UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

VIRTUAL REALITY

Drone-Assisted Smart Ambulance Systems for Emergency Response
Final Project

Author:

Ahmet Samet Kosum - s5635830

Mustafa Melih Toslak - s5431021

Natnael Berhanu Takele - s5446838

Professors:

Gianni Vercelli

Saverio Iacono

June 23, 2024

Contents

1 Introduction

In recent years, the integration of drones into various industries has accelerated, driven by advancements in technology and the demand for efficient solutions. Drones have emerged as invaluable tools in fields such as surveillance, delivery, and now, medical assistance. Research has highlighted their potential to revolutionize emergency response systems, particularly in scenarios where traditional ground transportation is hindered.

This project aims to simulate an emergency medical response scenario in the challenging terrain of Righi, located in Genoa, Italy. The scenario involves a smart ambulance equipped with a drone, designed to deliver life-saving medicines to a patient in an inaccessible area. The simulation will be developed using Unreal Engine, Cesium, Airsim, QGroundControl, PX4, and Python APIs, ROS , D-Flight to create a realistic and interactive environment for testing and validating the system's capabilities.

1.1 Scenario Description

In the simulated scenario, an ambulance is dispatched to aid an injured patient located on Righi, a hilly area with both paved roads and rugged trails. The ambulance can navigate the paved roads without issue, but upon reaching the start of the rough trail, it can no longer proceed due to the challenging terrain. This is where the onboard drone comes into play.

The drone, mounted on top of the ambulance, is equipped with life-saving medicines. Upon reaching the end of the derivable path, the drone is deployed to continue the journey. It flies over the rough terrain, navigating obstacles and following a predetermined path to reach the patient at the top of Righi. After delivering the necessary medications, the drone returns to the ambulance, which remains at the end of the paved road, ready to provide further assistance if needed.

Figure 1: Injured patient located on Righi

Figure 2: Drone ready for mission

1.2 Project Objectives

The primary objective of this project is to develop a simulated prototype of an adaptive real-time onboard path planner for the Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV) mission. The specific goals include:

- **Simulating the Environment:** Using Unreal Engine 5+ to create a detailed and realistic simulation of the Righi area, including both the drivable paths for the ambulance and the rugged trails for the drone.
- **Developing the Path Planner:** Implementing an adaptive path planning algorithm that allows the drone to navigate the challenging terrain autonomously, avoiding obstacles and ensuring timely delivery of the medicines.
- **Integrating Technologies:** Utilizing Airsim for realistic physics simulation of the drone, QGroundControl for mission planning and real-time monitoring, and PX4 for flight control. Python APIs will facilitate the integration of these components and automate various tasks within the simulation.

2 State of the Art

Unmanned Ground Vehicles (UGVs) integrated with Unmanned Aerial Vehicles (UAVs) represent a cutting-edge solution in emergency response, particularly in scenarios where rough terrain impedes ground movement. These hybrid systems combine the robust, all-terrain capabilities of UGVs with the aerial agility and rapid deployment potential of UAVs. In practice, when a UGV encounters insurmountable obstacles or rugged landscapes, an onboard UAV can be launched to continue the mission. This UAV can fly over difficult terrain to deliver crucial medical supplies, such as medications, first aid kits, or even vaccines, directly to the emergency site.[1]

Recent technological advancements have significantly enhanced the functionality and reliability of these integrated systems. Autonomous navigation and real-time communication allow both the UGV and UAV to operate with minimal human intervention, using sophisticated sensors and AI algorithms to navigate complex environments and coordinate their actions. Payload delivery mechanisms have also evolved, enabling UAVs to accurately drop supplies at specified locations or even hand them off to personnel on the ground.[2]

This synergy between UGVs and UAVs not only expands the operational range and flexibility of emergency response units but also ensures that aid reaches the affected individuals more quickly and efficiently. By overcoming the limitations of ground-based vehicles and leveraging the speed and versatility of aerial drones, these hybrid systems provide a robust, adaptable, and highly effective means of delivering humanitarian aid in disaster-stricken or hard-to-reach areas. These innovations are crucial in enhancing the resilience and responsiveness of emergency services, potentially saving lives and reducing the impact of disasters.[3]

