D5.1 — Drone Navigation and Motion Planning Strategies

WP5 — Contamination Situation Awareness

April 29, 2021

Authors:
Fotis Panetsos,
Panagiotis Rousseas,
Dr. George C. Karras,
Dr. Charalampos P. Bechlioulis,
Prof. Kostas J. Kyriakopoulos

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	D5.1 – Drone Navigation and Motion Planning Strategies			
WORK PACKAGE	WP5 – Contamination Situation Awareness			
DATE OF DELIVERY	CONTRACTUAL	30/04/2021	ACTUAL	30/04/2021
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 (all at the NTUA)			
ABSTRACT	The Drone Navigation and Motion Planning Strategies report (D5.1), provides an analytical description of the methods, algorithms, software and hardware infrastructure employed to support the autonomous motion planning and control functionalities of the PathoDRONE tool. The main objective is the integration of friendly Graphical User Interfaces provided by common mission planners, that can be easily operated by *First Responders*, with an efficient motion planning and control framework running on-board the drone. Hence, the First Responders will be able to plan missions by simply clicking the desired locations on the map of a common and intuitive mission planner software, without considering possible obstacles in the path of the multirotor, since autonomous navigation, efficient motion planning, collision and obstacle avoidance will be handled by the drone's on-board algorithms.			

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
1.0	08.04.21	Draft	A first complete draft of D5.1 is shared with partners for feedback, revision and information update.	NTUA
2.0	15.04.21	Draft	A full review of the completed draft of D5.1.	CERTH
3.0	18.04.21	Draft	A full review of the completed draft of D5.1.	UCY
4.0	23.04.21	Final	Final version of D5.1.	NTUA

PathoCERT

Disclaimer

Copyright message

PathoCERT

Contents

List of Figures

1 Introduction

Water is, and always will be, one of the most valuable resources for humanity. In the context of the modern climatic, ecologic and industrial paradigm, where pollution of water bodies is unfortunately becoming more and more common, safeguarding the water resources of our planet is of utmost importance. Additionally, the dangers posed to human health when responding to such threats necessitates extreme care in the way first responders tackle an emergency.

Specifically during field operations, first responders may need to operate within an unknown or uncertain water environment. They may need to map the extend of a flush-flood or the boundaries of a lake where their will conduct search-and-rescue, to improve their situational awareness. This should be done as efficiently as possible, under various environmental conditions. At the same time, the may use different types of sensors to collect information regarding water quality, to evaluate the potential threats for both first responders and civilians, and take the most appropriate protection measures.

In this regard, employing modern technologies or developing new ones to tackle these emerging problems is essential, if we are to effectively protect this treasured resource, while minimizing the danger posed to first responders, civil infrastructure and civilian endangerment. The goal of the PathoCERT project is to capture the complete spectrum of waterborne pathogen contamination management from detection and situation awareness, to epidemiological, threat risk assessment and criminal investigation, via a set of tools and methods available to first responders in emergency water-contamination situations.

A vital part of the PathoCERT framework is the PathoDRONE tool, an **autonomous drone swarm solution** along with an **innovative system for water sample collection**, suitable for various conditions (water in rivers, flush-flooding or lakes) as well as environmental conditions. The PathoDRONE tool, addresses some key issues in the struggle against water pollution. In a timely period, where drone technologies, sensing capabilities, on-board computing power and Artificial Intelligence have advanced in their respective manner, there exists a framework on which applicable technological solutions can be based. The deployment of robots in daily human operations is now closer to materializing than ever. The advantages of a shift towards autonomous solutions are extremely evident in applications such as responding to natural disasters, pollution incidents, where case-specific complications may lead to endangering first responders, or limit human accessibility to a site.

Airborne multi-rotor platforms have long since demonstrated extreme versatility. While, due to limitations in energy storage technologies, most platforms provide limited time on air, along with a narrow window of weather conditions suitable for flight, the advantages these platforms present can not be easily neglected. Some platforms exhibit great maneuvering capabilities with great variety in sizes and power, while being able to be deployed almost everywhere. In many real-world scenarios, versatility and adaptability are key, with response time being also of the essence. Even with such strict requirements, modern multi-rotor platforms provide a proper solution, enabling responders to not only maintain a safe distance from a dangerous on-site environment, but also to be more efficient, swift and effective, and

even to intervene in cases previously deemed impossible.

In conclusion, PathoDRONE is an essential, timely and well-motivated step towards autonomous solutions that answer to a grave modern set of perils, while safeguarding human health and environmental welfare. In the PathoCERT project this will be achieved through *Task 5.1: Drone-based situation awareness system with water sampling capabilities* and its relative sub-tasks. The deliverable *D5.1 Drone navigation and motion planning strategies*, presented herein, is part of *Subtask 5.1.1 Sensor-based navigation, motion planning, take-off/landing and robust control*, and its purpose is to describe the tools and methods developed for the drone autonomous navigation and motion planning within the PathoCERT project. Specifically for the small heights, real-time detection and avoidance of obstacles is realized and incorporated within the motion controller of the aerial vehicle utilizing feedback from the on-board perception sensors (e.g., LiDAR, RGB-D camera).

2 Related Literature

As previously delineated, autonomous multi-rotor aerial platforms have been widely recognised for their versatility and applicability [7], with many platforms that can through their diversity address a wide range of problems [10]. These airborne platforms consist a feat of multi-disciplinary engineering, implementing experience and technical knowledge from almost every engineering paradigm, from manufacturing and structural mechanics, to aerodynamics and control. More recently, advances in hardware and software design and optimization have enabled engineers to adopt a more holistic approach towards aerial autonomy, where the aerial platform includes all relevant computing power and software on-board. Drone platforms are now more autonomous than ever, with advances in communication theory [19] enabling fast and reliable communication, with developments in AI and visual servoing (vision-based robot control) paving the way towards enhanced sensing capabilities [16]. Furthermore, control theory has caught up with the demand for robust non-linear control tools, e.g., non-linear model predictive control, taking advantage of the enhanced computational power of modern computers. Regarding the application of drones in the context of modern societies, the concept of smart cities and more generally of the Internet-of-Things (IoT) provides new ways for integrating drone technologies in modern life [6, 21].

Commercially available drones have been introduced and widely adopted over the past few years. This fact demonstrates how efficient, effective and robust the design of such platforms has become. With regards to the research community, the open-source philosophy of a large part of the community has been extremely effective in democratising several aspects of the process of setting-up and flying an aerial platform. Concerning both the low level [1] and the high-level [20] aspects of drone control there exist robust, almost "plug-and-play" solutions. Furthermore, several simulation solutions exist in order to facilitate the testing and tuning of the software, as well as to enable sensing and hardware simulations [15]. The software-in-the-loop (SITL) mindset enables engineers to transition from simulator to real-world platforms almost effortlessly.

Along with advances in drone control capabilities, the recent improvements on hardware have enabled the adoption of advanced motion-planning schemes in such autonomous platforms. Nowadays, sampling-based methodologies have been widely adopted, owing to their robustness and relative simplicity. Some of the most famous planning algorithms include the RRT* (Rapidly-exploring random tree) algorithm for known workspaces [17], the A* algorithm for unknown workspaces [18], Dijkstra's algorithm [11], etc. These algorithms are widely incorporated in many open source frameworks in applied robotics, including the ROS framework. The computational power of modern on-board flight computers have enabled the adoption of such search-based algorithms, whose computational complexity would be prohibitive in the past.

In summary, the existing frameworks, technologies and algorithms enable the formation of real-world solutions employing multi-rotor aerial platforms. The PathoDRONE tool demonstrates how the above technical background is implemented in order to provide an applicable framework for solving a variety of problems.

3 Problem Statement

The main objective of Deliverable 5.1. is the integration of the friendly Graphical User Interface provided by the Mission Planner with the use of a ROS-based motion planning package, namely "move_base". Hence, the First Responders will be able to plan missions by simply clicking the desired locations on the map of Mission Planner without worrying about possible obstacles during the navigation of the multirotor. A local planner will be responsible for computing appropriate velocity commands which ensure the safe execution of the mission and the avoidance of undesirable collisions. It is highlighted that the absence of a local planner results in straight paths between the waypoints and, consequently, intermediate obstacles might jeopardize the safety and, thus, the success of the operation. To illustrate the importance of this framework, the mission of Figure 1 is considered. In this particular instance, a tree lies between the two target waypoints. Without the integration of a local planner, the vehicle would follow the yellow straight path, which intersects with the tree, and, as a result, a collision would occur. Conversely, the existence of a local planner results in the purple path which ensures the safe execution of the mission.

The aforementioned framework requires an additional step, responsible for translating the desired waypoints on the map to proper target goals for the move base ROS package. In order to accomplish this, the MAVROS package[1] is used so as to access the waypoints that are inputed by the user through the Mission Planner GUI. MAVROS is a ROS package which enables the communication between ROS and autopilots or Ground Control Stations with MAVLink communication protocol, e.g., Ardupilot and Mission Planner.

The waypoints are defined in the *World Geodetic System 1984* (WGS84) and, as a consequence, a conversion is performed to the Inertial coordinate frame **I**. Afterwards, the waypoints are successively sent as target goals to the move base ROS package until the vehicle completes the commanded mission. An overview of the above framework is shown in Figure 2.

A detailed analysis of the individual parts, which constitute the framework of Figure 2, is performed in the following sections.

