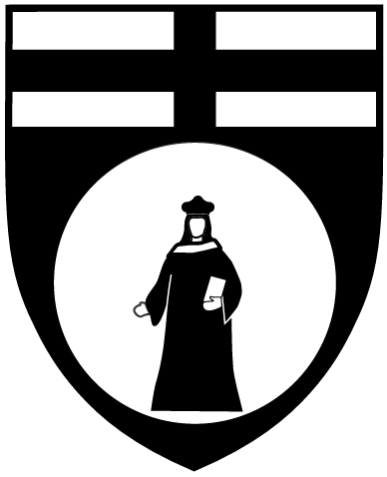

Dibris Dipartimento di Informatica, Bioingegneria,
Robotica e Ingegneria dei Sistemi

Virtual Reality for Roboics

Report

Urban Aerial Drug Delivery: Revolutionizing Healthcare with Quadrotor Technology

---

Advisor: Gianni Viardo Vercelli

UNIVERSITY OF GENOA, JULY 2024

Member list & Workload

No.	Full name	Student ID	Percentage of work
1	Gabriele Nicchiarelli	4822677	50%
2	Alessio Mura	4861320	50%

Contacts

Alessio Mura: (ale.mura2000@gmail.com)
Gabriele Nicchiarelli: (S4822677@studenti.unige.it)

Both developers are Master Students in Robotics Engineering at the University of Genova.

Contents

1 Introduction

Integrating advanced technologies becomes imperative in the ever-evolving transportation landscape, especially in critical scenarios like the delivery of life-saving medicines. The swift and reliable transport of medical supplies, particularly in emergencies or to remote locations, can mean the difference between life and death. Traditional methods often face limitations in terms of speed, accessibility, and scalability, prompting the exploration of innovative solutions that leverage emerging technologies.

This project explores the implementation of a simulated prototype of a drone real-time path planner tailored specifically for the missions involved in transporting life-saving medicines. Recognizing the urgency and sensitivity of such deliveries, our focus is on developing a system that can deliver medicines successfully in an urban scenario, specifically in the city of Genoa.

One of the challenges encountered in this endeavor was the limited battery life of the drone. To address this issue, various charging stations were integrated into the system. These waypoints serve as strategic locations where the drone can recharge, ensuring continuous operation and minimizing downtime during delivery missions.

ROS (Robot Operating System) serves as the cornerstone of our implementation, providing a cohesive platform for drone control and system orchestration, making our solution efficient and adaptable to various operational environments. Additionally, we utilize the AirSim ROS Wrapper to seamlessly connect Unreal Engine with ROS. This integration allows us to control drones within Unreal Engine's immersive environment, facilitating realistic visualization and simulation of mission scenarios. By combining simulation and control, we can thoroughly test and validate our path planner in diverse virtual environments, ensuring its efficacy and reliability in real-world applications.

In the subsequent sections of this report, we delve into the intricacies of our project, providing detailed insights into its design principles, algorithmic underpinnings, and functional capabilities.

Here is the link to the GitHub repository with all the information needed for installing and running the simulation:

https://github.com/gabri00/VR-assignment3

2 State of the art

In recent years, there has been a surge in interest and utilization of Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, across a wide range of sectors. One area where drones have particularly stood out is in the transportation of life-saving medicines. These unmanned aircraft have swiftly risen to prominence as invaluable assets in the delivery of critical medical supplies [1]. In this chapter, we delve into the application of UAVs in the delivery of medical supplies, exploring their advantages, disadvantages, challenges, and potential for enriching our comprehension of medicine transportation.

2.1 UAVs for drug delivery

Recent advancements in vehicular technology have propelled UAVs to the forefront of smart city applications and services. These versatile drones are increasingly vital across various domains including photogrammetry, air pollution monitoring, security and surveillance, and disaster management. Their ability to operate autonomously and gather data swiftly makes them indispensable in urban environments. Moreover, UAVs are revolutionizing logistics, particularly in package delivery, where they optimize efficiency in terms of both time and cost [2]. Utilizing the Global Navigation Satellite System (GNSS) for precise location determination, coupled with cloud processing enabled by robust 5G networks, drones facilitate real-time communication among devices simultaneously.