2.1 Examples

Unmanned Ground Vehicles (UGVs) equipped with drones exemplify advanced multi-domain robotic systems capable of tackling diverse emergency scenarios. Here are a few notable examples:

- **BRINC Drones and ICOR Technology**: The integration of UAVs and UGVs for first responders has been exemplified by BRINC Drones and ICOR Technology, which have developed systems to deploy UAVs from UGVs for public safety operations. This setup allows for seamless mission execution in dangerous situations.
- **Cooperative Path Planning of UAVs and UGVs for a Persistent Surveillance Task in Urban Environments**: Cooperative path planning of UAVs and UGVs for surveillance demonstrates how these systems can work together to cover large areas efficiently, minimizing travel time and enhancing mission effectiveness.

Figure 3: BRINC Drones - ICOR Technology

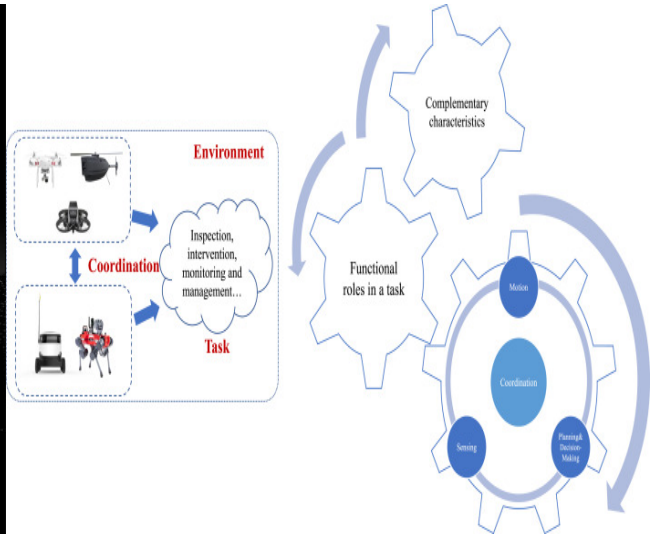

Figure 4: Cooperative Path Planning of UAVs and UGVs

3 Used Tools for Project

The development and implementation of the drone-assisted smart ambulance system leverage a variety of advanced tools and technologies to ensure robust simulation, control, and operational efficiency.

3.1 Unreal Engine 5.2.0

Unreal Engine 5 is utilized for creating a detailed and realistic simulation environment. This game engine provides high-fidelity graphics, advanced physics simulation, and extensive support for virtual reality (VR) and mixed reality (MR) applications. It allows for the creation of immersive scenarios that closely mimic real-world conditions, facilitating the thorough testing and validation of the system's capabilities.

Figure 5: Unreal Engine 5.2

3.2 Cesium v2.5 with Google Maps API

The accurate representation of the Genova map within the simulation is achieved using Google API and Cesium. Google API provides real-time data and mapping services, while Cesium offers detailed 3D geospatial visualization.[4] This combination ensures that the simulated environment is both precise and dynamically updated, reflecting real-world geographic and infrastructural details.

Figure 6: Cesium v2.5

Figure 7: Google Maps API

3.3 Airsim(Colosseum)

Airsim, developed by Microsoft, is an open-source simulator for drones and ground vehicles. It is used to simulate the physical and sensor dynamics of the drone for the project. Colosseum is forked from Airsim, a simulator for robotic, autonomous systems, built on Unreal Engine5+. It is open-source, cross platform, and supports software-in-the-loop simulation with popular flight controllers such as PX4, ArduPilot and hardware-in-loop with PX4 for physically and visually realistic simulations.[5]