[1] http://wiki.ros.org/mavros

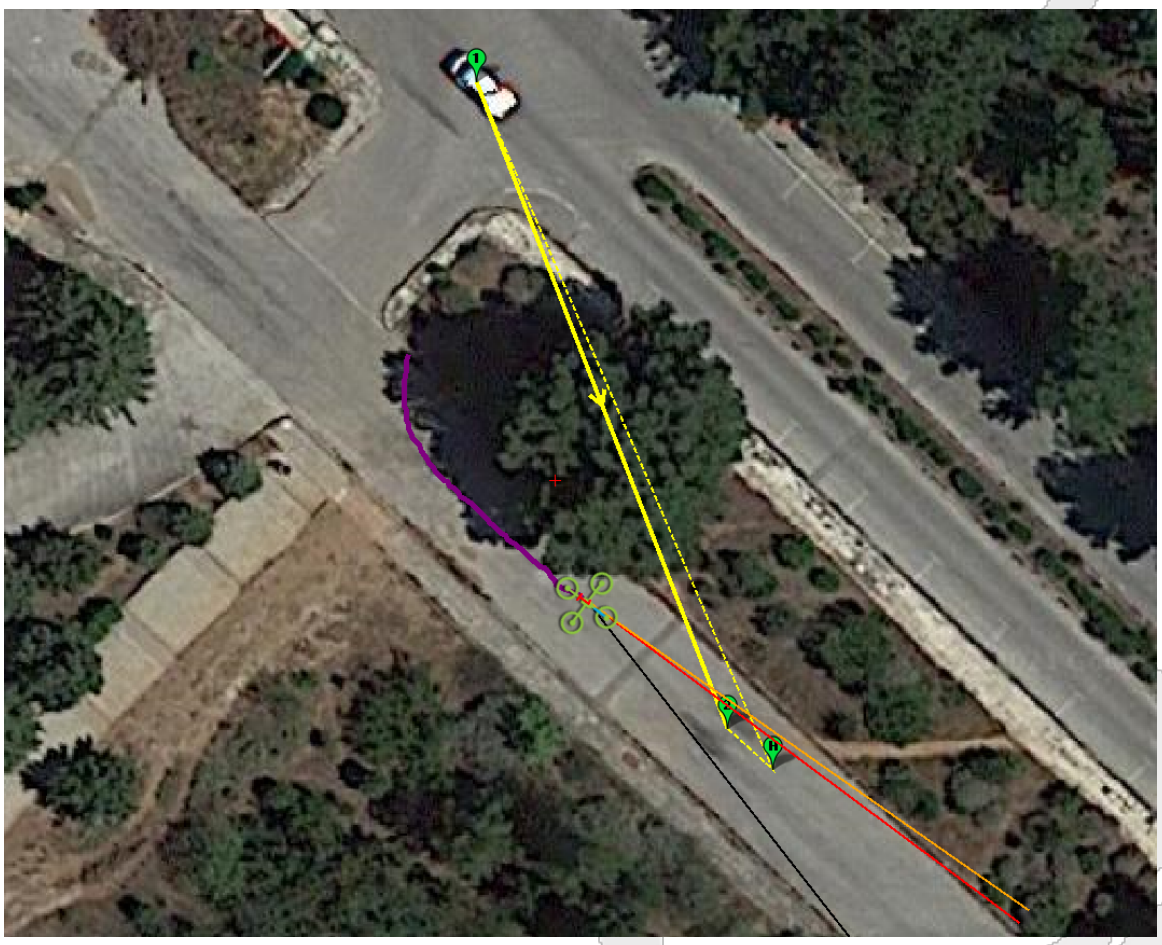

Figure 1: Example of a mission

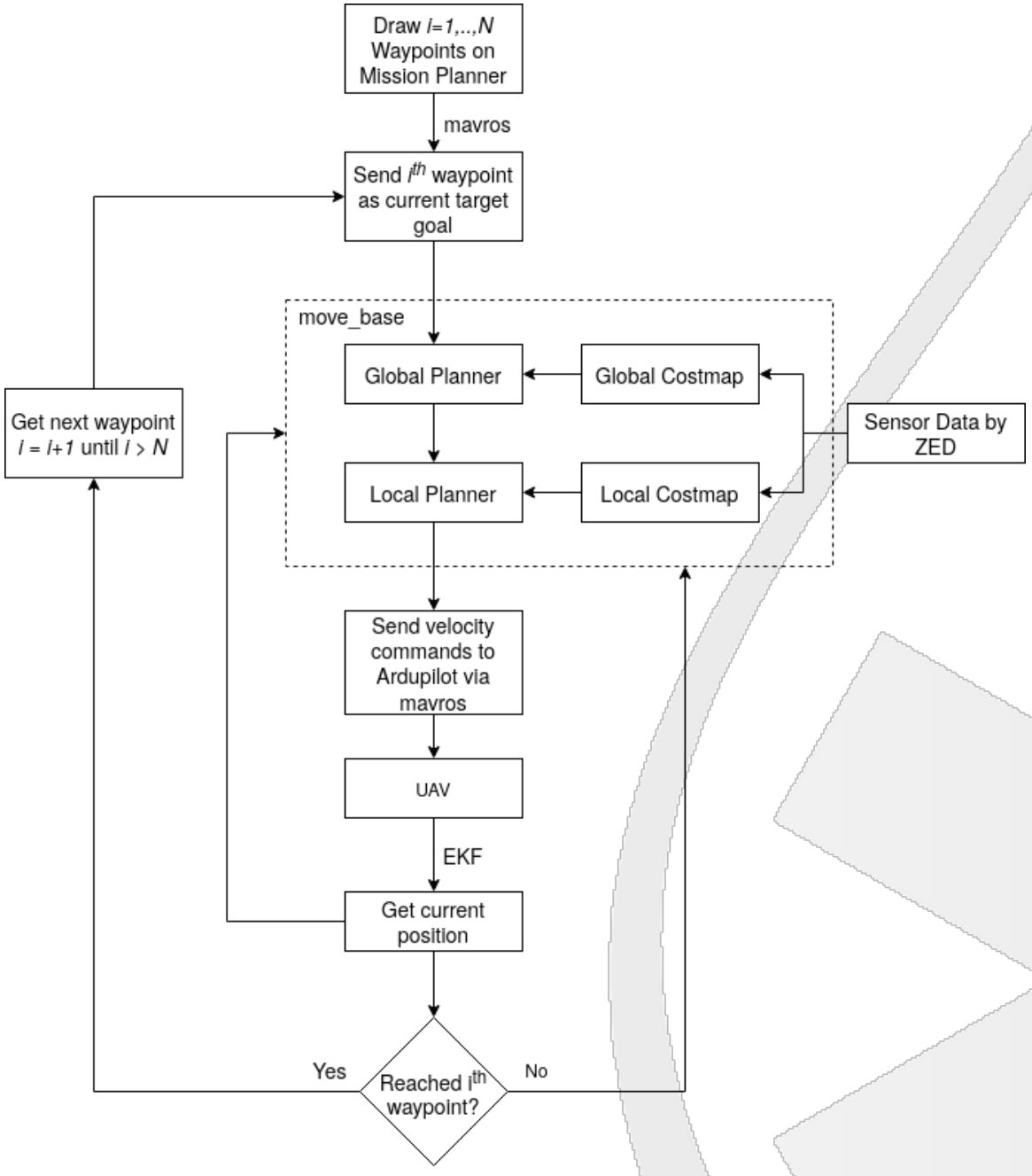

Figure 2: Integration of Mission Planner with the "move_base" ROS package

4 PathoDRONE Supporting Infrastructure

4.1 Vehicle Description

The *Unmanned Aerial Vehicle* (UAV) is a crucial part of the overall PathoDRONE tool and, thus, a vehicle which satisfies the requirements of this project is necessary for the success of the operation. The *NTUA octorotor* (Figure 3) is a complicated robotic system, composed of multiple parts, which turn it into a powerful and fully autonomous Unmanned Aerial Vehicle. More precisely, the NTUA octorotor is loaded with the Ardupilot firmware [1], responsible for controlling the aircraft through all regimes of flight. Ardupilot runs on the Cube Pixhawk 2.1 autopilot [4], 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.

Figure 3: NTUA octorotor

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.

The safe navigation of the NTUA octorotor requires a sensor capable of measuring the distance between the vehicle and the obstacles and, thus, the ZED 2 stereocamera is used. Additionally, the execution of computationally expensive algorithms such as image processing or occupancy grid mapping is a necessary prerequisite and, consequently, 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) [20] and, particularly, through the MAVROS node which provides communication between ROS and ArduPilot vehicles.

4.2 Autopilot

In order to effectively control the dynamics 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 ${}^{I}\mathbf{p}_{B_d}$, velocity ${}^{I}\mathbf{v}_{B_d}$ and heading ψ_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 is, eventually, translating 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 4.

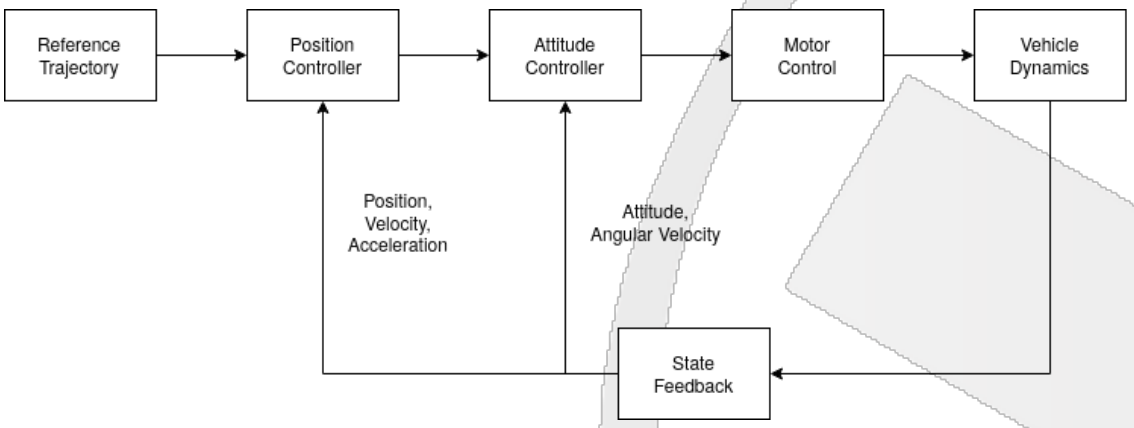

Figure 4: Ardupilot control architecture

4.3 Mission Planner

A Ground Control Station (GCS) is an essential tool in order to establish communication with an Unmanned Aerial Vehicle (UAV) and monitor the flight status. Among the commercially available GCS software, the widely adopted Mission Planner [3] offers a variety of desirable features which facilitate the flight experience. More specifically, Mission Planner provides a friendly Graphical User Interface (GUI) which displays real-time data concerning the vehicle performance and position (Figure 5). The communication between any aerial vehicle and the Mission Planner is achieved via wireless telemetry.

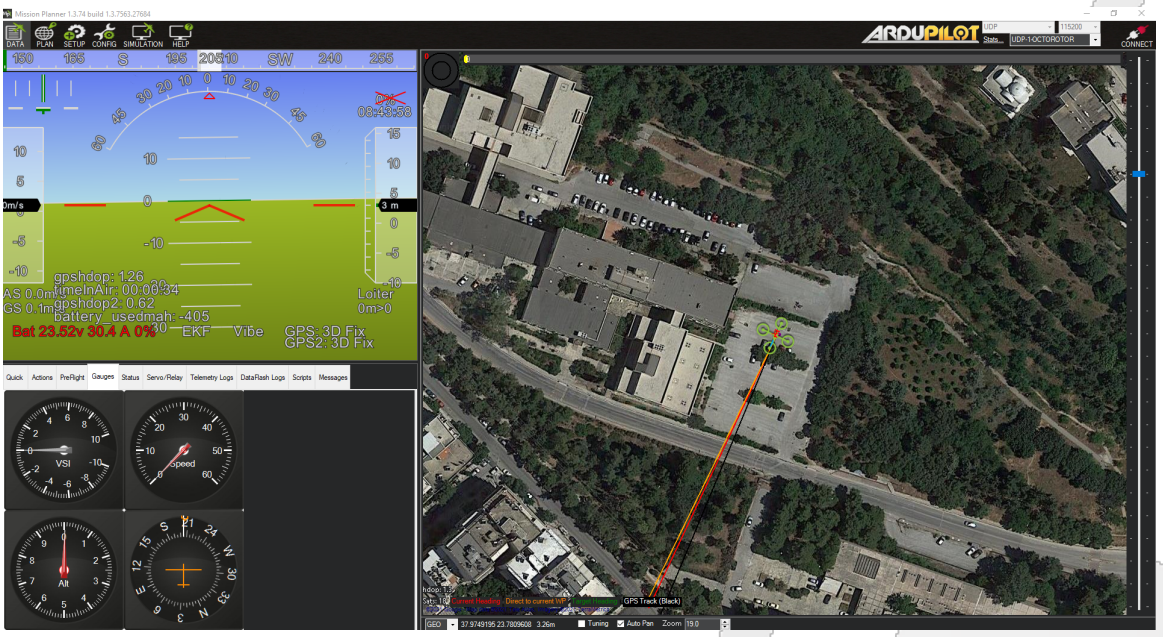

Figure 5: Mission Planner GUI

The Mission Planner environment includes additional features such as parameter tuning, recording and analyzing of telemetry logs. A notable option, given by Mission Planner, is the possibility to plan a mission by drawing a list of desired waypoints on the provided map, which displays a real-world satellite view of the area in the vicinity of the drone's home (take-off) position (Figure 6). Afterwards, the Ardupilot side is responsible for navigating the multirotor to the commanded locations and executing the whole mission autonomously.

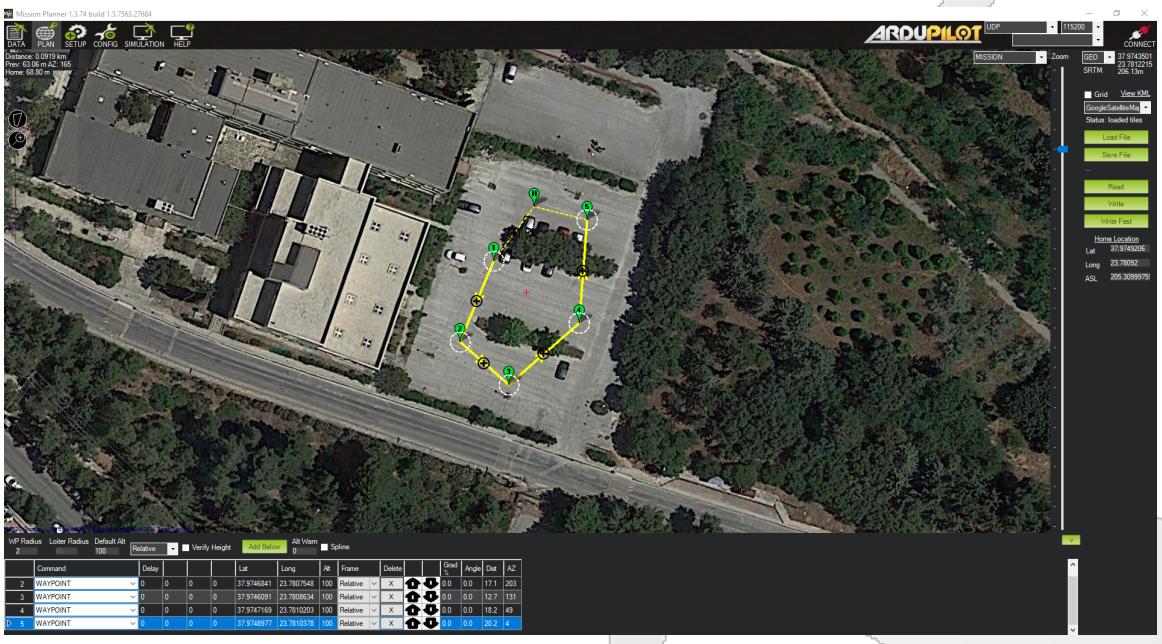

Figure 6: Planning a mission

4.4 Sensing

Although the framework of the Mission Planner software, using the Ardupilot, supports the execution of autonomous missions, there is no collission avoidance provision for any possible obstacles that might intersect the robot's pre-planned trajectory during the transition of the multirotor from one waypoint to another. Consequently, the integration of a local planner, responsible for recomputing the path in the presence of obstacles, is inevitable in order to guarantee the safe navigation of the NTUA octorotor during the whole operation.