Beyond commercial applications, UAV technology holds immense promise in public health. For example, drones can be deployed to deliver medical supplies and life support equipment directly to patients' homes [3], particularly benefiting individuals who are elderly, disabled, or immunocompromised. This autonomous delivery system not only enhances convenience but also reduces the need for physical visits to healthcare facilities, thereby improving overall urban public health.

The COVID-19 pandemic underscored the urgency of such innovations. A contactless "to-the-door" delivery system emerged as a crucial solution to meet healthcare demands while adhering to social distancing protocols and lockdown restrictions. This capability not only addresses current challenges but also prepares cities for future pandemics by bolstering healthcare logistics and resilience.

In essence, the integration of UAVs into smart city frameworks represents a leap toward more efficient, responsive, and resilient urban environments [4]. From enhancing public services to mitigating health crises, these advancements exemplify the potential of technology to shape a safer and more connected future.

2.2 Advantages

The potential use of drones in public health is vast and continuously expanding. Some applications include:

2.2.1 Transport of blood, medicines, and biologicals

In developing countries like India, there is often a shortage of safe blood in remote areas. Drones can transport blood from urban blood banks to rural health centers for surgeries or deliveries, eliminating the need for local blood storage and reducing costs [5]. They are also useful for delivering blood to mass casualty sites. The Rwandan government uses drones to quickly and cost-effectively deliver blood to clinics in hard-to-reach locations. In rural areas, drones can overcome delays in disease diagnosis caused by the lack of laboratory facilities. Health workers can collect patient samples and send them to the nearest laboratory via drone, receiving results and medications if needed. In Madagascar, drones have successfully transported blood samples from villages to city labs. In Malawi, dried blood samples from infants have been transported by drones for HIV testing. Similarly, sputum samples to detect tuberculosis have been sent from remote villages to urban centers in Papua New Guinea. Drones can also deliver essential medicines like anti-venom for snake and dog bites, preventing fatalities in rural areas [6].

2.2.2 Medical emergencies and disaster relief

During disaster relief operations, drones can assist in rescuing victims from collapsed buildings or locating lost fishermen at sea. They can deliver food, water, and medicines to disaster-stricken areas and injured patients on offshore ships, providing crucial supplies when traditional transport routes are blocked or damaged [7]. Drones can also serve as emergency ambulances, transporting devices like Automated External Defibrillators (AED) to cardiac arrest sites to increase survival rates. AED-equipped drones can provide visual feedback and assist bystanders in performing CPR, bridging the gap until professional medical help arrives [8]. Drones are also efficient in locating drowning victims and delivering flotation devices compared to traditional surf lifeguards, potentially saving lives in coastal regions and during floods [9].

2.2.3 Surveillance in difficult areas

Drones can quickly conduct surveys in inaccessible regions, providing real-time data and high-resolution imagery for various applications. Following the 2011 earthquake and tsunami in Fukushima, Japan, drones equipped with gamma-spectrometers were used to map nuclear contamination, providing crucial data on the affected environment. This information was vital for disaster response teams and helped in planning decontamination and evacuation efforts [10]. Drones can also be used for environmental monitoring, tracking wildlife, and assessing damage after natural disasters, making them invaluable tools for research and emergency management. Their ability to reach remote areas quickly and safely makes them ideal for surveying and responding to hazardous situations without putting human lives at risk [11].

2.3 Disadvantages

Here we present some limitations to deploying drones in public health:

2.3.1 Manpower and infrastructure

Effective drone operation necessitates trained personnel who can monitor flights and intervene if necessary. This includes pilots who understand drone flight dynamics and ground operators who manage logistics and data transmission. Introducing drones into sectors like medical transport may potentially displace traditional transport roles. For instance, drivers currently transporting medical supplies could face job losses. However, this shift also opens up opportunities for new roles specializing in drone operation and maintenance [12]. Traditional drones typically require dedicated runways or flat surfaces for takeoff and landing. In areas lacking such infrastructure, this poses a challenge. However, newer drone models with vertical takeoff and landing capabilities (VTOL) mitigate this issue, as they can operate in tighter spaces without needing extensive ground infrastructure.

