR 2008 Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots*, 01 2008.

[16] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, 2012.

[17] Rafik Mebarki, Vincenzo Lippiello, and Bruno Siciliano. Autonomous landing of rotary-wing aerial vehicles by image-based visual servoing in gps-denied environments. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2015.

[18] Md Shah Alam and Jared Oluoch. A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (uavs). *Expert Systems with Applications*, 179:115091, 2021.

[19] Stanford Artificial Intelligence Laboratory et al. Robot operating system.

[20] Tao Yang, Peiqi Li, Huiming Zhang, Jing Li, and Zhi Li. Monocular vision slam-based uav autonomous landing in emergencies and unknown environments. *Electronics*, 7(5), 2018.