An essential prerequisite is the existence of a sensor, capable of reliably measuring the distance between the vehicle's frame of reference and any possible obstacles. Hence, the NTUA octorotor is equipped with the ZED 2 stereocamera [5] (Figure 7), 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 7: 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. Each cell is marked as free, unknown or occupied according to the existence of a detected obstacle. The occupancy grid is augmented with a user defined inflation radius and, consequently, a 2D costmap is generated, as shown in Figure 8. The inflation serves as a safety factor by assigning cost values around the obstacles which decrease with distance. Eventually, a safe area between the multirotor and the surrounding environment is detected at each time instant. During the mission, the NTUA octorotor should always navigate inside the latter in order to avoid collisions with obstacles.

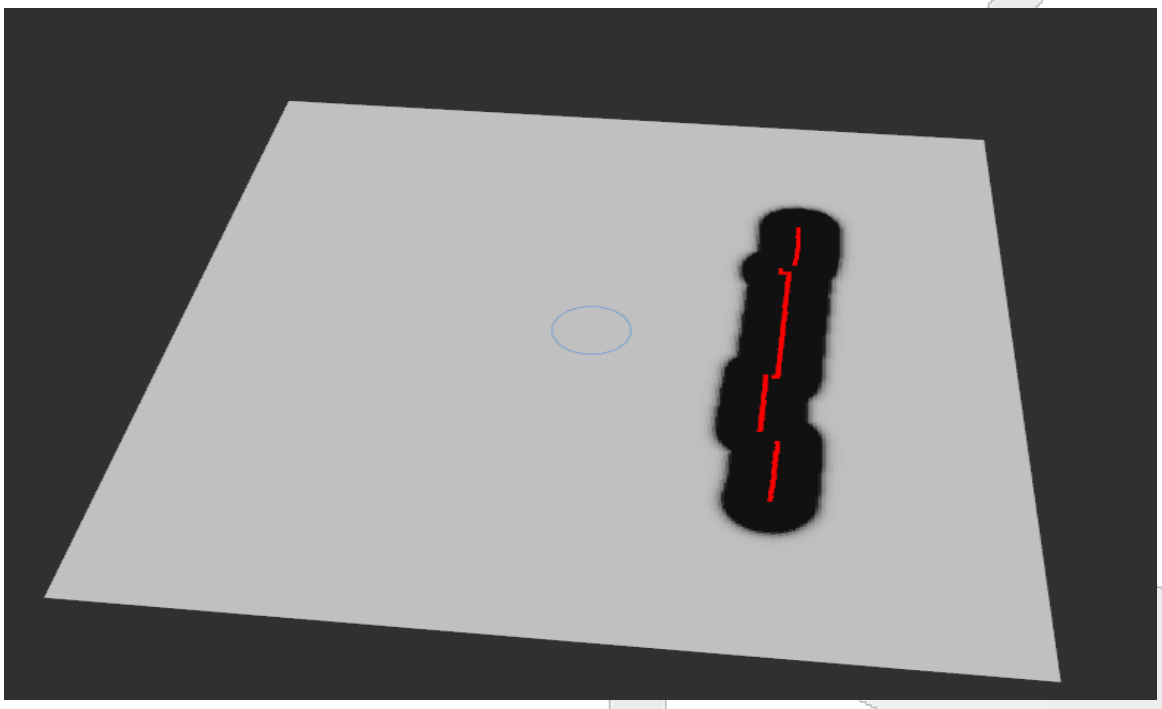

Figure 8: An example of a costmap using the ROS vizualization tool. The red cells correspond to the detected obstacles, the black ones to the inflation radius and the blue circle to the multirotor's footprint

4.5 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 9 and 10). Consequently, the acquisition of synthetic data and the testing of image processing algorithms are feasible.

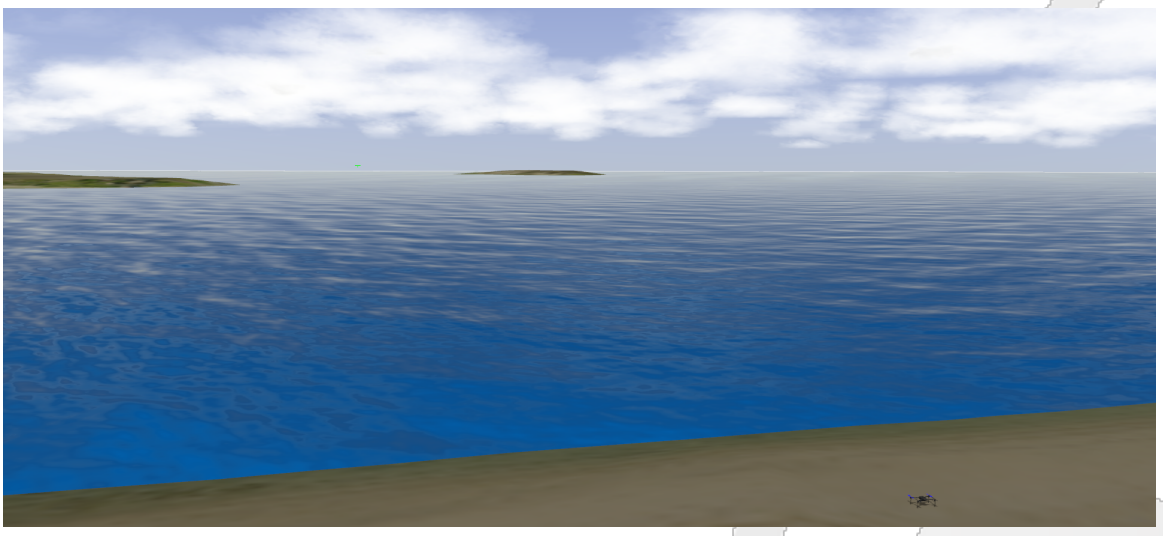

Figure 9: A Gazebo sea world

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.

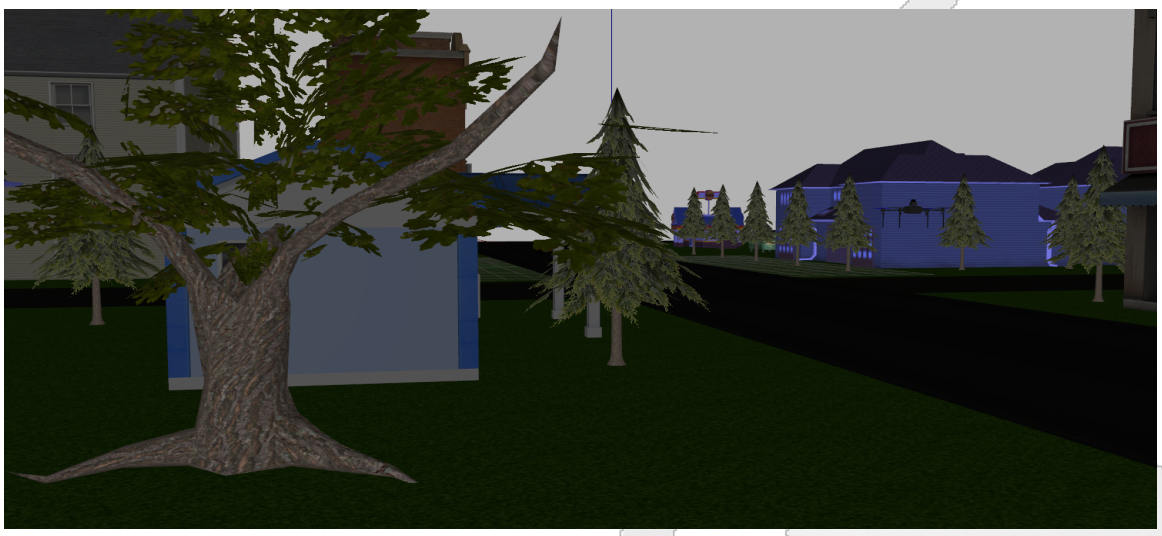

Figure 10: A Gazebo city world

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 11, 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.

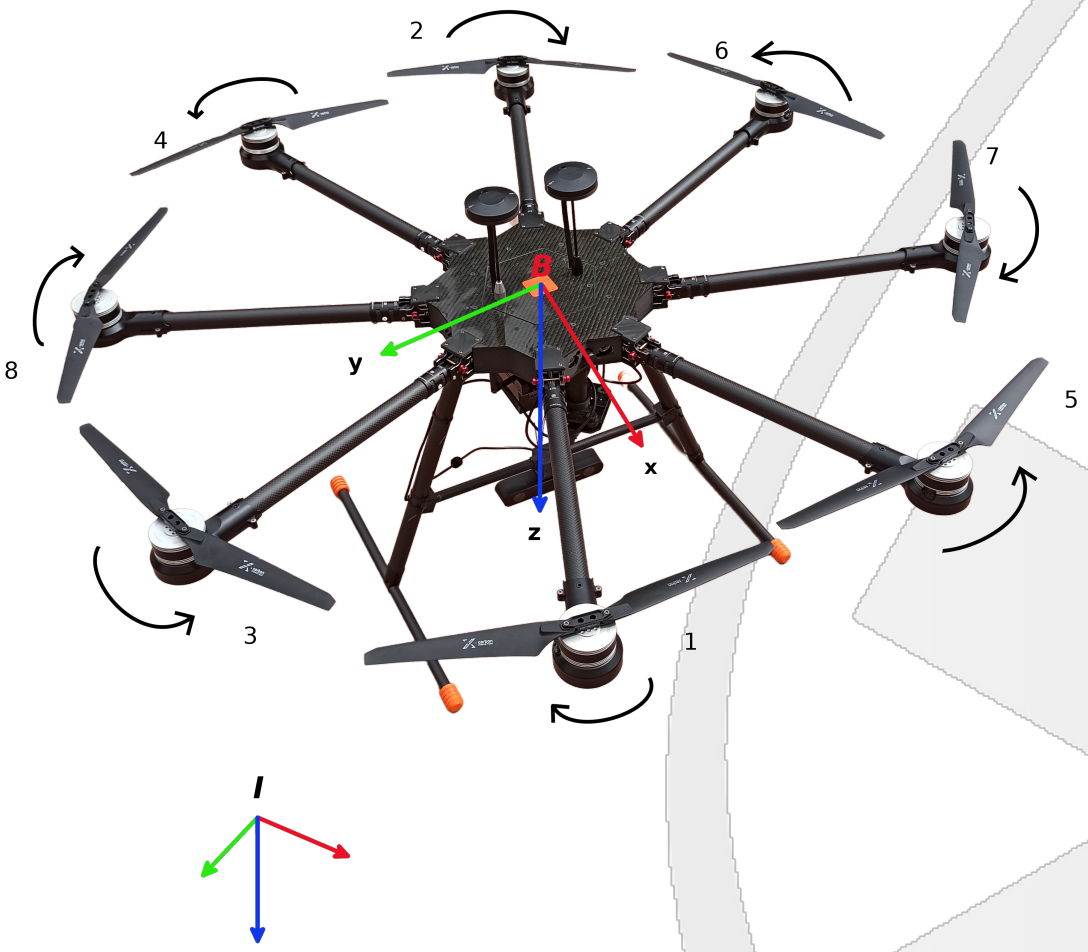

Figure 11: NTUA CSL octorotor's frame's

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 11. 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 [12], [14]:

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

$$ m{}^{I}\dot{\mathbf{v}}_B = {}^{I}\mathbf{R}_B\mathbf{F} \tag{2}$$

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

where ${}^{I}\mathbf{p}_B = \begin{bmatrix} x_B & y_B & z_B \end{bmatrix}^T$, ${}^{I}\mathbf{v}_B = \begin{bmatrix} v_{x_B} & v_{y_B} & v_{z_B} \end{bmatrix}^T$ 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 $\boldsymbol{\omega}_B$ 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 ϕ,θ,ψ respectively.

