Path-following and OA algorithm for an UAV
&
Minimal rigid-body transform contractor using interval approaches

Danut POP : danut.pop@ensta-bretagne.org

Leibniz Universität Hannover (LUH)
Welfengarten 1, 30167 Hannover
Supervisers : Aaronkumar Ehambram & Bernardo Wagner
emails : ehambram@rts.uni-hannover.de & wagner@rts.uni-hannover.de

École Nationale Supérieure des Techniques Avancées de Bretagne

2 Rue François Verny, 29200 Brest

Supervisers : Luc Jaulin & Benoît Zerr

emails : lucjaulin@gmail.com & benoit.zerr@ensta-bretagne.fr

Index

1 The equipment & the framework 4
1.1 The drone & the sensors 4
1.2 The framework: architecture and requirements 5

2 Research of the algorithms and tools 6
2.1 Path-following algorithms 6
2.1.1 Pure Pursuit 6
2.1.2 Lyapunov Method 7
2.2 Path Frame and error equation 7
2.2.1 Path Frame 7
2.2.2 Path Error Equation 8
2.3 Mathematical convergence of the controller 9
2.4 Obstacle Avoidance Algorithm 10
2.4.1 The different algorithms 10

3 Implementation 11
3.1 Results of the first tests 11
3.2 Analysis & explanation of the results 12
3.3 Solution to deal with overshooting & delay 14
3.4 Final results in simulation 16
3.5 Tests and results on the real drone 17
3.6 Obstacle Avoidance 18
3.7 Conclusion 19

4 Second phase: building a rigid-body transform minimal contractor for SLAM purposes 19
4.1 The polar equation 19
4.2 Minimal contractor for the 3D polar constraint 19

5 Conclusion 21

Introduction

Across the entirety of the globe, regardless of temporal proximity, seismic activity, coastal disturbances such as tsunamis, or natural degradation, owing to aging, can precipitate the collapse of buildings. The recent occurrences in cities like Marseille, regions bordering Turkey and Syria, or even within the United States, serve as compelling evidence that such unfortunate events can manifest ubiquitously and unpredictably:

- Border between Turkey and Syria (Feb. 6 2023): https://www.youtube.com/watch?v=buLMbZhp5rI
- France, Marseille (Apr. 9 2023): https://www.youtube.com/watch?v=VAJSIuj0SSA
- US, Miami (June 24 2021): https://www.youtube.com/watch?v=iO9kjwo4x-0

In these dire circumstances, individuals face the risk of sustaining injuries or even losing their lives. Consequently, the response to such catastrophic events must be both swift and secure. In practice, this necessitates the coordinated mobilization of a team of firefighters, medical personnel, and rescue teams. Their primary objective is to efficiently enter the affected building, conduct thorough search and rescue operations to locate victims, and then safely transport them to designated zones for proper medical treatment and care.

As shown in news reports, building collapses often exhibit a cascading effect, potentially impacting neighboring structures with undetermined time intervals that span from several minutes to numerous hours. This uncertain period poses considerable risks for rescuers, as the danger of entering a collapsed building to search for victims remains equally hazardous after a few minutes, hours, or even an extended duration such as a day.

Hence, there is a pressing necessity for a rapidly deployable and non-human rescue solution. One promising approach involves deploying drones to conduct internal scans of the affected building and search for victims. Upon completing the reconnaissance phase, the drones would return from the building equipped with a detailed interior map, along with the positions of potential victims. Thus far, the most suitable system for this crucial mission is the UAV (Unmanned Aerial Vehicle), more commonly known as an autonomous flying drones, given their adaptability and self-directed capabilities.

The Leibniz Universität Hannover (LUH) is conducting a research project formally named *UAV-Rescue* to demonstrate the feasibility of using drones for rescue operations.

Objectives

During the initial phase of the internship, spanning from March to the end of August, the primary objective is to implement and validate a robust and highly accurate state-of-the-art path-following algorithm. This algorithm will be tested and refined through simulations on the Gazebo platform before being deployed on the actual drone. It must ensure that the UAV's position error concerning the intended path is kept within a stringent tolerance of less than 10 cm. Additionally, the algorithm should enable the UAV to promptly rejoin the path in case of disturbances, such as wind-induced deviations.

Furthermore, a second essential functionality, local obstacle avoidance, must be developed. The UAV should possess the capability to circumvent unforeseen obstacles encountered during its flight path. In the event of a significant obstruction, the system should halt its original path-following trajectory and prompt the path planner to compute an alternative course.

During the subsequent and last phase, the focus shifts to solving a Simultaneous Localization and Mapping (SLAM) graph algorithm using interval analysis methods, contributing to further advancements in the UAV's navigational capabilities. More specifically, when the UAV is in motion, the same 3D point clouds can be observed from different perspectives between two consecutive measurements. The relationship between those two point clouds is affine and the problem amounts to finding the coefficients. Those coefficients represent the rotation and translation between the cloud points. Therefore, the last phase of the internship is dedicated to employing an interval-based approach to develop a succinct, minimal rigid-body contractor to effectively tackle this intricate problem.