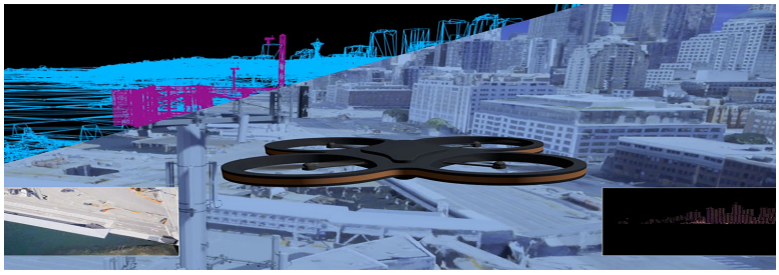

Figure 8: Airsim (Colesseum)

3.4 MAVLink

MAVLink, short for Micro Air Vehicle Link, is a lightweight communication protocol designed for exchanging information between unmanned aerial vehicles (UAVs) and ground control stations (GCS), as well as other components like onboard computers. It is widely used in the drone industry for its efficiency and reliability in transmitting data.[6]

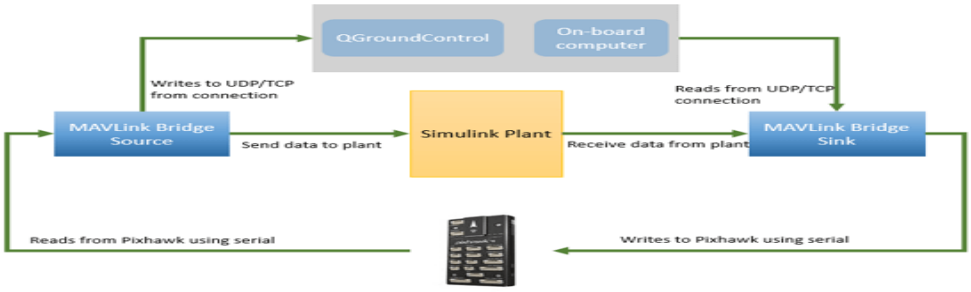

Figure 9: MAVLink

3.5 PX4 and QGroundControl

The PX4 open-source flight control software is employed for managing the drone's flight operations. It provides advanced flight modes and robust control algorithms. QGroundControl acts as the ground control station software, offering a user-friendly interface for mission planning, real-time monitoring, and manual control of the drone. This setup ensures precise and reliable drone operations in the simulated environment.[7]

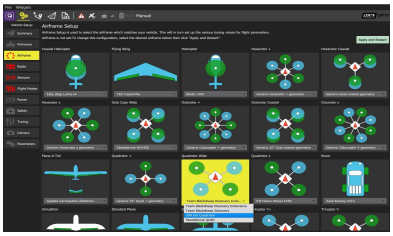

Figure 10: PX4

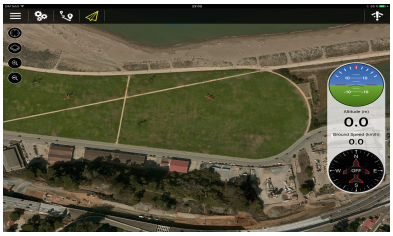

Figure 11: QGroundControl

3.6 Spline

The ambulance navigation within the simulation is managed using Spline technology, which allows for smooth and accurate path following along predefined routes. This ensures that the ambulance can effectively navigate through the drivable parts of the terrain before deploying the drone for the final leg of the journey.