2.3.2 Technical Limitations

Drones typically have a limited payload capacity, ranging between 2 to 4 kg. This restricts their ability to carry heavier items compared to commercial planes or helicopters, limiting their application for large-scale deliveries or transportation of bulky goods. As drones become more sophisticated and capable, their weight and cost tend to increase. Balancing intelligence with affordability and practicality becomes crucial in determining their operational feasibility [13]. The safety and efficiency of drone operations are not yet fully established. Ensuring reliable and secure transportation, especially for delicate items like biological samples, requires robust packaging and handling protocols to prevent tampering and damage during transit. Transporting sensitive medical supplies such as vaccines requires maintaining a cold chain. Drones need to be equipped with ice packs or coolers to ensure temperature stability throughout the delivery process, which adds complexity and weight to their design. Battery life remains a significant concern for drones, as it limits operational range and duration. Advances in battery technology and the exploration of solar-powered drones, such as Facebook's Aquila, aim to extend flight times and reduce dependency on traditional charging methods [14]. Drones' ability to withstand adverse environmental conditions like strong winds, turbulence, and precipitation is not universally clear. These factors can impact flight stability and pose risks to both the drone and its payload. Electromagnetic interference can disrupt signal reception between drones and ground control stations, potentially compromising operational control and data transmission reliability [15].

2.3.3 Regulations and legality

Regulations and legality play pivotal roles in shaping the deployment and operational scope of medical drones globally. In India, stringent regulations require specific aviation authority authorization for commercial drone use, including medical applications, to ensure airspace safety. However, these rules hinder the widespread adoption of medical drone deliveries and restrict innovative healthcare solutions' flexibility [16]. In contrast, the United States implements FAA regulations requiring licenses for commercial drone operators, limiting drones to 25 kg, and mandating operations within visual line-of-sight for enhanced airspace and public safety [17]. Globally, navigating these regulatory landscapes poses challenges, with complex bureaucratic processes delaying deployment and increasing operational costs. Strict safety protocols, pilot certifications, and operational limitations further impede the scalability and efficiency of medical drone initiatives aimed at enhancing healthcare delivery.

Operational limitations imposed by regulations, such as VLOS requirements and altitude restrictions, are intended to mitigate risks but can also constrain the effectiveness of medical drone operations. Variations in drone regulations across different countries present additional challenges. Multinational companies and global initiatives face complexities in harmonizing operations and complying with diverse regulatory requirements when deploying drones across jurisdictions. This fragmentation necessitates collaborative efforts between stakeholders, governments, and regulatory bodies to establish standardized practices that support innovation while ensuring safety [18].

3 Tools

For this project, we constructed a simulated environment using Unreal Engine (v. 5.2.2). We employed AirSim and ROS to control the drone. This combination of technologies facilitated the seamless integration of drone control within the simulated environment, allowing for realistic and dynamic simulation of drone missions.

3.1 Unreal Engine and Cesium

To build a simulated city for our project, we installed Cesium from the Unreal Engine Marketplace [19]. This plugin allowed us to reconstruct the city of Genova using Google Maps satellite images. To do so we followed these steps:

1. Connect to Cesium ion, to get global 3D geospatial data, and create a new token to access its resources.
2. Create a globe within our simulated environment.
3. Add the *Google Photorealistic 3D Tiles* to import photorealistic terrain.
4. Select the desired coordinates where to locate our georeference with the *CesiumGeoreference* actor.

After these steps, we adjusted the georeference to match the coordinates of Genoa. Using the Google 3D Tiles also allowed us to import automatically the collision box for buildings and other objects.

3.2 Colosseum (AirSim)

To connect the drone to the UE environment and to develop the control side we utilized *Colosseum* [20] which is basically a successor of *AirSim* for versions of UE greater than 5.0.
AirSim is an open-source simulation platform for testing autonomous systems, specifically UAVs, in a virtual environment. Its primary function is to provide a high-fidelity simulation environment that accurately replicates the physics and dynamics of vehicles. Additionally, AirSim offers a suite of sensors to equip the drone. AirSim's potential is significantly enhanced by the development of additional tools. One such tool is the ROS (Robot Operating System) Wrapper, which integrates AirSim with ROS, a flexible framework for writing robot software. This integration allows for seamless communication between AirSim and ROS, enabling the use of ROS packages and tools for developing and testing autonomous systems.