1 The equipment & the framework

1.1 The drone & the sensors

The selected vehicle for deployment is an octocopter-like drone. Herewith is an image depicting the system:

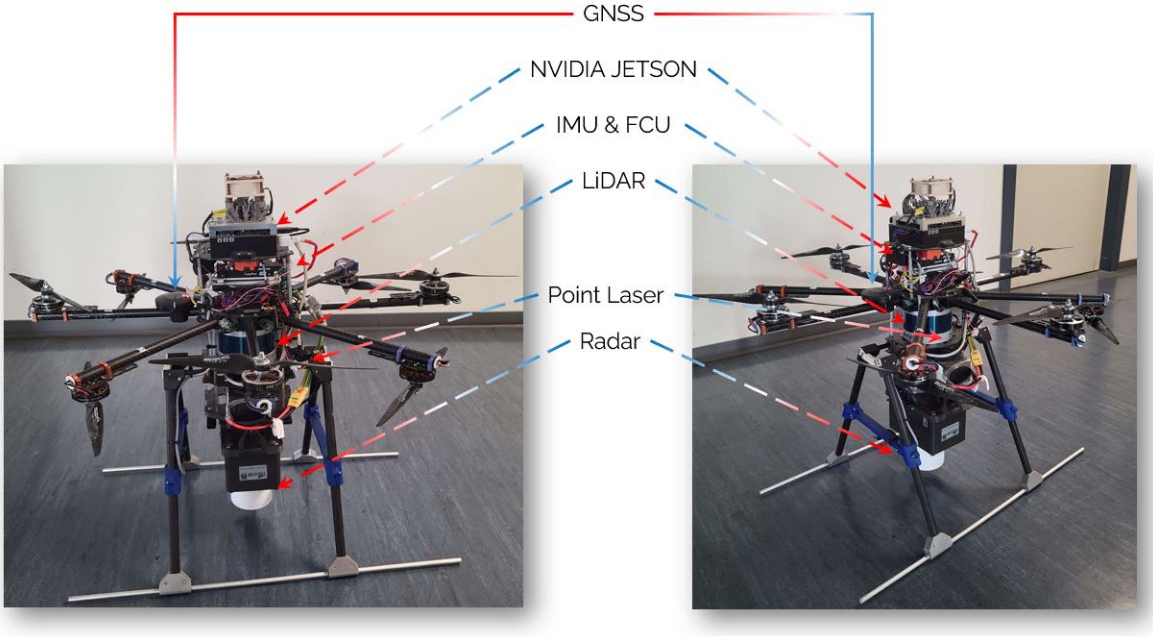


Figure 1: UAV and its sensors

It is equipped with the following sensors:

- NVIDIA Jetson AGX-Xavier
- Cube Orange FCU (Flight Controller Unit)
- LiDAR Velodyne & Radar
- GNSS
- Inertial Measurement Unit (IMU)
- Point Laser Sensor

Herein is a detailed account of the sensors utilized to ensure effective operation and navigation: The NVIDIA Jetson serves as the embedded motherboard responsible for essential computations. The Cube Orange Flight Control Unit utilizes desired accelerations as input to calculate motor commands required for flight control. To enable SLAM capabilities, the system integrates the Velodyne LiDAR sensor along with an IMU and a Laser Point sensor, which collectively contribute to mapping the environment and estimating local position and speed.

In situations where the drone commences its operation outdoors, the GNSS is employed to transform the SLAM's local reference into a global one, providing accurate global positioning information. Moreover, the system incorporates a Radar sensor, which plays a key role in human detection. By combining these components, the drone system achieves a comprehensive and advanced suite of functionalities, facilitating safe and efficient operations in diverse environments.

1.2 The framework: architecture and requirements

The LUH Team has developed a framework based on ROS 1 to facilitate seamless software integration that enables it to control the drone effectively through the utilization of MAVROS (MAVLink-based communication node for ROS).
As part of the team's ongoing efforts to enhance functionality, programs will be implemented as packages. However, it is essential to consider certain specific requirements during this process:

- Detect and always allow the user to take over control inputs via the remote. This feature ensures that operators have the ability to intervene and override autonomous commands, thereby enhancing safety during operations.
- Implementation of a ROS Action Server architecture for takeoff, landing, and path-following. Utilizing the ROS Action Server will enable efficient coordination of these maneuvers. Additionally, the system will feature a real-time Human-Machine Interface (HMI) that displays essential information, providing operators with real-time data for informed decision-making during flight. This combination of user control override, ROS Action Server, and real-time HMI will create a reliable and capable drone control system, suitable for a wide range of applications.