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:

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

In the specific case of the chosen 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:

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

6 PathoDRONE Planning

6.1 Global Planner

Having already presented the low-level control architecture along with the sensing capabilities of the platform, a global planner is introduced hereafter. Given a goal and a global costmap, the global planner is responsible for calculating an obstacle-free path. The ROS-based framework for the PathoDRONE allows the selection of the "move_base" package so as to provide safe navigation of the robot. The latter uses the Navfn algorithm [8] in order to navigate the robot towards the desired location. The Navfn global planner implements the Dijkstra's algorithm, a graph breadth-first-search algorithm capable of finding a minimum cost path from the starting point to a goal in the global costmap (Figure 12).

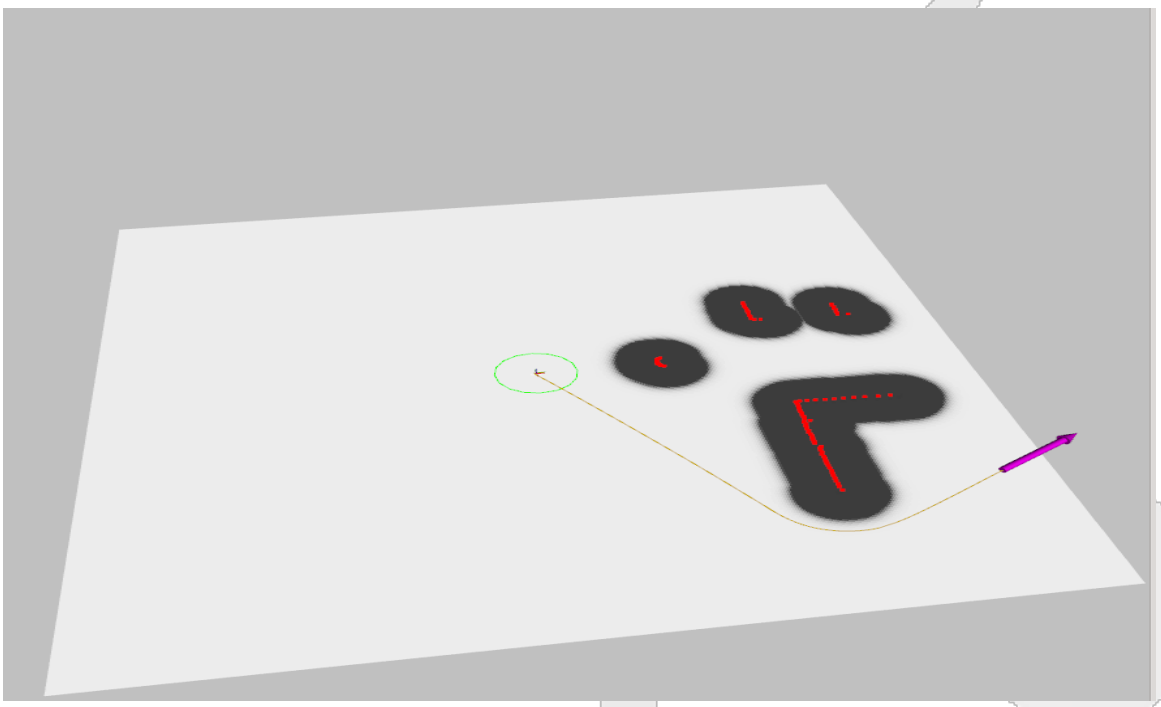

Figure 12: An example of a global path produced by the Navfn planner. The purple arrow represents the desired goal pose.

6.2 Local Planner

A local planner is required in order to produce admissible velocity commands, which assure that the robot will stay close to the computed global plan during the transition from the current position to the target one. In addition, the importance of a local planner is highlighted by the fact that, throughout the mission, dynamic or previously unseen obstacles may be observed, which were not considered in the computation of the global path. In order to avoid undesirable collisions, these obstacles are inserted into a local costmap and the local planner is responsible for recalculating the path according to this costmap. It is mentioned that the local costmap moves together with the robot (the center of the costmap coincides with the center of the robot footprint) and its size is highly dependent on the range of the on board sensor.

Among the choices provided by the ROS Navigation Stack, the Dynamic Window Approach (DWA) local planner [9] is selected. Briefly, this planner samples, at each cycle, safe velocities according to the vehicle constraints and the local costmap and, eventually, the velocity that maximizes an objective function is selected and sent to the robot. More specifically, the DWA algorithm implements the following main steps:

1. At each time instant, a number of pairs (u_i, ω_i) is sampled, where u_i is the linear velocity of the vehicle in the xy plane and ω_i is the rotational velocity about the body z axis. The search space is restricted according to the user defined bounds on velocities and accelerations and, as a result only reachable velocities are considered. Moreover, for each pair (u_i, ω_i), a trajectory is predicted by assuming that the robot moves with the above pair for a certain time interval. The resulting trajectories that cause collisions with obstacles are excluded.

2. For each predicted trajectory, a value is assigned according to an objective function which incorporates criteria such as the distance to obstacles, the proximity to the global path and the progress to the target pose.

3. The pair (u_i, ω_i), which maximizes the above objective function is selected and is sent to the vehicle's velocity controller.

4. The steps 1-3 are repeated until the robot accomplishes the desired navigation task.

7 Simulation Results

In order to validate the performance of the algorithm, rigorous simulation studies were conducted prior to carrying out real world experiments. The simulation environment is a realistic city consisting of various obstacles such as trees and buildings (Figure 13), which is what we would expect in the scenario of flush-flooding within an urban environment. The multirotor has no prior knowledge concerning the existence of these obstacles and, as a result, the local planner is a key factor for the safe execution of the mission. The vehicle is equipped with a LiDAR capable of detecting objects up to 10 meters.

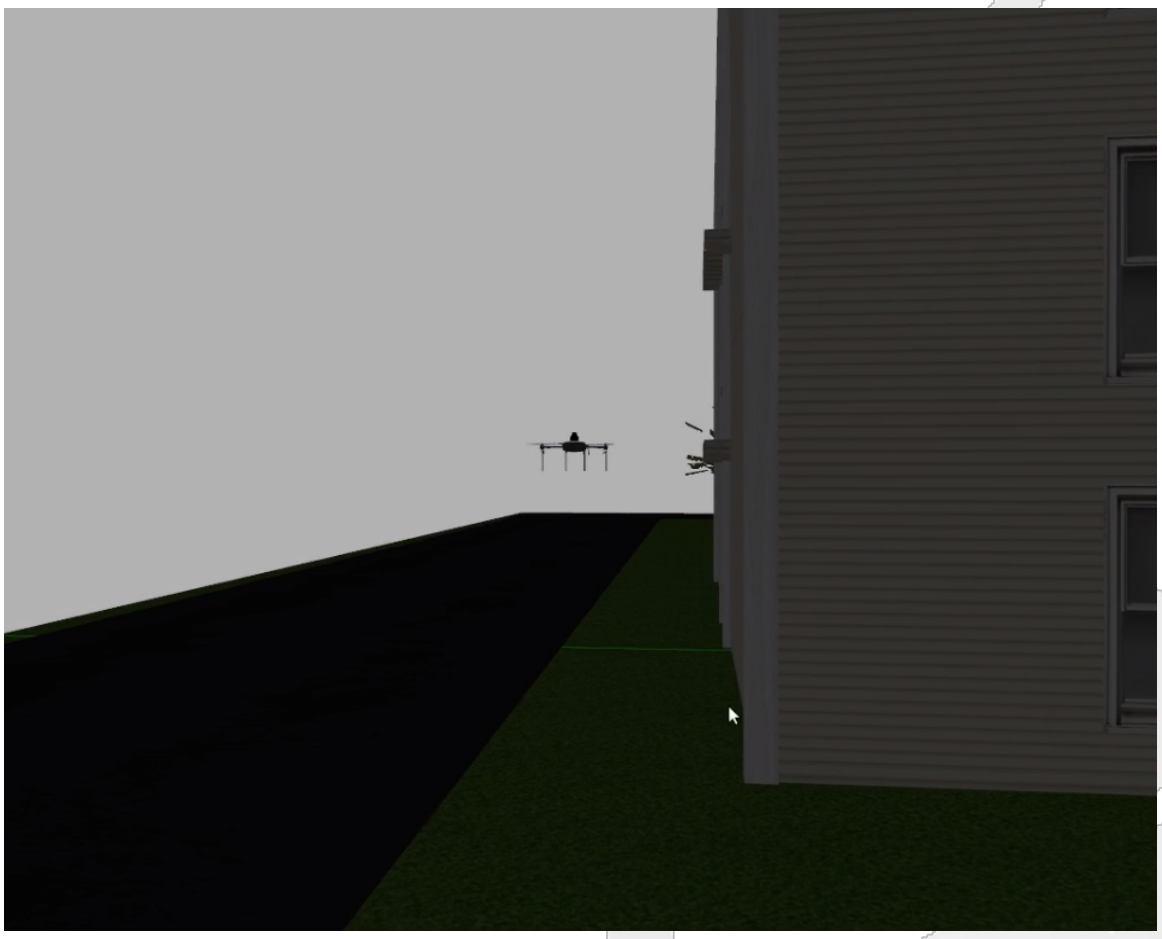

Figure 13: The simulation environment

The first step of the simulation procedure is the planning of a mission. Hence, using the MAVProxy Ground Control Station, 6 waypoints are drawn on the map (Figure 14). The commanded mission is translated into successive target goals for the move base package. By exploiting the measurements obtained by the distance sensor, a costmap is built in real-time during the simulation (Figure 15). The latter is used by the "move_base" package which

is responsible for computing feasible velocity commands that ensure the safe navigation of the multirotor towards the current desired location. Eventually, the vehicle follows the curved path of Figures 16 and 17 which simultaneously satisfy the requirements for obstacle avoidance and autonomous execution of the mission. It is recalled that the absence of a local planner would result in straight paths between the consecutive waypoints and, consequently, the collision between the vehicle and the surrounding obstacles would be unavoidable.