3.2.1 AirSim ROS Wrapper

In our project, we integrated ROS with AirSim to better control the drone [21]. ROS, a widely adopted open-source framework in robotics, provides a flexible and modular platform for controlling autonomous systems. This integration offers several advantages, including access to ROS extensive library of tools and algorithms, facilitating the development and testing of our control strategies. Additionally, ROS distributed architecture enables concurrent and collaborative development, enhancing modularity and code reusability. The integration also allows for easy transition from simulation to real-world deployment, reducing risks and costs.

3.2.2 AirSim API

The developed API enables programmatic interaction with the spawned vehicle in the simulation [22]. It creates a client that provides functions for controlling the vehicle and retrieving its state, as well as methods for capturing data from sensors, pausing the simulation, changing time-of-day and weather settings, recording the simulation, and controlling multiple vehicles.
AirSim supports two game modes (Car and Multirotor), so there are specific client instances for each mode: *CarClient* and *MultirotorClient*. In our case, since we required a drone we utilized an instance of the *MultirotorClient*.

3.3 D-Flight

D-Flight, short for Digital Flight Control System, is an autonomous flight control system developed by ENAV, the Italian National Civil Aviation Authority [23]. This system is designed to enable autonomous drone flights in both controlled and uncontrolled airspace, ensuring the safety of aerial operations and compliance with civil aviation regulations.
One distinctive feature of D-Flight is its ability to integrate with existing air traffic management (ATM) systems. This means that the system can interact with air traffic control and others in the civil aviation sector, enabling the safe planning and execution of drone flight missions in shared airspace. Additionally, D-Flight provides an interface that allows operators to program drone flight missions, monitor their progress, and intervene if necessary. This interface can be accessed via mobile devices or computers, enabling flexible and remote control of drone flight operations.
In particular, we used the D-Flight services to control the flight restrictions in the city of Genova to plan our drone operations respecting the elevation limits imposed.

4 Project description

In the following section, we will analyze the various fundamental components of our project. This detailed examination aims to provide a comprehensive understanding of each part, highlighting its significance, functionality, and contribution to the overall success of the project.
The main parts of the assignment are:

1. Fly the drone from the start location to the goal while respecting the elevation limits imposed by D-Flight.
2. Allow the drone to perform online obstacle avoidance.
3. Manage the drone autonomy and allow the drone to reach recharging stations if necessary.

Below is a detailed representation of the software architecture (Fig. 1). This scheme represents the how we interfaced the ROS architecture with UE and AirSim to manage and control the simulated drone within the Unreal Engine environment.