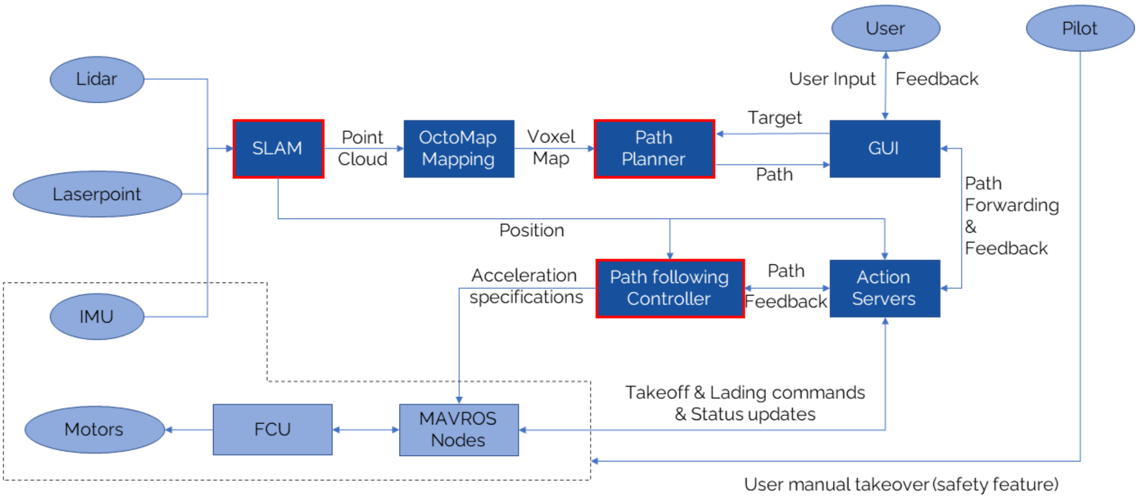


Figure 2: Framework Architecture

NOTE: MAVROS stands for MAVLink on ROS. It is a ROS package that enables the communication between computers running ROS and MAVLink-enabled autopilots (e.g. the FCU). MAVLink is a lightweight messaging protocol that is commonly used in the drone industry.
MAVROS provides a number of features for controlling drones. These features include:

- Communication: MAVROS provides a reliable and efficient way to communicate with MAVLink-enabled autopilots.
- Control: MAVROS can be used to control the movement and behavior of drones.
- Telemetry: MAVROS can be used to collect telemetry data from drones, such as their position, speed, and altitude.

For example, the BlueROVs at ENSTA Bretagne use MAVROS for receiving commands.

2 Research of the algorithms and tools

2.1 Path-following algorithms

A thorough search of the most used path-following algorithms was conducted. Herein are the results:

2.1.1 Pure Pursuit

A former intern already implemented a path-following algorithm using a **pure pursuit** method. Herein is a more detailed description of the protocol: The algorithm starts by representing the desired path as a series of waypoints. As the vehicle moves, it picks a point on the path to reach, known as the lookahead point, by considering its current location and a lookahead distance (free parameter to adjust). This lookahead point is then adjusted to lie on the path. The algorithm calculates the steering angle required to direct the vehicle toward this adjusted target point. By iteratively updating the target point and steering angle, the vehicle follows the path while adapting to changing conditions and maintaining a smooth trajectory. Presented below is an illustration of this procedural sequence:

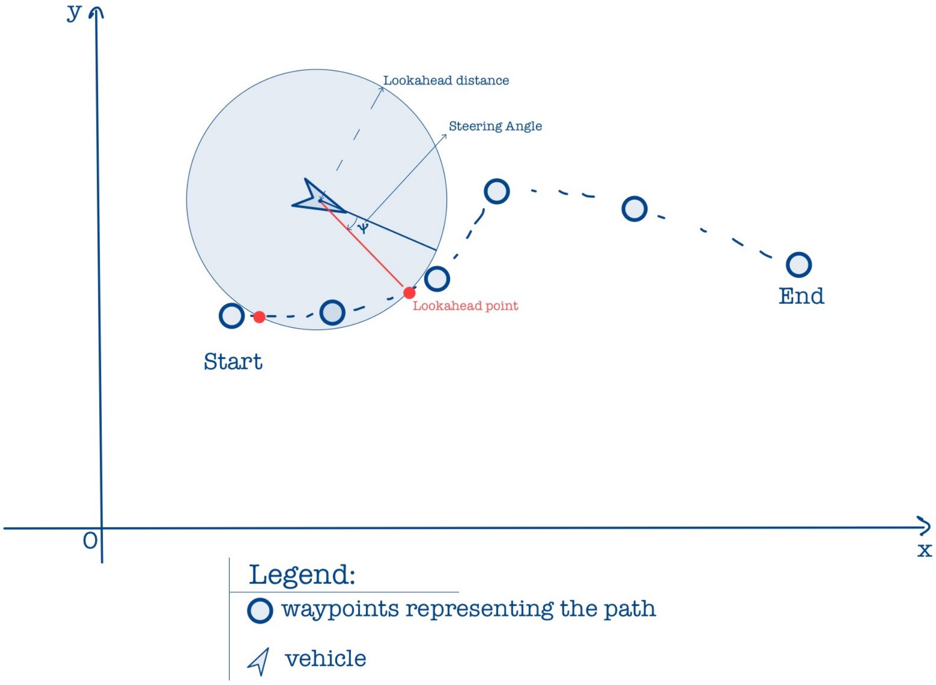


Figure 3: Pure Pursuit Algorithm

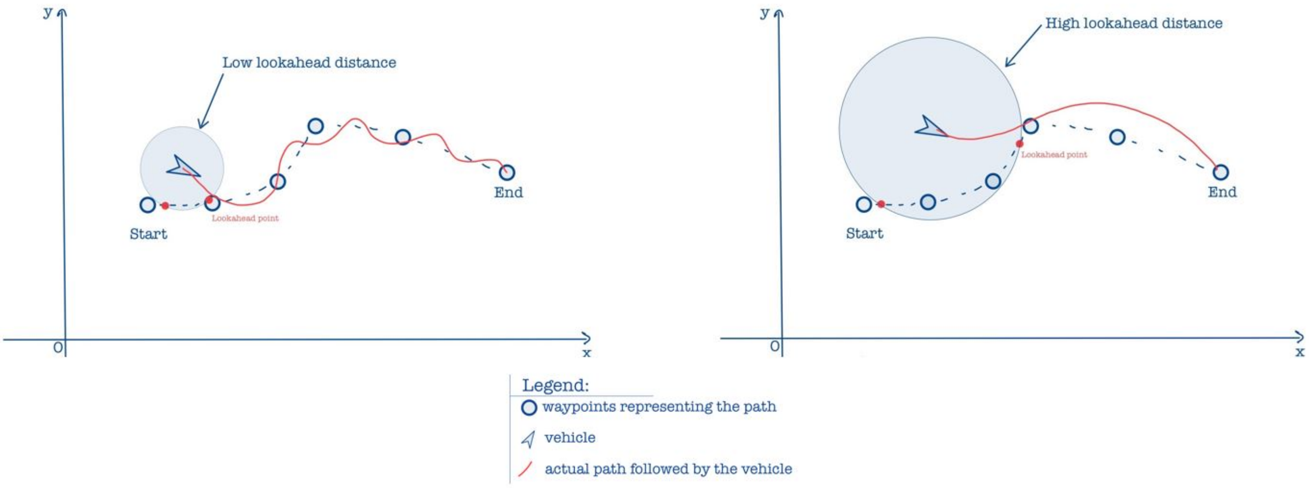


Figure 4: Pure-Pursuit lookahead distance influence

2.1.2 Lyapunov Method

The Lyapunov method was invented by the Russian mathematician Aleksandr M. Lyapunov in 1899. Lyapunov was interested in the stability of dynamical systems, and he developed the method bearing his name as a way to prove that a dynamical system is stable.
The Lyapunov theorem is stated below:

Definition (Lyapunov function):

Let the following system $\dot{x}(t) = f(x(t))$ with $x \in (\mathbb{R}^n)^{\mathbb{R}}$ and 0 being an equilibrium point. A system evolving in agreement with this equation is termed an *autonomous system* (the system's variations rely solely on its state). Let $V \in \mathbb{R}^{\mathbb{R}^n}$ be a real value function. Let $\dot{V}$ define $\dot{V}(x) = \nabla V(x) f(x)$. Then, V is said to be a Lyapunov function if it satisfies:

- V is positive definite
- $\forall x \in \mathbb{R}^n, \dot{V}(x) < 0$ and $\dot{V}(0) = 0$

Lyapunov functions are like energy functions. The conditions stated above assure that every state is dissipative and only the null state is stable.

Theorem (Lyapunov global asymptotic stability):

Let the following system $\dot{x}(t) = f(x(t)$ with $x \in (\mathbb{R}^n)^{\mathbb{R}}$ and 0 being an equilibrium point (i.e $f(0) = 0$). If there exists a Lyapunov function, then all the trajectories (i.e. the solutions to the differential equation) converge to the null vector. We would say that the system is globally asymptotically stable.

We can use the Lyapunov theorem to prove the convergence of nonlinear control laws in robotics for example.

2.2 Path Frame and error equation

2.2.1 Path Frame

The Frenet-Serret frame, though commonly employed for parameterized curves in 2D space, was not utilized in this 3D context due to its inherent singularity, which arises when the curve has vanished second derivative (the frame is not defined). Indeed, the second vector of Frenet's frame always points in the direction of the curvature and when curvature becomes zero locally, the second vector has to go from its current value to the opposite instantly. Instead, the Parallel Transport Frame method was chosen as an alternative approach. The decision to opt for the Parallel Transport Frame stemmed from its capability to (as the name suggests) transport an initial frame along the path in a continuous manner.

Let a parameterized curve f in 3D in terms of its curvilinear abscissa s. We can attach a frame at each s. The Frenet frame composed of the three vectors $\begin{pmatrix}\mathbf{T} & \mathbf{N} & \mathbf{B}\end{pmatrix}^T$ (with $\mathbf{T}$ being tangent to the path) satisfies the following equation:

$$\frac{d}{ds}\begin{pmatrix}\mathbf{T}(\mathbf{s})\\ \mathbf{N}(\mathbf{s})\\ \mathbf{B}(\mathbf{s})\end{pmatrix} = \begin{bmatrix} 0 & \kappa(s) & 0\\ -\kappa(s) & 0 & \tau(s)\\ 0 & -\tau(s) & 0\end{bmatrix}\begin{pmatrix}\mathbf{T}(\mathbf{s})\\ \mathbf{N}(\mathbf{s})\\ \mathbf{B}(\mathbf{s})\end{pmatrix}$$

With $\kappa(s)$ and $\tau(s)$ being respectively the curvature and torsion of the curve.