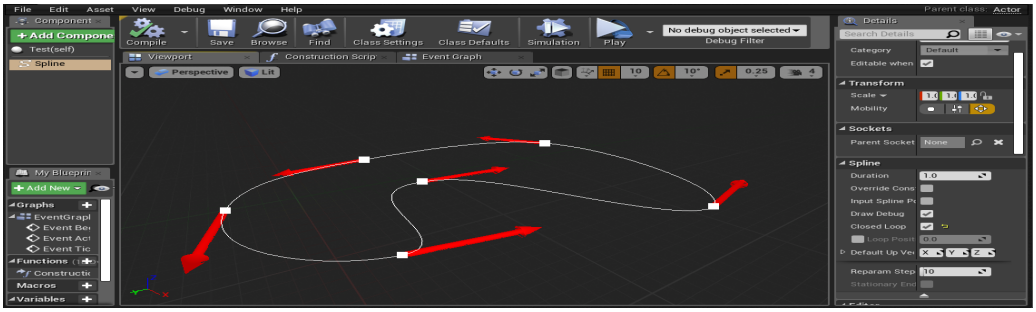

Figure 12: Spline

3.7 D-Flight

D-Flight is a platform that provides essential data and services for drone operations within regulated airspaces. It ensures compliance with aviation regulations by offering flight plan validation, real-time airspace information, and no-fly zone enforcement. This integration is critical for ensuring that drone operations are safe and legally compliant both in the simulation and in potential real-world applications. [8]

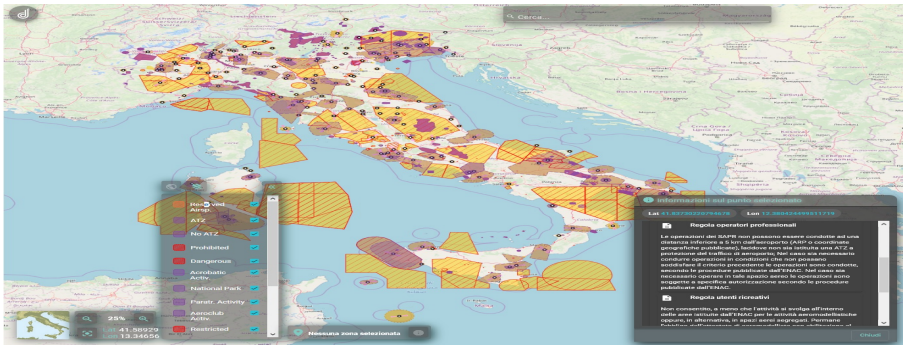

Figure 13: D-Flight

3.8 Sensor Data Processing: ROS-Based System

A ROS-based (Robot Operating System) framework is used for sensor data processing and integration. This system facilitates the collection, processing, and utilization of real-time data from various sensors on the drone. It supports the autonomous decision-making processes and enhances the overall system responsiveness and accuracy.

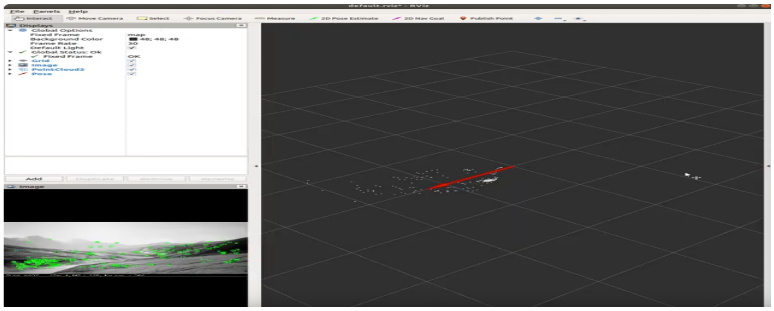

Figure 14: Data visualization on ROS (Noetic)

4 How to Create the Genoa Map?

To begin, we need to establish an environment for our project. The objective is to recreate the Genova map to test and simulate the drone in a virtual setting where it will operate. We utilize **Cesium** for its geospatial 3D mapping capabilities, which allow for the visualization of high-resolution, dynamic, real-world geospatial data in a web browser. **Cesium** provides a comprehensive set of APIs and tools for creating interactive and immersive geospatial applications.

First, we create an account on the **Cesium** platform online. Next, we download **Cesium** from the Epic Games library and integrate it into our **Unreal Engine** project. After adding the **Cesium** plugin to the list of game plugins (restarting **Unreal** each time a plugin is added), we connect our account to the **Cesium** plugin and create a new empty level to utilize the ”Georeference” panel of **Cesium**. Using the ”Tileset” module and **Google Maps API**, we provide high-resolution data for the entire world.