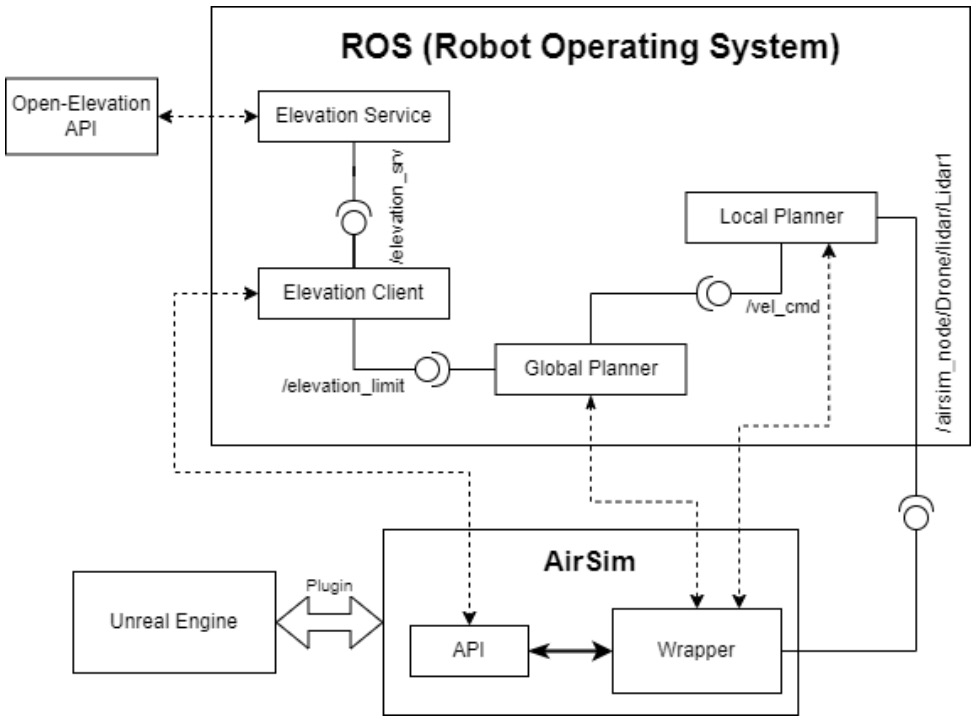

Figure 1: Software Architecture.

4.1 Flight restriction areas

To recreate the flight restriction areas, we used d-flight to evaluate the restricted access areas within the city to identify where drones are prohibited from flying or the maximum altitude at which they can operate (Fig. 2).

4.1.1 Extraction of the areas

Since D-Flight doesn't provide a direct API to extract these areas we created by hand an approximation of the flight restriction areas using Google Earth tools [24], this interface allows us to define 2D polygons on the Google Earth satellite map and associate properties to these areas. In particular, we created six polygons, associated a color code (blue, yellow, orange, and red) to each region to match the D-Flight color code, and directly defined the elevation limit inside the description of each area. We finally exported the polygons in Keyhole Markup Language (KML) format (Fig 3). The KML files provided us with precise georeferenced coordinates of each area, which were crucial in ensuring that our drones could interpret these restrictions. Integrating this data with the drone's GPS allows it to recognize and adhere to flight restrictions based on its current location.

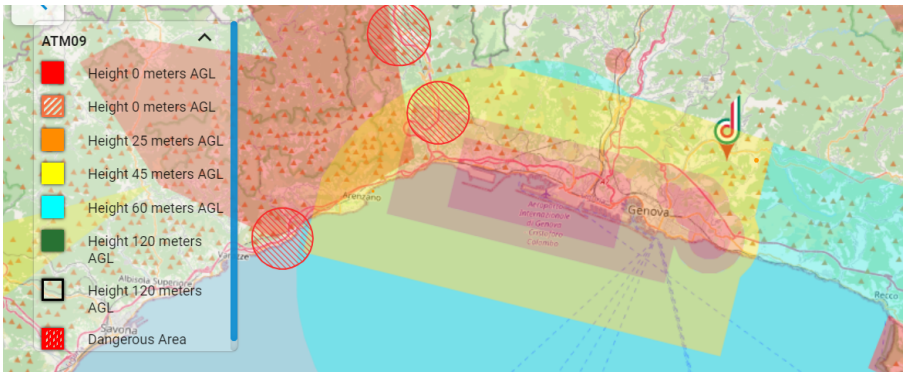

Figure 2: D-flight restriction areas (Genova).

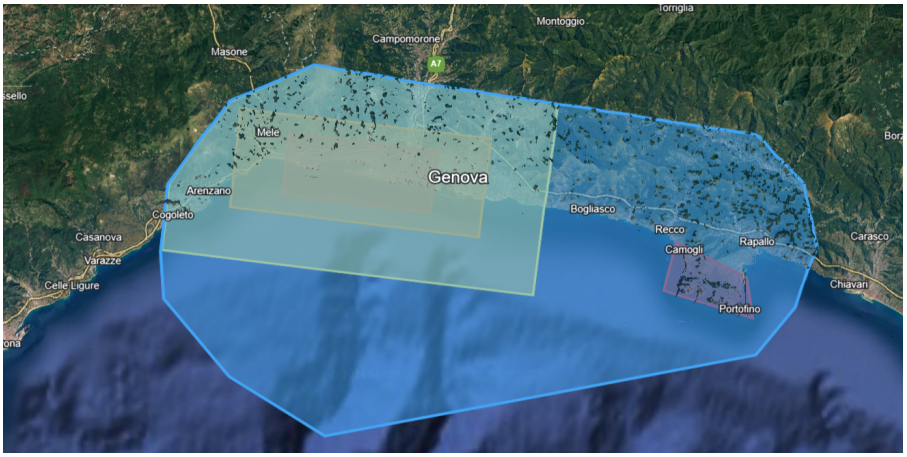

Figure 3: Reconstruction of the restriction areas.