The Parallel transport frame satisfies the following equation [1]:

$$\frac{d}{ds}\begin{pmatrix}\mathbf{T}(\mathbf{s})\\ \mathbf{N}(\mathbf{s})\\ \mathbf{B}(\mathbf{s})\end{pmatrix} = \begin{bmatrix} 0 & k_1(s) & k_2(s)\\ -k_1(s) & 0 & 0\\ -k_2(s) & 0 & 0\end{bmatrix}\begin{pmatrix}\mathbf{T}(\mathbf{s})\\ \mathbf{N}(\mathbf{s})\\ \mathbf{B}(\mathbf{s})\end{pmatrix}$$

With k_1, k_2 being determined as such:

$$\begin{aligned} k_1 &= \kappa\ cos(\theta)\\ k_2 &= \kappa\ sin(\theta)\\ \theta &= -\int_0^s \tau(x)\,dx \end{aligned} \tag{1}$$

We will call $\mathbf{Q}$ the proportionality matrix tying $\begin{pmatrix}\mathbf{T} & \mathbf{N} & \mathbf{B}\end{pmatrix}$ to it's derivative with respect to the curvilinear abscissa.

2.2.2 Path Error Equation

In this section we will establish the error equation of the point on the path. Let a Robot evolving in the 3D space be described by its position $\mathbf{P}$ and orientation matrix $\mathbf{R}$ such as:

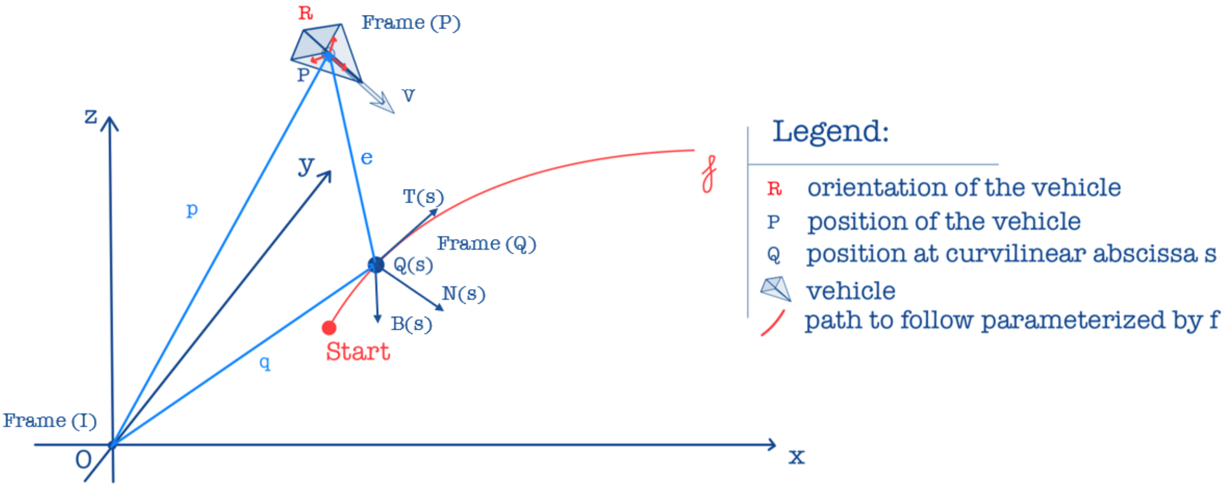

R
Frame (P)
P
V
Z
y
e
p
T(s)
f
Frame (Q)
Q(s)
N(s)
B(s)
Start
q
Frame (I)
O
x
Legend:
R orientation of the vehicle
P position of the vehicle
Q position at curvilinear abscissa s
vehicle
path to follow parameterized by f

Figure 5: Vehicle moving in 3D space

Let $\mathbf{p}, \mathbf{q}$ respectively represent the vectors $\overrightarrow{OP}, \overrightarrow{OQ}$ in the Frame (I).
Let $\mathbf{e}$ represent the error vector expressed in the path's frame, i.e., the vector $\overrightarrow{PQ}$ in Frame (Q).
Let $\mathbf{R_Q} = \begin{pmatrix}\mathbf{T} & \mathbf{N} & \mathbf{B}\end{pmatrix}$ be the rotation from (Q) to (I) and $\mathbf{R} = \begin{pmatrix}\hat{\mathbf{u}} & \hat{\mathbf{v}} & \hat{\mathbf{w}}\end{pmatrix}$ the orientation of (P).
Deriving with respect to the inertial frame (I) yields:

$$\begin{aligned}
\frac{d\mathbf{p}}{dt} &= \frac{d\mathbf{q}}{dt} + \frac{d(\mathbf{R_Q} \times \mathbf{e})}{dt} \\
\iff \frac{d\mathbf{p}}{dt} &= \frac{d\mathbf{q}}{dt} + \frac{d\mathbf{R_Q}}{dt}\mathbf{e} + \mathbf{R_Q}\dot{\mathbf{e}} \\
\iff \mathbf{R_Q}^T\frac{d\mathbf{p}}{dt} &= \mathbf{R_Q}^T\frac{d\mathbf{q}}{dt} + \mathbf{R_Q}^T\frac{d\mathbf{R_Q}}{dt}\mathbf{e} + \dot{\mathbf{e}} \\
\iff \mathbf{R_Q}^T\mathbf{V} &= \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix}^T + \omega_{\mathbf{Q}} \wedge \mathbf{e} + \dot{\mathbf{e}} \\
\iff \mathbf{V_Q} &= \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix}^T + \omega_{\mathbf{Q}} \wedge \mathbf{e} + \dot{\mathbf{e}} \\
\iff \dot{\mathbf{e}} &= \mathbf{V_Q} - \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix}^T - \omega_{\mathbf{Q}} \wedge \mathbf{e}
\end{aligned} \tag{2}$$