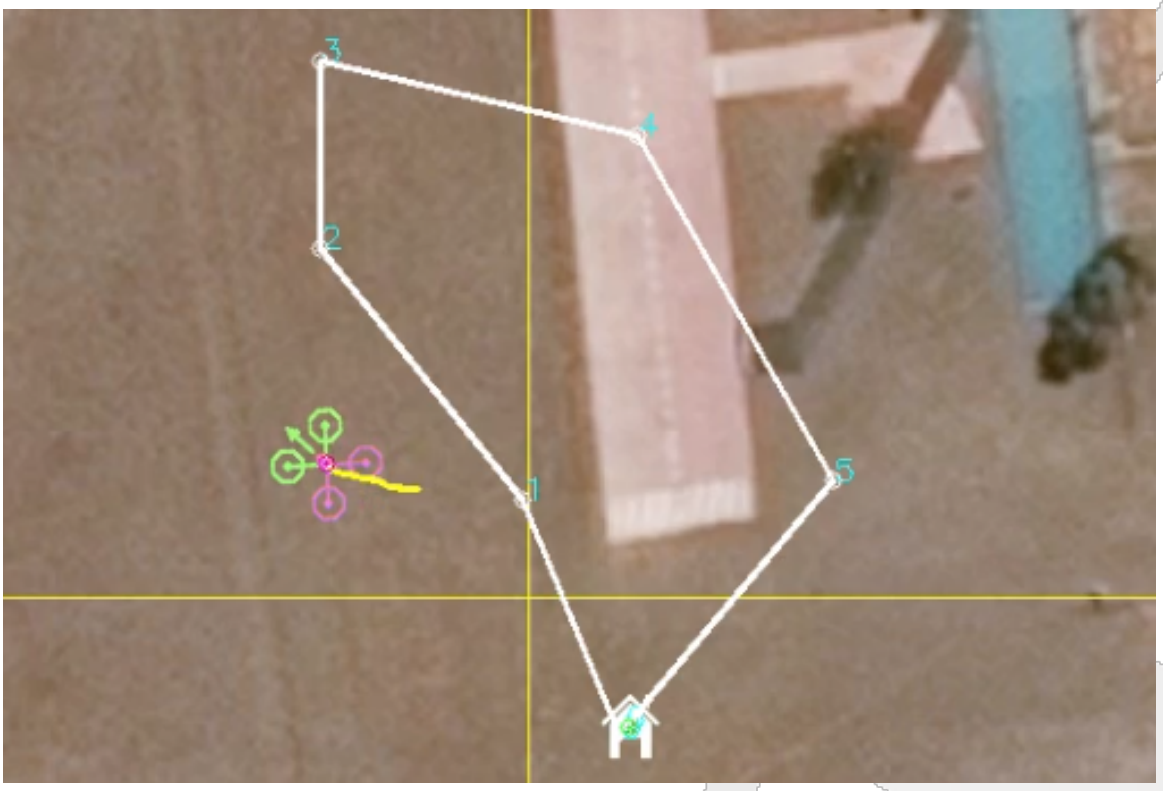

Figure 14: The target mission of the simulation study

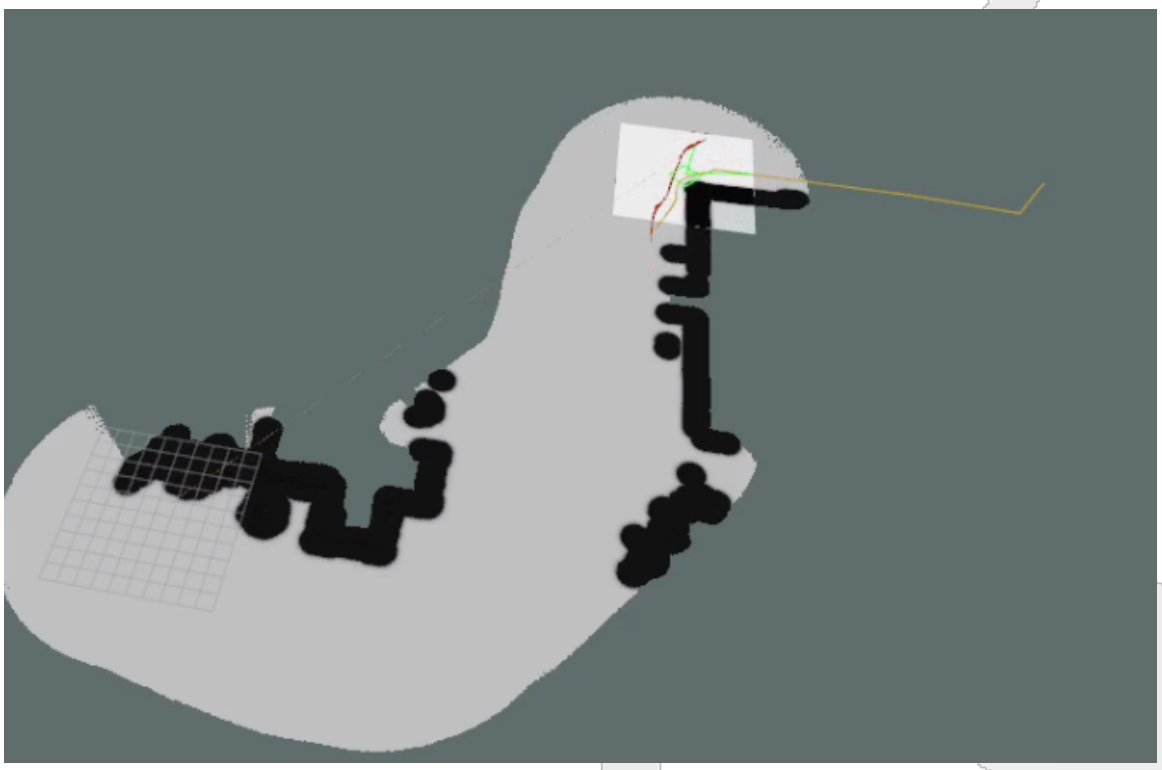

Figure 15: The costmap built during the simulation

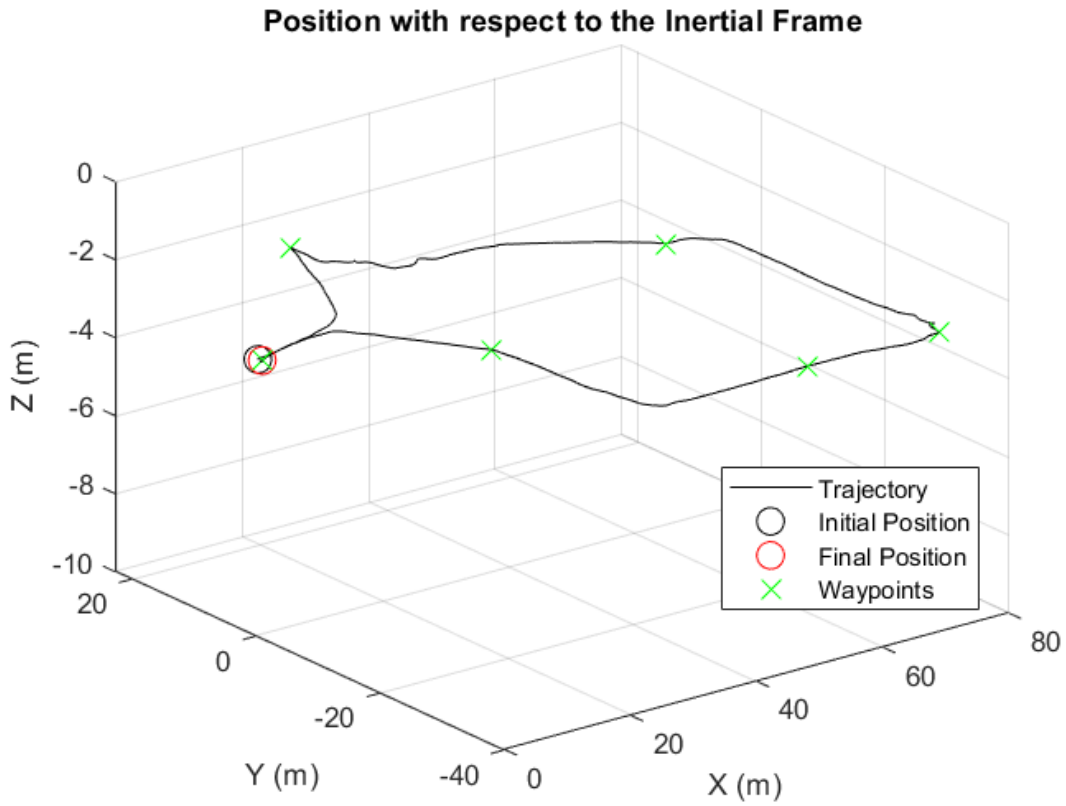

Figure 16: The 3D trajectory of the vehicle

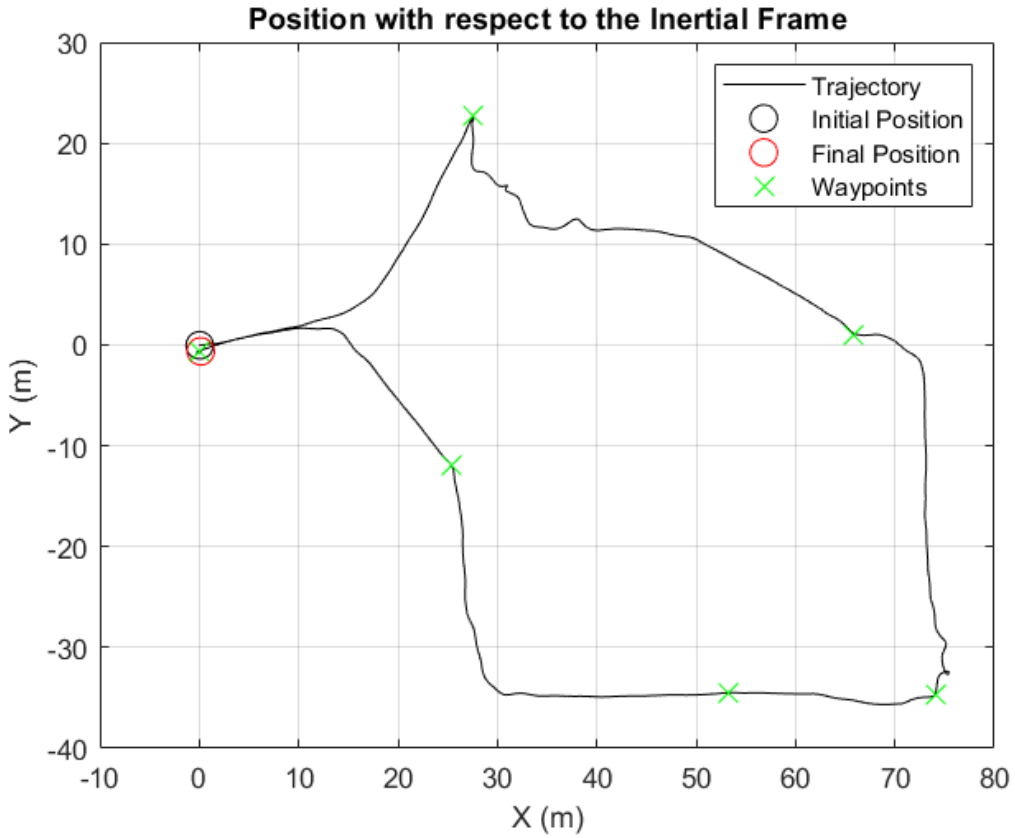

Figure 17: The 2D trajectory of the vehicle in x-y plane

8 Experimental Results

Following the validation of the proposed framework in the simulation environment, experiments were carried out, inside the NTUA campus, in order to test further the performance of the algorithm in the real world. All the experiments were conducted with the NTUA octorotor, presented in Section 4, and two portable computers. The first one is equipped with Intel(R) i5-8250U CPU @ 1.60GHz, NVIDIA GeForce MX130 GPU and 8GB of RAM running Ubuntu 18.04 and ROS Melodic[2] installed. The system specifications of the second laptop are Intel(R) i7-8565U CPU @ 1.80 GHz, Intel(R) UHD Graphics 620 and 16GB of RAM with Windows 10 Pro and Mission Planner installed. The first laptop is essential in order to communicate via the Secure Shell Protocol (SSH) with the on-board Jetson Xavier where all the necessary software programs are executed. Furthermore, the second laptop runs the Mission Planner application and, consequently, the communication with the autopilot, the monitoring of the flight status and the planning of missions are feasible.

8.1 Experiment I

In a preliminary phase, the performance of the proposed framework is tested in an environment consisting of sparse obstacles. More precisely, the first field experiment is carried out in the outdoor environment of Figure 18, where sparse trees and street lights may jeopardize the safe and autonomous navigation of the NTUA octorotor.

Figure 18: The outdoor environment of field Experiment I

A mission (Figure 19) is planned in such a manner that the straight paths between the waypoints intersect with the obstacles. Hence, during the navigation of the NTUA

[2]http://wiki.ros.org/melodic/Installation/Ubuntu

octorotor, the measurements obtained by the on-board ZED 2 stereocamera are exploited so as to detect the obstacles and construct a 2D occupancy grid (Figure 20).

Figure 19: The target mission of Experiment I

Eventually, the implementation of the suggested algorithm guides safely the octorotor to the desired locations, as depicted in Figures 21 and 22. An overview of the individual parts which comprise the framework i.e., on-board image, external camera, Mission Planner and costmaps, is demonstrated in Figure 23.