4.1.2 KML standard

To give a quick insight into the KML file format, we report an extract of our KML containing the flight restriction zones. Each area is represented by a `Placemark`, which has a unique ID. The `Placemarker` is defined by one or more polygons, a name, and a description. Also, a polygon can be defined by outer boundaries and inner boundaries to create "holes" in the polygon. Finally, each vertex of the polygon is represented by latitude, longitude, and altitude. The latter is always set at zero since it's not used for our specific application.

Listing 1: "Genova/Sestri 10/28 - 2" area in KML format

```
<Placemark id="06E8D0A0A9301B772B6B">
    <name>Genova/Sestri 10/28 - 2</name>
    <description>25</description>
    <styleUrl>#__managed_style_0F3A032A8E301B779BBA</styleUrl>
    <Polygon>
        <outerBoundaryIs>
            <LinearRing>
                <coordinates>
                    8.970149186245829,44.43400872669746,0
                        8.722274926242237,44.46342916051803,0
                        8.706214284577364,44.3940899529064,0
                        8.953273047256165,44.36417709790977,0
                        8.970149186245829,44.43400872669746,0
                </coordinates>
            </LinearRing>
        </outerBoundaryIs>
        <innerBoundaryIs>
            <LinearRing>
                <coordinates>
                    8.916799793451505,44.42656287067967,0 8.769244943517801,44.4443433516666,0
                         8.758810557605276,44.40105050966812,0
                        8.905707351854096,44.38310333734523,0
                        8.916799793451505,44.42656287067967,0
                </coordinates>
            </LinearRing>
        </innerBoundaryIs>
    </Polygon>
</Placemark>
```

4.1.3 Code integration

To manage this information, two Python ROS scripts have been created:

1. `elevation_cli.py` handles the client-side operations. First, it loads the KML file, and then it periodically (every 5 seconds) checks the drone elevation's state. It retrieves the drone coordinates and determines if it is within a restricted area. If so, it fetches the elevation making a request to the service provided by the `elevation_service.py` script, and publishes the maximum allowable altitude. The altitude that the drone can reach at a specific point is defined as $limit = terrain\_elevation + area\_limit$

2. The terrain elevation has to be retrieved from an external source since AirSim isn't able to get it. To do so, the `elevation_service.py` provides a ROS service that fetches elevation data for specified coordinates by querying an external API, *Open-Elevation* [25], processes the response, and returns the elevation.

Together, these scripts ensure the drone operates safely and adheres to the defined flight restrictions.

4.2 Obstacle avoidance and navigation

The most critical task of the assignment was to allow the drone to navigate in the urban area safely avoiding object collisions. To do so, the drone has to be equipped with sensors to have the perception of the surroundings; for this project, we chose to use *LIDAR* (Laser Imaging Detection and Ranging) sensors.
These types of sensors use a method for determining ranges by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. Lidar may operate in a fixed direction (e.g., vertical) or it may scan in multiple directions, in which case it is known as lidar scanning or 3D laser scanning. Lidar is commonly used to make digital 3D representations of areas on the Earth's surface by varying the wavelength of light. It has also been increasingly used in control and navigation for autonomous vehicles.
The LIDAR settings have been specified in the `settings.json`. In particular, since we are interested only in scanning the horizontal plane, we set the vertical Field of View (FOV) to 0° and the horizontal FOV to 150°. Also, the "accuracy" of the LIDAR has been set to a reasonable value by setting the `RotationsPerSecond` to 10 and the `PointsPerSecond` to 500. These values have been chosen accurately to avoid computational overload.
After the definition of the sensor part, we've taken into account the control part by splitting the controller into a global planner and a local planner.

4.2.1 Local planner

The local planner is constantly looking for obstacles using the LIDAR sensor data retrieved with a ROS message directly provided by AirSim. The points returned by the sensor are then split into front, left, and right, to define three main areas in which the obstacles can be identified.
The control logic is straightforward: if the drone finds an obstacle in front of it, it turns left or right. To decide where to turn we used the sum of the distances as metrics. For example, if the object detected is more prominent on the right, then the drone turns to the left. The sign of the speed gain determines the direction of the drone along the Y-axis: the left has a negative speed while the right has a positive sign.