The quantity $\omega_{\mathbf{Q}} = \mathbf{R_Q}^T\frac{d\mathbf{R_Q}}{dt}$ represents the angular rotation rate of Frame (Q) with respect to (I) expressed in (Q). $\mathbf{V_Q} = \mathbf{R_Q}^T\mathbf{V}$ is the velocity of the vehicle expressed in (Q). Moreover we can develop $\omega_{\mathbf{Q}}\wedge$ as follows:

$$\begin{aligned}
&\omega_{\mathbf{Q}}\wedge = \mathbf{R_Q}^T\frac{d\mathbf{R_Q}}{dt} \\
&\omega_{\mathbf{Q}}\wedge = \begin{pmatrix}T & N & B\end{pmatrix}^T \times \begin{pmatrix}\frac{dT}{dt} & \frac{dN}{dt} & \frac{dB}{dt}\end{pmatrix} \\
&\omega_{\mathbf{Q}}\wedge = \frac{ds}{dt}\begin{pmatrix}T & N & B\end{pmatrix}^T \times \begin{pmatrix}\frac{dT}{ds} & \frac{dN}{ds} & \frac{dB}{ds}\end{pmatrix} \\
&\omega_{\mathbf{Q}}\wedge = \dot{s}\begin{pmatrix}T & N & B\end{pmatrix}^T \times \begin{pmatrix}T & N & B\end{pmatrix} \times \mathbf{Q}^T \\
&\omega_{\mathbf{Q}}\wedge = \dot{s}\mathbf{I}_3 \times \mathbf{Q}^T \\
&\omega_{\mathbf{Q}}\wedge = -\dot{s}\mathbf{Q} \\
&\omega_{\mathbf{Q}}\wedge = -\dot{s}\begin{bmatrix}0 & k_1 & k_2 \\ -k_1 & 0 & 0 \\ -k_2 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0\end{bmatrix}
\end{aligned} \tag{3}$$

Therefore we can write $\omega_{\mathbf{Q}} = \dot{s}\begin{pmatrix}0 & -k_2 & k_1\end{pmatrix}^T$.

Combining this information with Equation (3), the resultant equation governing the error can be formulated as follows:

$$\dot{\mathbf{e}} = \mathbf{V_Q} - \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix}^T - \omega_{\mathbf{Q}} \wedge \mathbf{e} \tag{4}$$

2.3 Mathematical convergence of the controller

The following notations refer to the system above. Therefore, let us suppose that we have a holonomic system. This means that $\mathbf{V}$ can be controlled independently of the orientation. Furthermore, a model for the dynamics of the vehicle is required.
We will assume the system follows a general model for a vehicle moving in 3D space as follows:

$$\dot{\mathbf{V}}_r = \frac{1}{m}\mathbf{f_r} - \omega_{\mathbf{r}} \wedge \mathbf{V_r} \tag{5}$$

$\mathbf{V_r}$ and $\omega_{\mathbf{r}}$ respectively represent the linear and angular velocities expressed in the body frame (P).
$\mathbf{f_r}$ expresses the forces that are applied on the system. Those forces could, for instance, be the thrust of the motors. Given the holonomic nature of the system, we can presume that we directly control $\dot{\mathbf{V}}_r$ by the simple fact that $\mathbf{f_r}$ can take any vector value and the relation is fully invertible.

Subsequently, the intention is to build a function and derive it such that the variables of control appear. Hence, they could be selected in a manner that ensures compliance with the Lyapunov theorem. In practice, using the norm of a vector squared as a Lyapunov function usually yields good results. Therefore, let us define a potential Lyapunov candidate function and derive it as follows:

$$\begin{aligned}
&\mathcal{L}(t) = \frac{1}{2}\|\mathbf{e}(t)\|^2 = \frac{1}{2}\mathbf{e}^T \times \mathbf{e} \\
\Longrightarrow\ &\dot{\mathcal{L}}(t) = \dot{\mathbf{e}}^T \times \mathbf{e} \\
\Longleftrightarrow\ &\dot{\mathcal{L}}(t) = (\mathbf{V_Q} - \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix}^T - \omega_{\mathbf{Q}} \wedge \mathbf{e})^T \times \mathbf{e} \\
\Longleftrightarrow\ &\dot{\mathcal{L}}(t) = \mathbf{V_Q^T} \times \mathbf{e} - \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix} \times \mathbf{e} \\
\Longleftrightarrow\ &\dot{\mathcal{L}}(t) = \begin{pmatrix}u_{\mathbf{Q}} & v_{\mathbf{Q}} & w_{\mathbf{Q}}\end{pmatrix} \times \mathbf{e} - \begin{pmatrix}\dot{s} & 0 & 0\end{pmatrix} \times \mathbf{e} \\
\Longleftrightarrow\ &\dot{\mathcal{L}}(t) = \begin{pmatrix}u_{\mathbf{Q}} - \dot{s} & v_{\mathbf{Q}} & w_{\mathbf{Q}}\end{pmatrix} \times \mathbf{e}
\end{aligned} \tag{6}$$