However, our project focuses on the Genova map, so we need to utilize the specific latitude and longitude coordinates of Genova for accurate navigation.

Figure 15: Genoa Map 1

Figure 16: Genoa Map 2

5 How to Control UAV (Drone)?

We utilized **Airsim** to simulate the drone's behaviors and interactions within the environment. To integrate Airsim with Unreal Engine, we added the compiled Airsim plugin to our Unreal Engine project by copying the plugin files into the appropriate directories within the project. Subsequently, we modified the configuration files in the project to ensure proper integration.

Within Unreal Engine, we established a new level and incorporated the Airsim plugin. We then configured the drone's sensor properties and ensured its correct positioning in the virtual Genoa environment by modifying the **settings.json** file. This setup enables us to visualize and test the drone in various scenarios within the simulation.

5.1 Manual Control of the UAV (Drone)

To evaluate the drone's behavior, we initially utilized the **Python API** provided by Airsim. This API allows seamless switching between manual and autonomous control of the drone. If any issues arise during the autonomous navigation, we can promptly switch to manual mode to ensure safe operation.

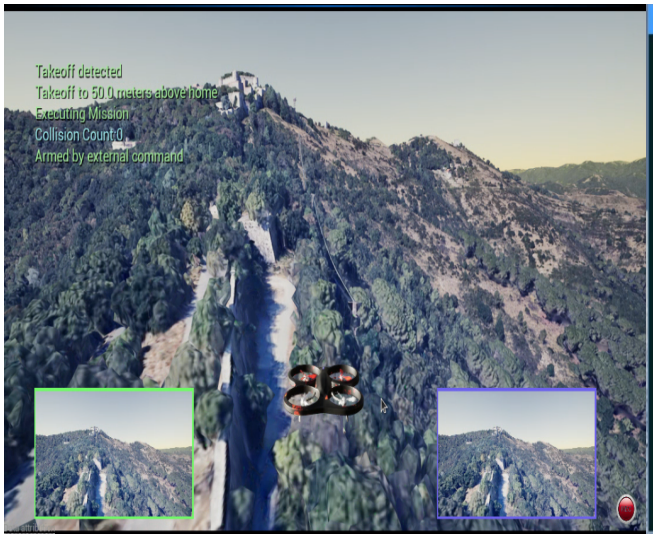

Figure 17: Manual Control

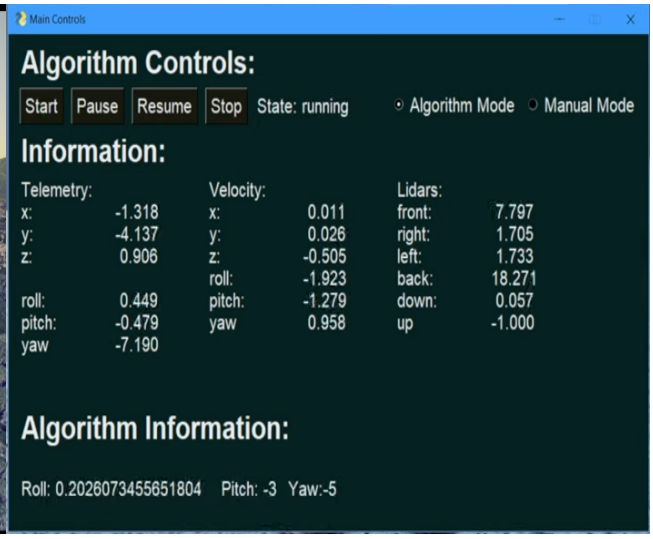

Figure 18: Airsim Python API

5.2 Autonomous Control of the UAV (Drone)

The UAV, or drone, is controlled using a combination of **PX4** flight control software and **QGroundControl**. **PX4** manages the drone's flight operations, providing advanced flight modes and robust control algorithms necessary for precise and reliable drone control. **QGroundControl** functions as the ground control station software, providing mission planning and real-time monitoring capabilities. This enables operators to efficiently plan and manage drone missions. However, QGroundControl lacks obstacle avoidance functionality. To address this limitation, we combined QGroundControl with Bug2 algorithm to handle such situations.