Listing 2: Turn sign calculation

```
    if sum(right_distances) > sum(left_distances):
        turn_sgn = 1
    else:
        turn_sgn = -1
```

Since we referenced the drone movements to its local reference frame (which is NED coordinates), when it encounters an obstacle we provide speed only on the Y-axis, while for forward movements we provide a velocity vector on the X-axis.
Finally, the local planner sends the computed velocity to the global planner with a custom ROS message.

4.2.2 Global planner

The global planner takes care of most stuff:

1. Initializing the drone with all the required parameters.
2. Loading weather effects.
3. Handling the autonomy of the device.
4. Controlling the drone.

It initializes a ROS timer of 1 second that serves as a control loop. Note that we kept the control relatively slow due to lags in our environment, however, theoretically, it can be faster at the cost of more computational power.
The controller handles the movements along the Z-axis and those on the horizontal plane (XY), movements on the horizontal plane are allowed only if the drone has reached the desired elevation.
To handle movements on the horizontal plane, the global planner aligns the drone's "head" with the goal continuously and sends the velocity command received from the local planner to the drone.

4.3 Battery management

The drone autonomy is handled inside the `global_planner.py` file. The durability of the device is computed beforehand. If the drone can't reach the goal he will head towards a charging station first.
The power consumption takes into account the drone's weight, the payload weight, and the weather disturbances:

$$autonomy = \frac{100}{0.03 * (weight_{drone} + weight_{payload}) + weather}$$

The worse the weather conditions, the lower the autonomy. The same can be applied to the weights.
Only 90% of this autonomy value is considered in order to have some overhead in case of obstacles encountered.
Once the drone computes its autonomy, it will check if it can reach the goal directly. If it can't head towards the goal, it will look for the closest charging stations and go to that station before reaching the goal.

5 Conclusions

In conclusion, our project aims to deliver a straightforward and effective solution for navigating drones while ensuring obstacle avoidance, specifically designed for transporting medical supplies. Throughout this assignment, we addressed crucial aspects such as adhering to flight zone regulations and ensuring the autonomy of the UAVs to complete their missions. The system developed offers a user-friendly interface that allows autonomous drone navigation, facilitating the safe and efficient transport of medical goods.
In conclusion, this project has demonstrated the feasibility and effectiveness of using advanced simulation, control algorithms, and ROS integration for autonomous drone navigation and obstacle avoidance. This project lays a solid foundation for future advancements in UAV technology and its application in critical tasks for medical goods transportation.

5.1 Possible improvements

1. To improve obstacle avoidance beyond the limitations of our current simple lidar-based algorithm, despite computational constraints, implementing more complex algorithms based on cameras and machine learning could be a promising direction. Cameras can provide visual data that lidar alone might miss, such as small or transparent obstacles, and improve the accuracy of obstacle recognition. Utilizing machine learning models trained on diverse datasets to recognize and classify obstacles in real-time camera feeds. These models can distinguish between different types of obstacles and predict their movements or behaviors, enabling proactive avoidance maneuvers.

2. The battery management could be improved by handling the battery consumption online instead of doing an offline autonomy computation. This could reduce the risk of wasting battery during the mission.

6 References

References

[1] Al-Wathinani AM, Alhallaf MA, Borowska-Stefańska M, Wiśniewski S, Sultan MAS, Samman OY, Alobaid AM, Althunayyan SM, Goniewicz K. 2023. Elevating Healthcare: Rapid Literature Review on Drone Applications for Streamlining Disaster Management and Prehospital Care in Saudi Arabia. *Healthcare (Basel)*. 11(11):1575. doi: 10.3390/healthcare11111575.

[2] Francesco Betti Sorbelli. 2024. UAV-Based Delivery Systems: A Systematic Review, Current Trends, and Research Challenges. *ACM J. Auton. Transport. Syst.* 1(3):Article 12. doi: 10.1145/3649224.

[3] Jalel EUCHI. 2021. Do drones have a realistic place in a pandemic fight for delivering medical supplies in healthcare systems problems? *Chinese Journal of Aeronautics*. 34(2):182-190. doi: 10.1016/j.cja.2020.06.006.