Unfortunately, $\mathcal{L}$ cannot be a Lyapunov function because it is enough that $\mathbf{V_Q} = e$ and $\dot{s} = 0$ to have $\dot{\mathcal{L}} = 0$ and the state of the system not being the null state.
Nevertheless, let us assume for an instant that the system was controlled by its velocity and not its acceleration. In (6), let $v_{\mathbf{Q}} = -k \times e_2, w_{\mathbf{Q}} = -k \times e_3, \dot{s} = u_{\mathbf{Q}} + k \times \mathbf{e_1}$ $(k \in R_+)$ and $u_{\mathbf{Q}}$ free.
This would yield:

$$\dot{\mathcal{L}}(t) = -k\|e\|^2 \leq 0$$

It is a well-known fact that if a function asymptotically converges to 0 as $t \longrightarrow \infty$, its derivative does not necessarily to 0 and the reverse is also true. Nevertheless, the technique used in [2] can be employed, and therefore the following Lemma can be used to prove the convergence of the system:

Barbalat's Lemma

If $f : \mathbb{R} \longrightarrow \mathbb{R}$ is differentiable twice, f' is uniformly continuous function and $\underset{t\to\infty}{lim} f(t) \in \mathbb{R}$,
Then $\underset{t\to\infty}{lim} f'(t) = 0$.

For a function to be uniformly continuous, it is sufficient that its derivative is bounded.
It is easy to see that $\ddot{\mathcal{L}}$ is bounded because $\dot{\mathcal{L}}(t) = -k\|e\|^2 = -k.\mathcal{L}(t) \implies \ddot{\mathcal{L}} = -k.\dot{\mathcal{L}}$. Therefore, $\dot{\mathcal{L}}(t) = \mathcal{L}(0)e^{-kt}$, which yields that $\ddot{\mathcal{L}}$ is bounded. Thus $\dot{\mathcal{L}}$ is uniformly continuous.
Moreover, the convergence of $\mathcal{L}$ is assured because it is a positive function with a negative derivative.
Barbalat's theorem exhibits that $\dot{\mathcal{L}}(t) \underset{t\to\infty}{\longrightarrow} 0$, i.e., $\mathbf{e} \underset{t\to\infty}{\longrightarrow} 0$.
(Note that here Barbalat's lemma was not necessary to prove that $\dot{\mathcal{L}}$ goes to 0, due to its expression, but this is a showcase of the theorem).
Consequently, all solutions to (4) converge to 0 as time approaches infinity. In simpler terms, the error diminishes to zero, resulting in the vehicle's convergence to the intended path.
This technique is powerful because it not only provides a proof of convergence but also yields the control law which might not necessarily be linear.

Regrettably, in our case, the system's control lies in its acceleration rather than its velocity. However, the silver lining will be Lasalle's principle, which will be utilized to address the dynamic case. Put simply, Lasalle's theorem will provide the mathematical substantiation that having an acceleration that asymptotically drives the velocity of the vehicle towards the desired velocity ($\mathbf{V_Q}$ found previously), the error still converges to zero. Let us provide some definitions and properties necessary to state Lasalle's theorem.

Definition (Positive invariant set):

Suppose $\dot{x} = f(x)$ is an autonomous system, $x(t, x_0)$ is a trajectory, and x_0 is the initial point.
Let $\mathcal{O} = \{x \in \mathbb{R}^n \mid \varphi(x) = 0\}$ where φ is a real-valued function.
The set $\mathcal{O}$ is said to be positively invariant if $x_0 \in \mathcal{O}$ implies that $\forall\ t \geq 0,\ x(t, x_0) \in \mathcal{O}$.
Put simply, if the start of a trajectory resides in $\mathcal{O}$, it will remain in $\mathcal{O}$.

Theorem (LaSalle's invariance principle):

Let Ω be a positively invariant set of the autonomous system. Suppose that every solution starting in Ω converges to a set $E \subset \Omega$ and let M be the largest invariant set contained in E. Then every bounded solution starting in Ω converges to M as $t \longrightarrow \infty$.

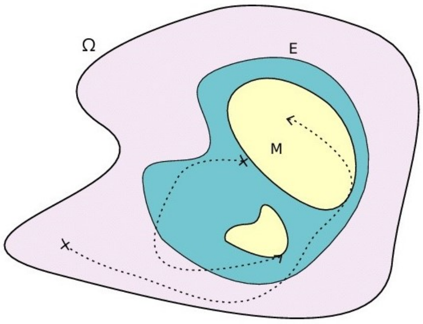


Figure 6: An illustrative overview of Lasalle's theorem

Consequently, let $\mathbf{x}$ denote the state of the system. Let $\Omega = \mathbb{R}^n$, with n being the dimension of the state space.
Let E denote the set $\{\mathbf{x} \in \Omega / \begin{pmatrix}\dot{s} & v_{\mathbf{Q}} & w_{\mathbf{Q}}\end{pmatrix} = \begin{pmatrix}u_{\mathbf{Q}} + ke_1 & -ke_2 & -ke_3\end{pmatrix}\}$.
This set corresponds to the trajectories possessing the exact desired velocity. Above, it was shown that trajectories included in this set converge towards the set $M = \{\mathbf{x} \in \Omega / \mathbf{e} = 0\}$. Therefore, M is the largest invariant set of E (if the error is null, the expression chosen for the velocity assures that the error stays null). Lasalle's theorem implies that every trajectory starting in Ω will converge to the set M. In simple terms, every trajectory is such that we convege to the path.
We chose as a control law $(\dot{s}, v_{\mathbf{Q}}, w_{\mathbf{Q}}) = (u_{\mathbf{Q}} + ke_1, -ke_2, -ke_3)$.
In practice the control law $(\dot{s}, v_{\mathbf{Q}}, w_{\mathbf{Q}}) = (u_{\mathbf{Q}} + \nu_d.th(\frac{e_1}{d_0}), -\nu_d.th(\frac{e_2}{d_0}), -\nu_d.th(\frac{e_3}{d_0}))$ is better suited because it is bounded and it allows to control the approach velocity to the path ($\nu_d > 0$) as well as the aggressiveness ($d_0 > 0$ is a characteristic distance). To find the necessary accelerations, the previous expressions can be derived and used as control desired values for a PD controller.
Using the same approach and with simplicity, it can be demonstrated that this control law also drives the system's error to converge to zero.