To detect objects in the environment, we did not use collision boxes. Instead, we employed four different distance sensors positioned at 0 degrees, 90 degrees, 180 degrees, and 360 degrees. We used a preferred direction to compute the Bug2 algorithm, which is implemented by selecting the optimal direction that avoids collisions within the preferred path.

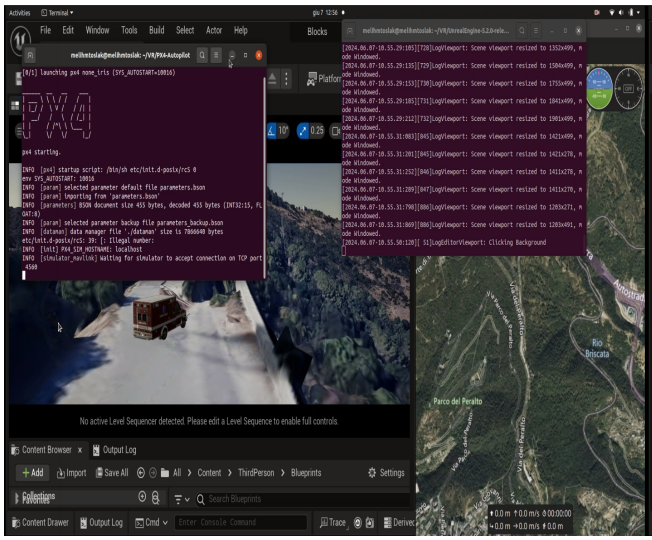

Figure 19: MAVLink-PX4

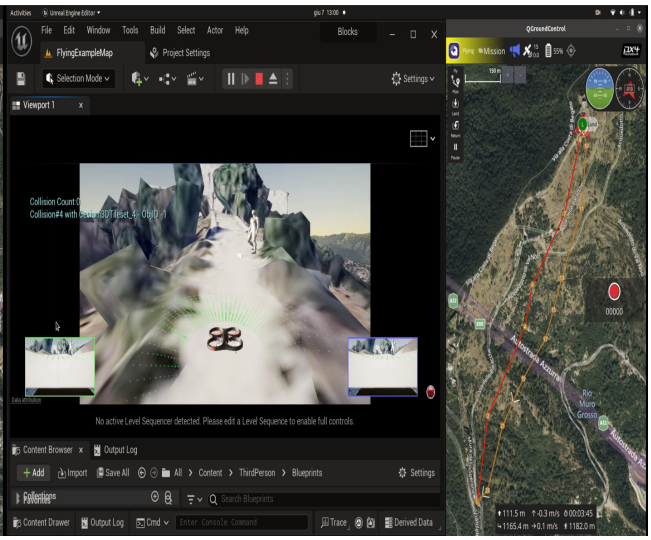

Figure 20: PX4-Qground control

The **D-Flight** website helps us understand the zones in Genoa where drone flights are permitted. By incorporating this information, we created a simple ROS node that alerts the user if the drone enters restricted zones or flies outside the allowed altitude range. This ensures that the drone operates within legal flight zones, adhering to state regulations. The **D-Flight** map uses different colors to indicate various flight restrictions, helping users understand where and how high they can fly the drone.

We utilized D-Flight as an information source to identify permitted and restricted flight zones within Genoa. Future improvements could involve integrating real-time updates from D-Flight, enhancing the ROS node to deliver more comprehensive alerts, and incorporating machine learning algorithms to predict potential violations and recommend safer flight paths.