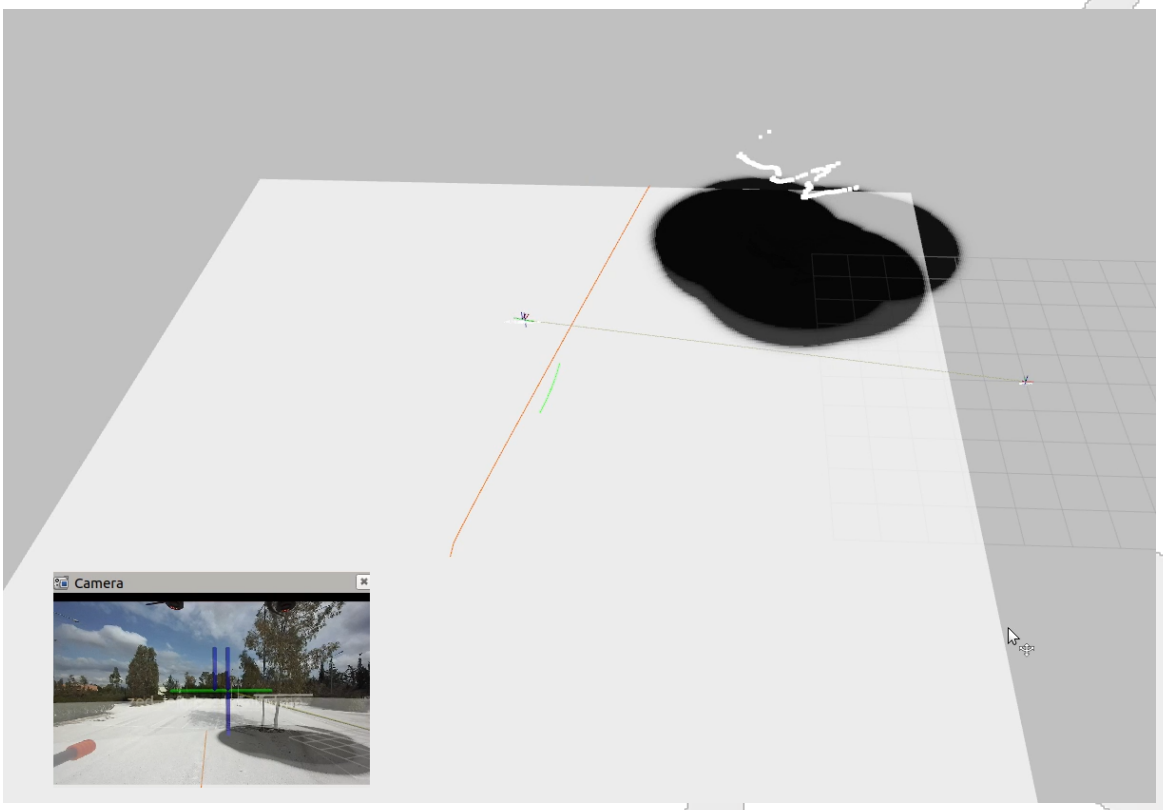

Figure 20: The local costmap, built in real time, during the Experiment I

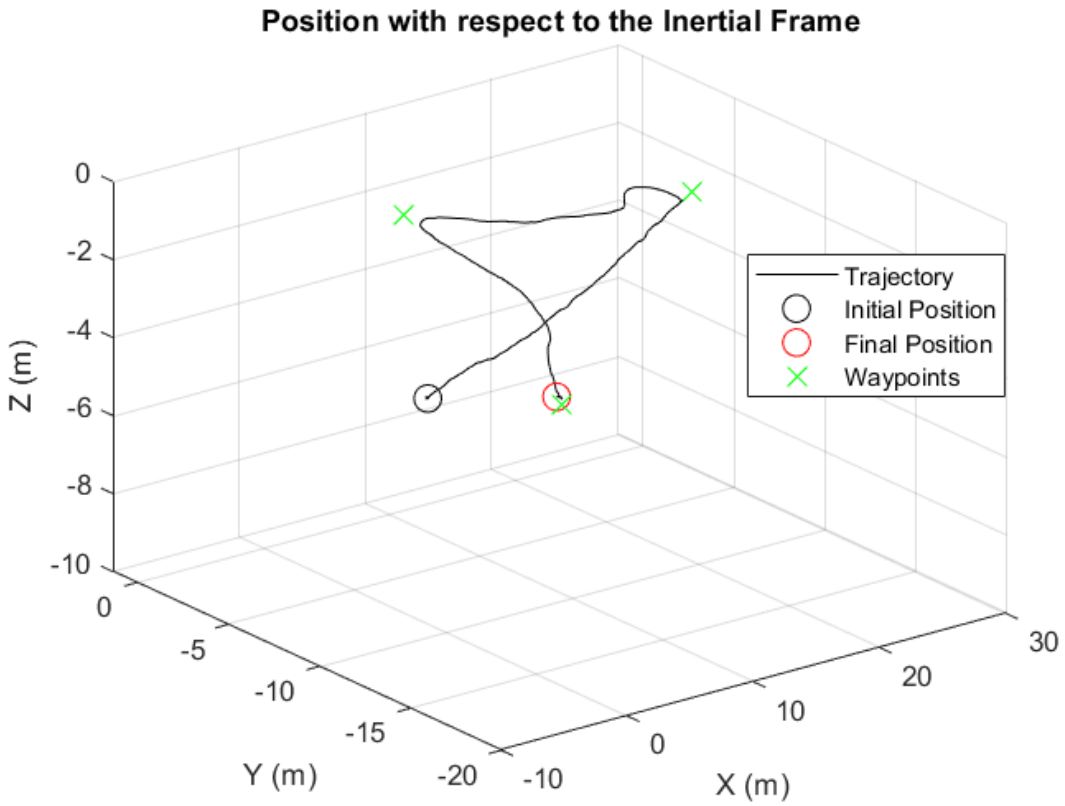

Figure 21: The 3D trajectory of the vehicle during the Experiment I

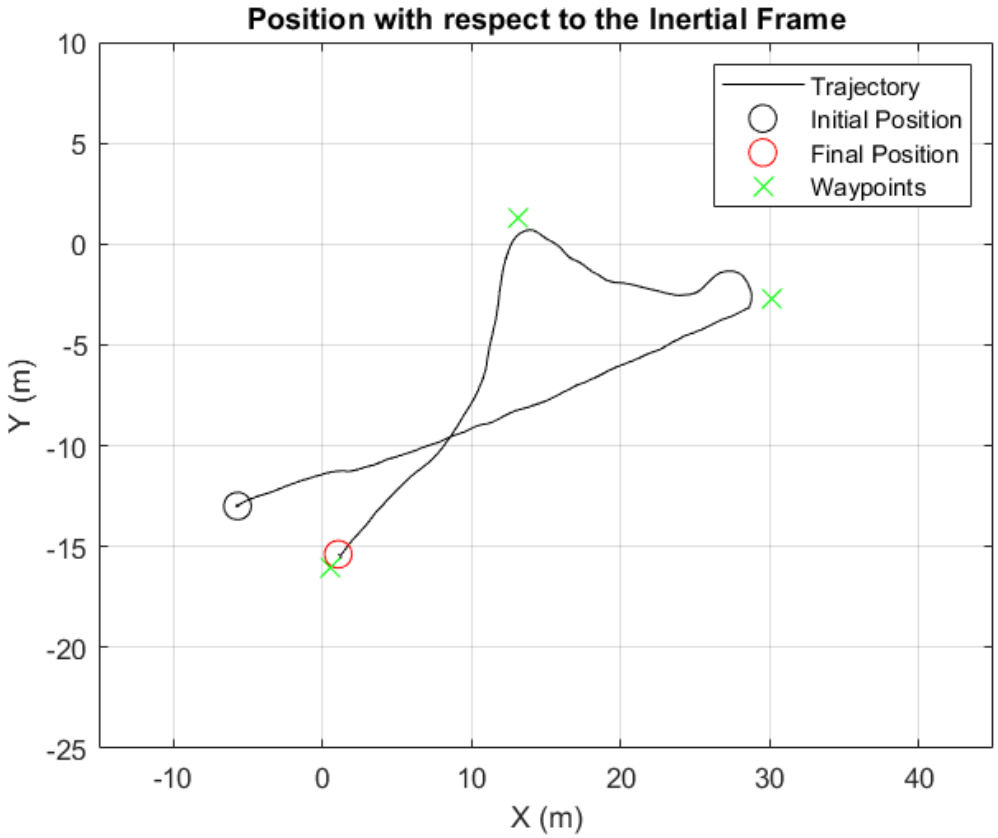

Figure 22: The 2D trajectory of the vehicle in x-y plane during the Experiment I

Figure 23: An overall view of the Experiment I

8.2 Experiment II

In order to further investigate the reliability of the suggested framework, a second experiment is conducted in a more challenging outdoor environment (Figure 24). In this case, the NTUA octorotor avoids trees in a row inside a narrow space, surrounded by buildings and trees, in order to accomplish the commanded mission of Figure 25. Due to the existence of multiple obstacles, a significant number of cells in the costmap (Figures 26 and 27) is marked as occupied during the execution of the mission and, as a consequence, the octorotor navigates in a more restricted space compared to the first field experiment.

Figure 24: The outdoor environment of field Experiment II

As illustrated by the Figures 28 and 29, notable deviations from the rectilinear paths between the successive waypoints are required in order to direct with safety the octorotor to the desired locations. The aforementioned fact verifies the challenging environment of the second experiment, where the NTUA octorotor is commanded both to avoid collisions and to navigate autonomously.