[4] Vodák, Josef, Dominika Šulyová, and Milan Kubina. 2021. Advanced Technologies and Their Use in Smart City Management. *Sustainability*. 13(10):5746. doi: 10.3390/su13105746.

[5] Mora, P., & Araujo, C. A. S. 2021. Delivering blood components through drones: a lean approach to the blood supply chain. *Supply Chain Forum: An International Journal*. 23(2):113–123. doi: 10.1080/16258312.2021.1984167.

[6] Laksham, Karthik Balajee. 2019. Unmanned aerial vehicle (drones) in public health: A SWOT analysis. *Journal of Family Medicine and Primary Care*. 8(2):342-346. doi: 10.4103/jfmpc.jfmpc_413_18.

[7] Sharifah Mastura Syed Mohd Daud et al. 2022. Applications of drone in disaster management: A scoping review. *Science & Justice*. 62(1):30-42. doi: 10.1016/j.scijus.2021.11.002.

[8] Clément Derkenne et al. 2021. Automatic external defibrillator provided by unmanned aerial vehicle (drone) in Greater Paris: A real world-based simulation. *Resuscitation*. 162:259-265. doi: 10.1016/j.resuscitation.2021.03.012.

[9] K. Ajgaonkar et al. 2020. Development of a Lifeguard assist Drone for coastal search and rescue. In *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pp. 1-10. doi: 10.1109/IEEECONF38699.2020.9389382.

[10] Takuya Kurihara et al. 2020. Precision spectroscopy of cesium-137 from the ground to 150 m above in Fukushima. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 978:164414. doi: 10.1016/j.nima.2020.164414.

[11] Shahbazi, M., Théau, J., & Ménard, P. 2014. Recent applications of unmanned aerial imagery in natural resource management. *GIScience & Remote Sensing*. 51(4):339–365. doi: 10.1080/15481603.2014.926650.

[12] Mohsan, Syed Agha Hassnain et al. 2022. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review. *Drones*. 6(6):147. doi: 10.3390/drones6060147.

[13] M. Hassanalian, A. Abdelkefi. 2017. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*. 91:99-131. doi: 10.1016/j.paerosci.2017.04.003.

[14] Abidali, A., Agha, S.A., Munjiza, A. et al. 2024. Development of a solar powered multirotor micro aerial vehicle. *Sci Rep*. 14:5771. doi: 10.1038/s41598-024-54079-9.

[15] Lykou, Georgia, Dimitrios Moustakas, and Dimitris Gritzalis. 2020. Defending Airports from UAS: A Survey on Cyber-Attacks and Counter-Drone Sensing Technologies. *Sensors*. 20(12):3537. doi: 10.3390/s20123537.

[16] Mehrotra, Kushagra. 2024. Drone Regulations: the Indian Scenario. Available at SSRN: https://ssrn.com/abstract=4817307 or doi: 10.2139/ssrn.4817307.

[17] Bradley, Holden Luc Schmidt. 2020. Identifying Opportunities Available Through Utilization Of Unmanned Aerial Delivery Programs In Support Of Public Safety In The United States, Based On Present Use Cases Around The World. Available at: http://purl.flvc.org/fsu/fd/FSU_libsubv1_scholarship_submission_1587494473_43a93295.

[18] McTegg, Stephen John et al. 2022. Comparative Approach of Unmanned Aerial Vehicle Restrictions in Controlled Airspaces. *Remote Sensing*. 14(4):822. doi: 10.3390/rs14040822.

[19] Cesium for Unreal. Available at: https://cesium.com/platform/cesium-for-unreal/.

[20] CodexLabs LLC. Colosseum. Available at: https://github.com/CodexLabsLLC/Colosseum.

[21] Microsoft. AirSim ROS Packages. Available at: https://microsoft.github.io/AirSim/airsim$_{r}os_{p}kgs$/.

[22] Microsoft. AirSim. Available at: https://microsoft.github.io/AirSim/.

[23] d-flight. Available at: https://www.d-flight.it/web-app/.

[24] Google Earth. Available at: https://earth.google.com/web/.

[25] Jorl17. Open Elevation API Documentation. Available at: https://github.com/Jorl17/open-elevation/blob/master/docs/api.md.