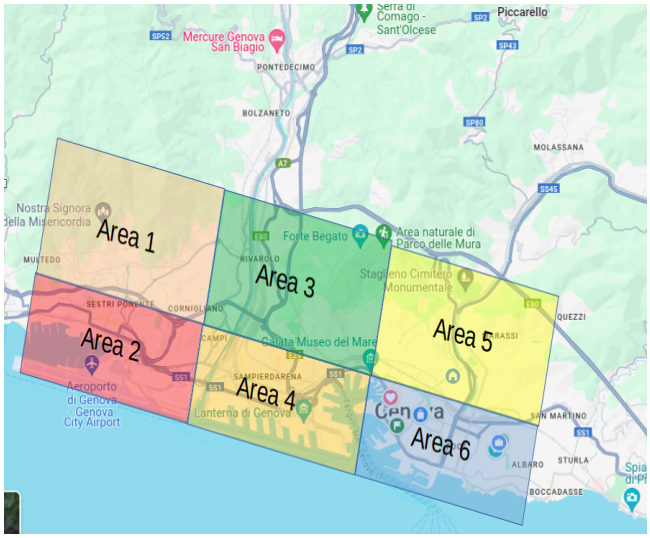

Figure 21: D-Flight Zones

```python
def gps_callback(msg):
    latitude = msg.latitude
    longitude = msg.longitude
    altitude = msg.altitude

    # Check if the drone is within the specified area
    if (AREA_1_LATITUDE_MIN <= latitude <= AREA_1_LATITUDE_MAX and
        AREA_1_LONGITUDE_MIN <= longitude <= AREA_1_LONGITUDE_MAX):
        # Check if the altitude is below the restricted minimum
        if altitude < AREA_1_MIN_ALTITUDE:
            rospy.logwarn("Please increase the altitude of the vehicle, restricted minimum altitude in this area is 100 meters")
        else : rospy.logwarn(" restricted minimum altitude in this area is 100 meters")

        # Check if the drone is within the specified area
    if (AREA_2_LATITUDE_MIN <= latitude <= AREA_2_LATITUDE_MAX and
        AREA_2_LONGITUDE_MIN <= longitude <= AREA_2_LONGITUDE_MAX):
        rospy.logwarn("Please leave the area immediately")

        # Check if the drone is within the specified area
    if (AREA_3_LATITUDE_MIN <= latitude <= AREA_3_LATITUDE_MAX and
        AREA_3_LONGITUDE_MIN <= longitude <= AREA_3_LONGITUDE_MAX):
        # Check if the altitude is below the restricted minimum
        if altitude < AREA_3_MIN_ALTITUDE:
            rospy.logwarn("Please increase the altitude of the vehicle, restricted minimum altitude in this area is 50 meters")
        else : rospy.logwarn(" Safe to fly, restricted minimum altitude in this area is 50 meters")


        # Check if the drone is within the specified area
    if (AREA_4_LATITUDE_MIN <= latitude <= AREA_4_LATITUDE_MAX and
        AREA_4_LONGITUDE_MIN <= longitude <= AREA_4_LONGITUDE_MAX):
        # Check if the altitude is below the restricted minimum
        if altitude < AREA_4_MIN_ALTITUDE:
            rospy.logwarn("Please increase the altitude of the vehicle, restricted minimum altitude in this area is 150 meters")
        else : rospy.logwarn("Safe to fly, restricted minimum altitude in this area is 150 meters")

```

Figure 22: Information provided from d-flight node

6 How to Control UGV (Ambulance)?

The UGV, or ambulance, plays a crucial role in our emergency response simulation. Here's how we integrate and control it:

6.1 Integrating the Ambulance into the Simulation

We sourced the ambulance model from **Sketchfab**. This involved downloading a suitable 3D model that fits our simulation requirements.After downloading the model, we imported it into **Unreal Engine**. This involves adding the model as a static mesh and configuring its properties to fit within the virtual Genoa map.

Within **Unreal Engine**, we place the ambulance model in the desired starting location within the Genova map. We ensure that the ambulance is properly scaled and oriented to match the environment.

6.2 Controlling the Ambulance

The ambulance's navigation is managed using **Spline** technology within the simulation. **Spline** allows for smooth and accurate path following along predefined routes. This ensures the ambulance can efficiently navigate through the drivable parts of the terrain to reach the deployment points for the drone.