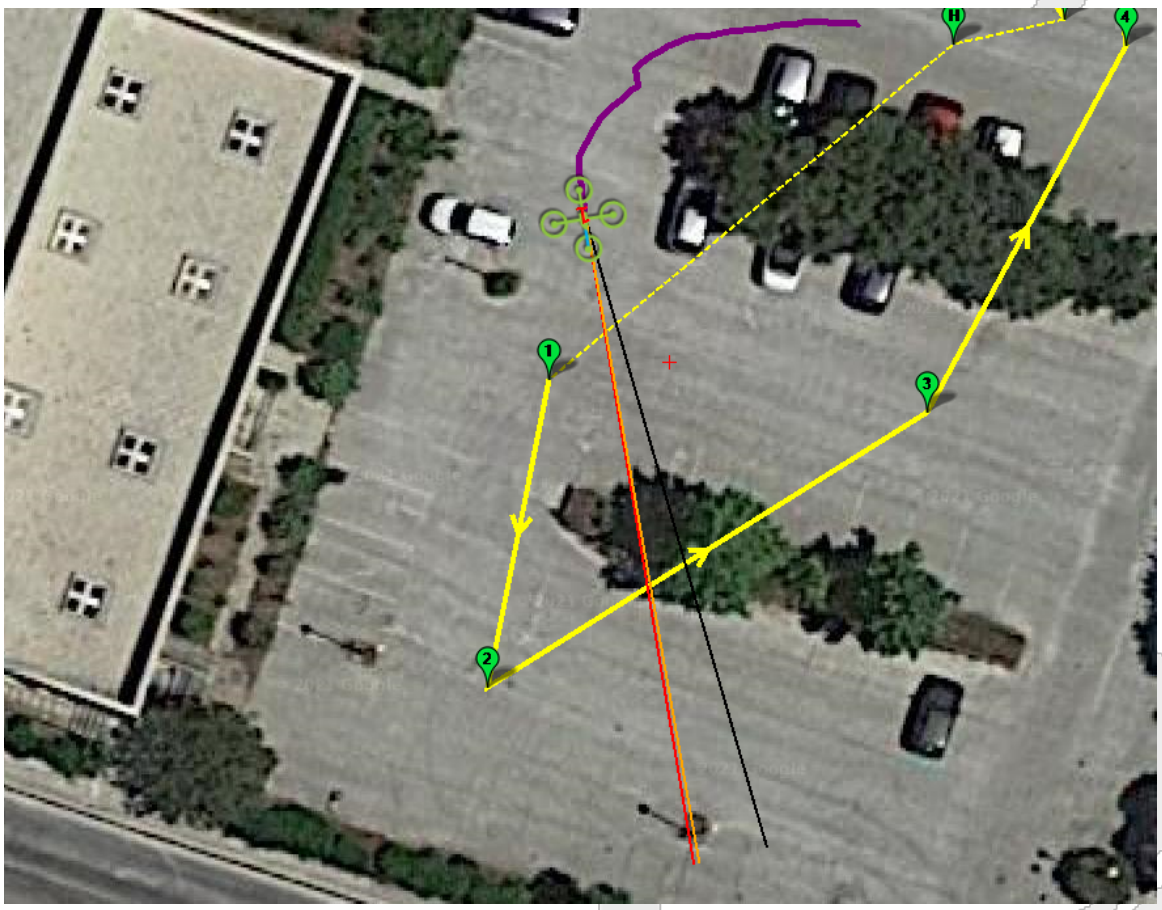

Figure 25: The target mission of Experiment II

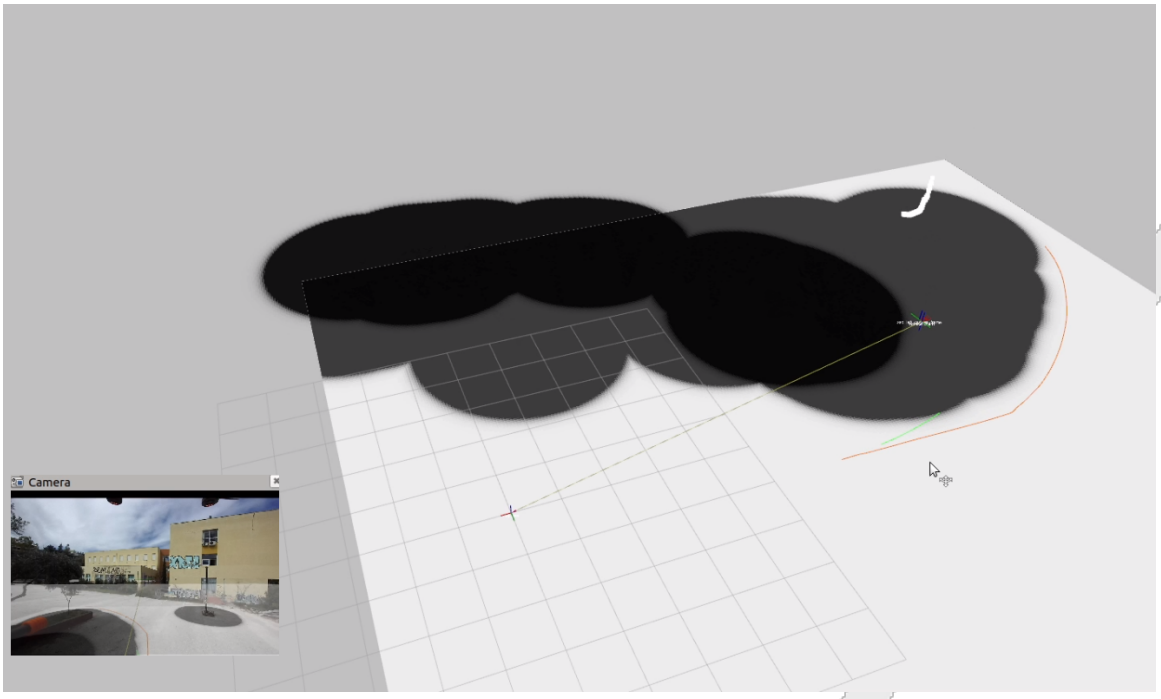

Figure 26: The local costmap, built in real time, during the Experiment II

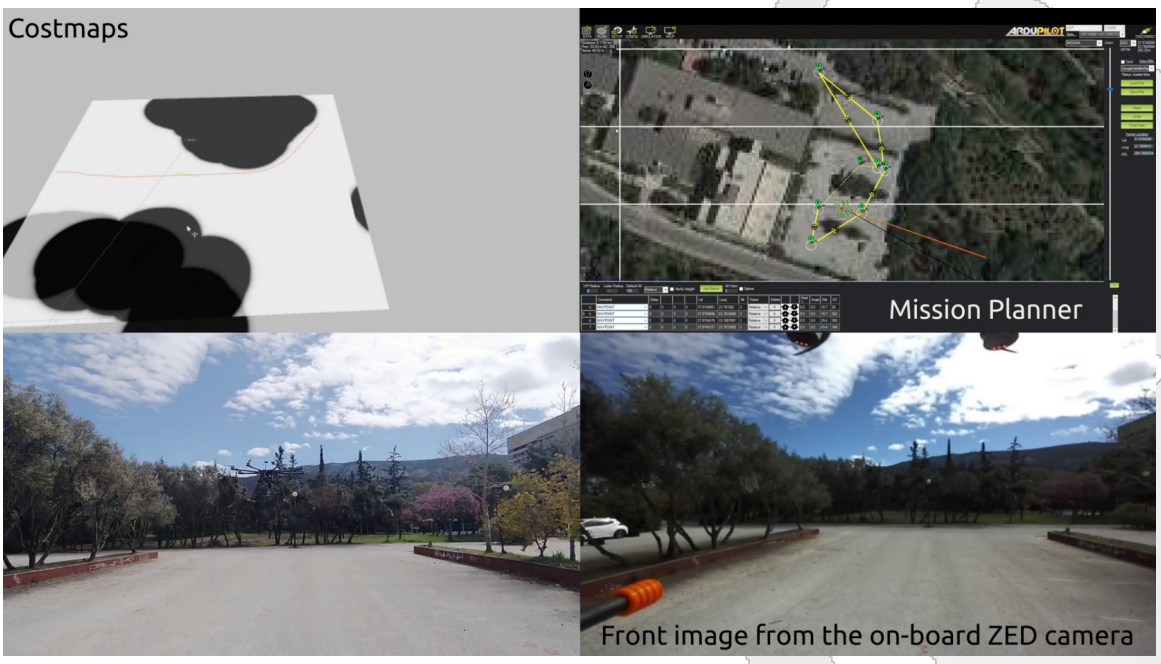

Figure 27: An overall view of the Experiment II

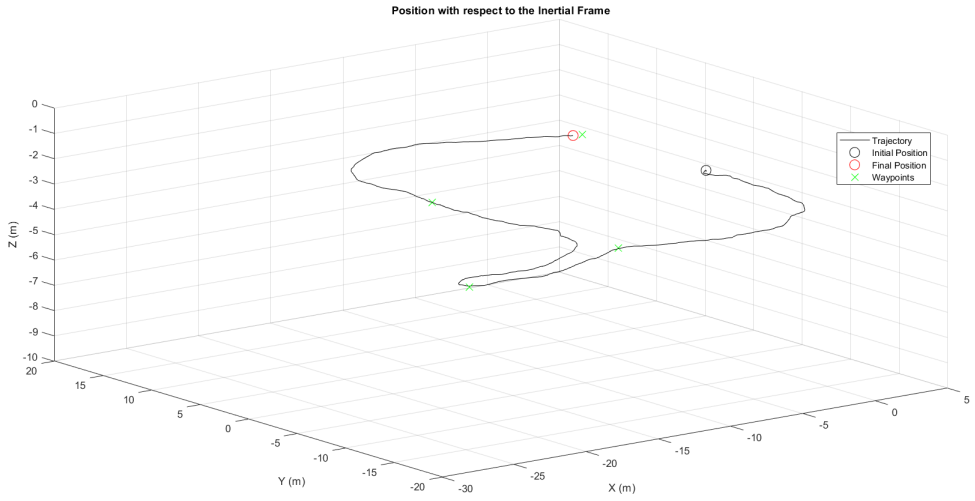

Figure 28: The 3D trajectory of the vehicle during the Experiment II

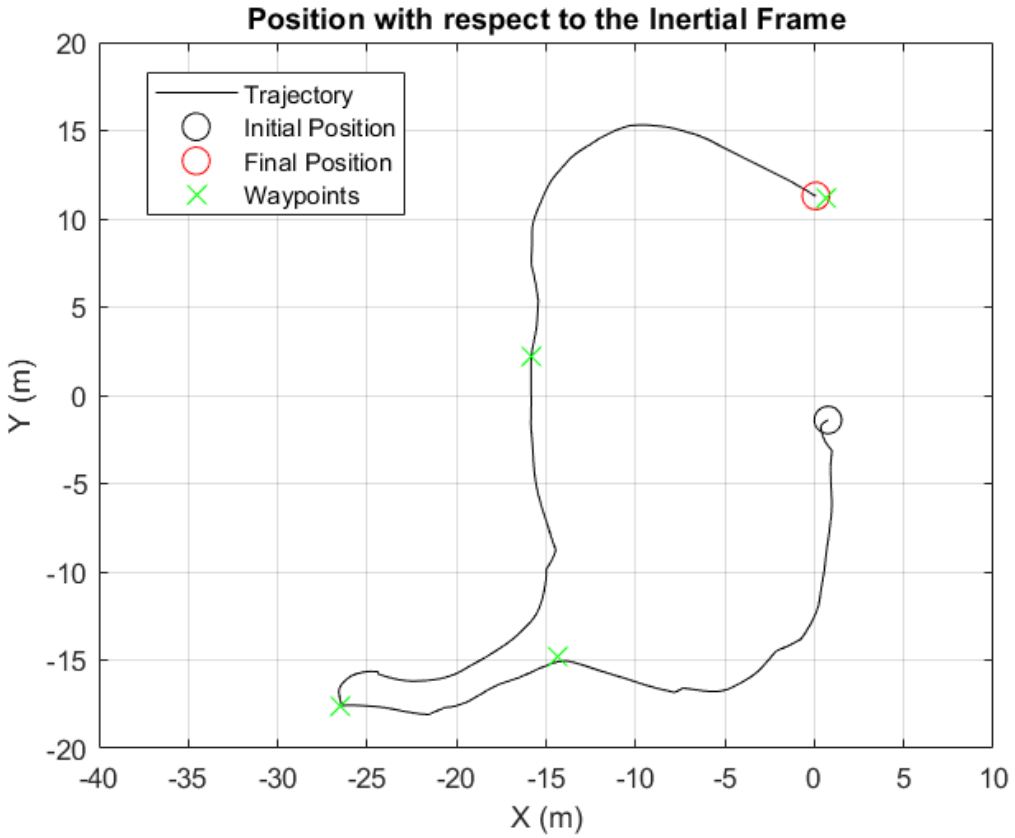

Figure 29: The 2D trajectory of the vehicle in x-y plane during the Experiment II

8.3 Experiment III

The previous experiments are carried out at low altitudes and, thus, a third experiment is performed, where the NTUA octorotor avoids a dense tree of significant height (Figure 30). Using the Mission Planner application, a mission consisting of two waypoints (Figure 31) is designed. The straight path between the waypoints intersects with the obstacle and, hence, reactive planning is required when the obstacle is detected by the on-board ZED stereocamera as depicted in Figures 32 and 33. Indeed, the proposed framework successfully computes feasible paths around the obstacle, demonstrated in Figures 34 and 35, which ensure the safe completion of the target mission.