To create the movement of the ambulance, we use **blueprints** within **Unreal Engine**. We define a spline for the path that the ambulance will follow and create an actor blueprint that includes actions like timeline, lerp, orientation, location, and transforms to make the ambulance follow the spline accurately.

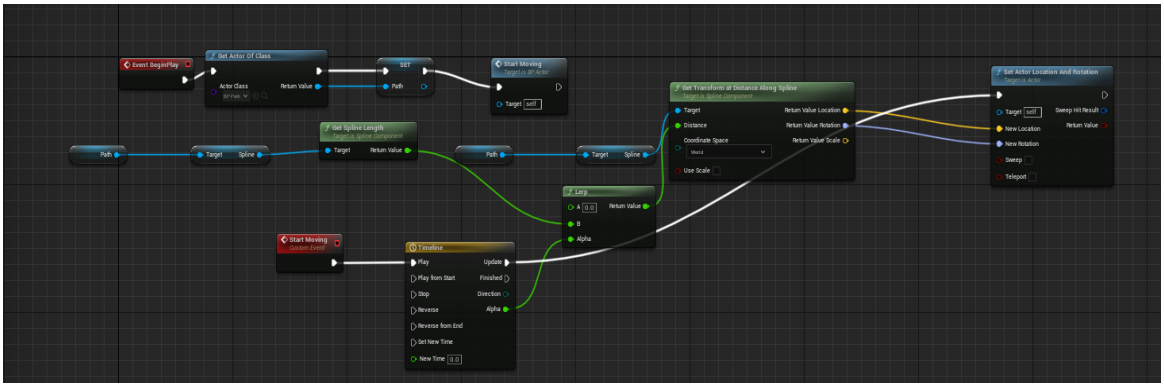

Figure 23: Spline path follower actor BP

By integrating these technologies, we ensure that the ambulance operates effectively within the simulation, capable of navigating complex terrains and coordinating with the drone during emergency response missions.

Figure 24: Spline path for ambulance

7 Conclusion

This project shows how combining smart ambulances and drones can improve emergency medical response, especially in tough terrains where regular vehicles can't go. By using advanced simulation tools, we created a realistic and interactive environment to test and prove that this system works effectively.

In the future, we can make this system even better by adding more advanced AI and machine learning techniques. This will help the drone navigate and make decisions in more complex situations. We can also test the system in different types of terrains, weather conditions, and emergency scenarios to ensure it works well in various conditions. The next step is to move from simulation to real-world testing in controlled environments to see how the system performs in real-life situations. Additionally, exploring how to manage multiple UAVs and UGVs working together in larger operations could lead to new possibilities for widespread use in both urban and rural areas.

By continuing to develop these technologies, we can greatly improve the efficiency and effectiveness of emergency medical services, potentially saving lives and helping people in disaster-stricken or hard-to-reach areas. This project not only highlights the technical possibilities but also shows the importance of working together across different fields to advance healthcare and emergency response systems.

References

[1] Robotics Meta, *Robotics for Disaster Response and Search Operations*. `https://roboticsmeta.com`

[2] Robotics Centre, *Disaster Response and Aid*. `https://robotics-centre.com`

[3] Unmanned Systems Technology, *Robotic Systems: Aerial, Ground, Underwater & Amphibious Robots*. `https://www.unmannedsystemstechnology.com`

[4] Cesium, *Cesium for Unreal*. `https://cesium.com/learn/unreal/`

[5] Microsoft, *AirSim*. `https://microsoft.github.io/AirSim/`

[6] MAVLink, *MAVLink*. `https://mavlink.io/en/about/implementations.html`

[7] Qgroundcontrol, *PX4-QGroundcontrol*. `https://docs.qgroundcontrol.com/Stable_V4.3/en/qgc-user-guide/fly_view/fly_view.html`

[8] D-flihgt, *D-Flight*. `https://www.d-flight.it/new_portal/`