Figure 30: The outdoor environment of field Experiment III

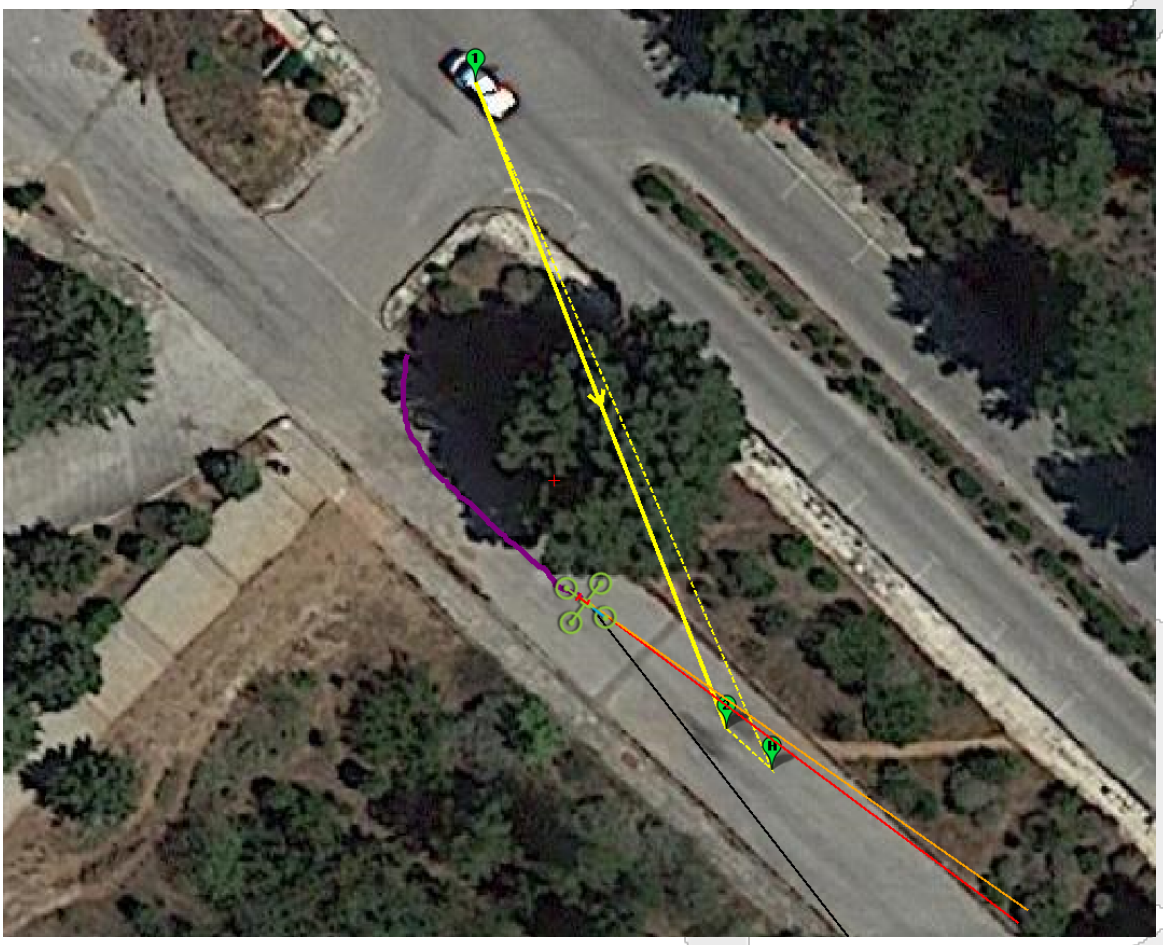

Figure 31: The target mission of Experiment III

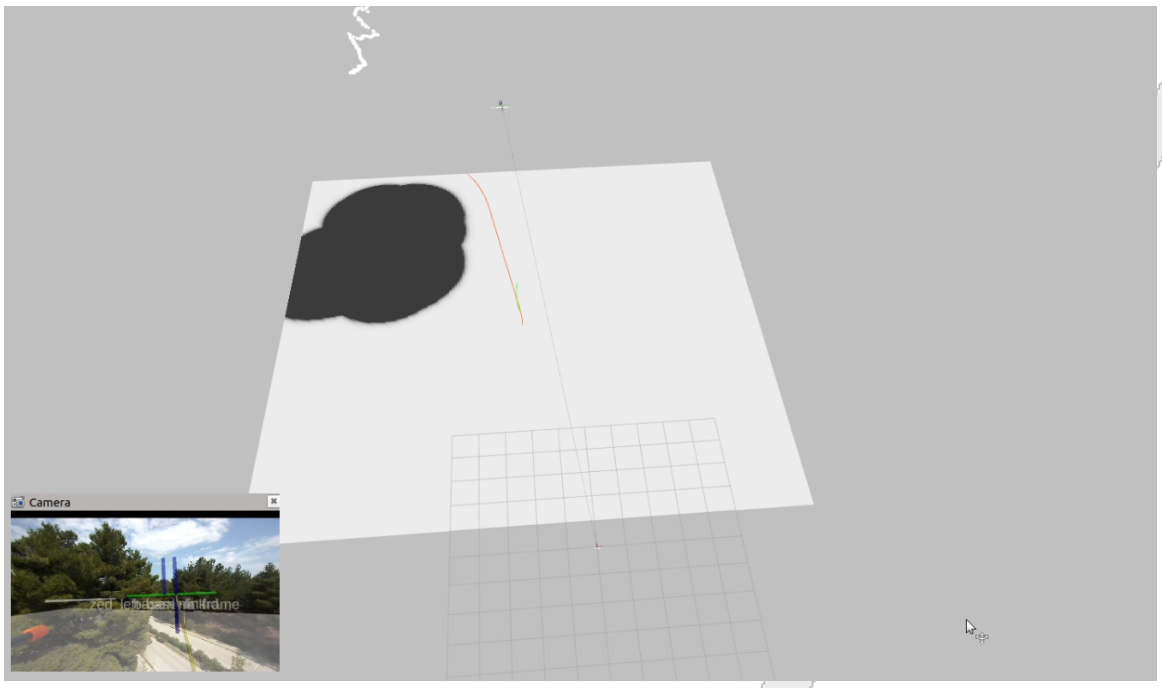

Figure 32: The local costmap, built in real time, during the Experiment III

Figure 33: An overall view of the Experiment III

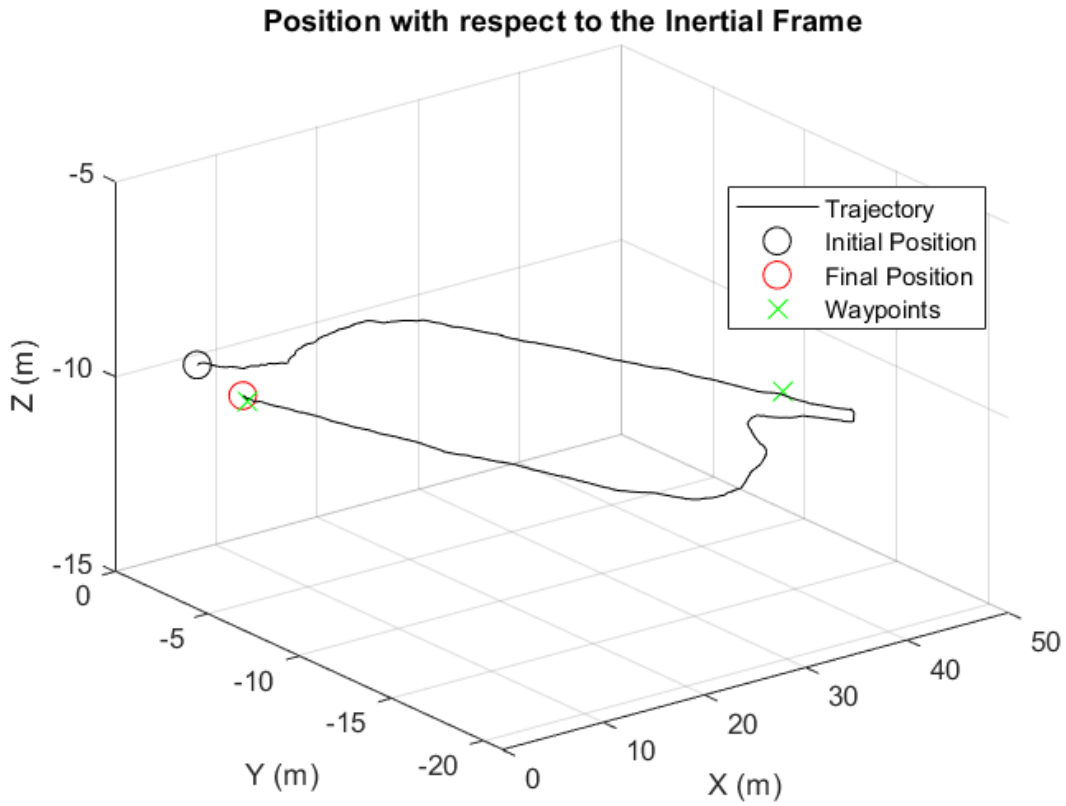

Figure 34: The 3D trajectory of the vehicle during the Experiment III

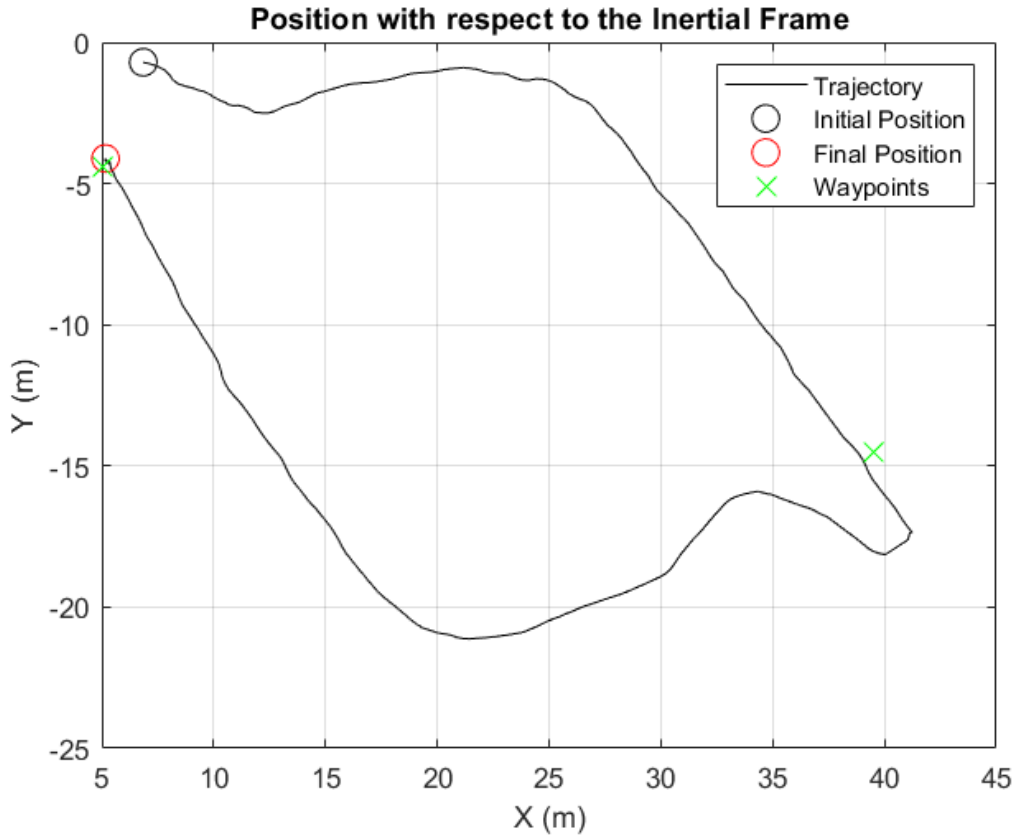

Figure 35: The 2D trajectory of the vehicle in x-y plane during the Experiment III

9 Conclusion

Having presented both the theoretical and applied background of the proposed framework, along with rigorous simulation-based and experimental results, we can reflect upon the efficacy of the suggested methods.

The adopted ROS-based structure, presents some very significant advantages. The open-source nature of the framework, along with the Ardupilot autopilot can be readily employed on any platform and is not limited to the specific hardware that was used for the experimental demonstrations presented in the text. Thus transitioning to any other platform is as simple as possible. The relevant literature and the community behind it also provides with support for any issues that might arise. Additionally, the current solution is easy to modify to suit any future needs, in contrast to a commercial solution, where any such modification would be more involved.

The motion planning schemes that are supported by the aforementioned framework, are first of all robust, having gone through extensive testing by the community and secondly are tunable and modifiable to the extent of one's familiarity with the respective literature and software development tools. Therefore, such modular solutions can be extensively adapted if necessary to fit any mission's specifications.

In this spirit, the sensing capabilities of the chosen platform can be adapted in a modular fashion, where a variety of sensors can be introduced to fit several different scenarios.

Concerning the results, both experimental and simulation-based, the above modular nature and tunability translate to very reliable real-world results. In the various presented scenarios, the platform proved effective in completing the missions, while successfully avoiding collisions with surrounding objects.

Outdoor experiments were realistically similar to the environments that a multi-rotor vehicle might encounter in water-related incidents as per the project's envisioned use cases, with the density of obstacles even being intentionally high. While the environments that any first responders might encounter are largely unknown, it is to be expected that a reasonably open-air area can be found in close proximity to a body of water, such that the experiments presented herein are but a worst-case estimate of real emergencies.

In conclusion, we have demonstrated how the proposed framework fully addresses the challenges presented by the outline of the project's specifications. Great emphasis was placed on modularity with respect to both the hardware and several aspects of the software, along with ease of tunability and robustness. The promising results of the experiments inspire great confidence to the proposed tools and demonstrate the indicative performance of the employed methods in tackling the demanding problem it sets out to address.

References

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

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

[3] Mission planner. https://ardupilot.org/planner/.

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

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

[6] Fadi Al-Turjman, Mohammad Abujubbeh, Arman Malekloo, and Leonardo Mostarda. Uavs assessment in software-defined iot networks: An overview. *Computer Communications*, 150:519–536, 2020.

[7] Matthew Ayamga, Selorm Akaba, and Albert Apotele Nyaaba. Multifaceted applicability of drones: A review. *Technological Forecasting and Social Change*, 167:120677, 2021.

[8] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 341–346 vol.1, 1999.

[9] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

[10] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 2017.

[11] Adeel Javaid. Understanding dijkstra algorithm. *SSRN Electronic Journal*, 01 2013.

[12] 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.

[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] 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.

[15] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In Itsuki Noda, Noriaki Ando, Davide Brugali, and James J. Kuffner, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[16] Sparsh Mittal. A survey on optimized implementation of deep learning models on the nvidia jetson platform. *Journal of Systems Architecture*, 97:428–442, 2019.

[17] Iram Noreen, Amna Khan, and Zulfiqar Habib. Optimal path planning using RRT* based approaches: A survey and future directions. *International Journal of Advanced Computer Science and Applications*, 7, 11 2016.

[18] Luis Henrique Oliveira Rios and Luiz Chaimowicz. A survey and classification of A* based best-first heuristic search algorithms. In Antônio Carlos da Rocha Costa, Rosa Maria Vicari, and Flavio Tonidandel, editors, *Advances in Artificial Intelligence – SBIA 2010*, pages 253–262, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[19] Abhishek Sharma, Pankhuri Vanjani, Nikhil Paliwal, Chathuranga M.Wijerathna Basnayaka, Dushantha Nalin K. Jayakody, Hwang-Cheng Wang, and P. Muthuchidambaranathan. Communication and networking technologies for uavs: A survey. *Journal of Network and Computer Applications*, 168:102739, 2020.

[20] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[21] Sofiane Zaidi, Mohammed Atiquzzaman, and Carlos T. Calafate. Internet of flying things (ioft): A survey. *Computer Communications*, 165:53–74, 2021.