2.4 Obstacle Avoidance Algorithm

2.4.1 The different algorithms

During the course of this internship, an additional task was requested, which involved the development of a local obstacle avoidance (OA) system alongside the preexisting path following algorithm. This supplementary task entailed a process encompassing research, implementation, simulation validation, and real-world experimentation within a limited time frame of one and a half months.

In this research phase, where a thorough search into the intricacies of obstacle avoidance algorithms was conducted, the following results [3][4] were found: the most frequently used algorithms are, depending on the context and need, A*, Rapidly-exploring Random Trees (RRT), RRT*, and RRT* Connect. These algorithms were evaluated in terms of their suitability for addressing the task of local obstacle avoidance within the given context as well as the given time constraint. Subsequently, we will be proceeding with the implementation of the RRT algorithm. The choice was determined by a main factor: simplicity.

3 Implementation

3.1 Results of the first tests

The preceding control law was implemented in Python and evaluated within the Gazebo simulation environment. It's important to note that the model and drone employed in the simulation differ from the practical drone. An intern was engaged in developing the simulation for the actual drone. Consequently, a more lightweight and agile quadcopter drone was utilized in the simulation, which means that upon practical implementation, certain major adjustments to the gains may be necessary.
Nevertheless, the control law was tested on different tracks as such:

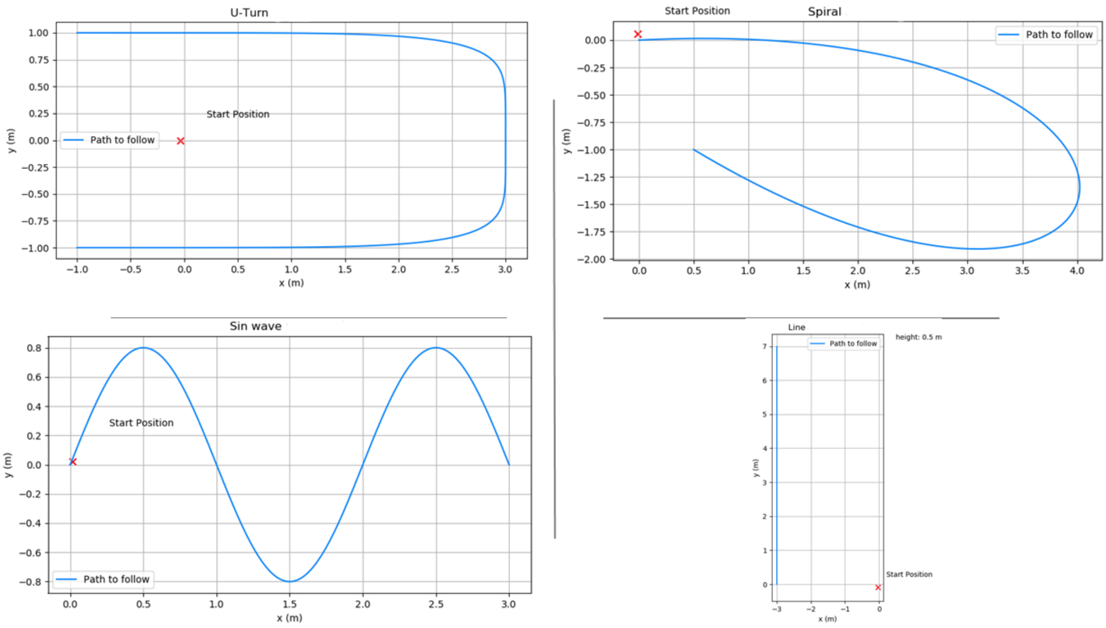


Figure 7: Test tracks

Several flight trajectories were devised to evaluate the drone's performance under diverse conditions. The selected trajectories include a U-turn, a sinusoidal wave, a spiral, and a straight-line path.

The straight-line trajectory was incorporated as a fundamental reference, serving as a baseline for assessing basic flight capabilities. Its simplicity allows for a clear examination of the drone's stability and tracking accuracy.

The choice of a U-turn trajectory is substantiated by its relevance to indoor environments, where the drone frequently executes such turns. Additionally, u-turns exhibit pronounced curvature, making them suitable for assessing the drone's agility and maneuverability.

The adoption of a spiral trajectory derives from its real-world applicability, as spiraling flight paths are commonly encountered. This trajectory enables the evaluation of the drone's ability to navigate complex paths with increasing curvature.

Finally, the sinusoidal wave trajectory was included to challenge the drone's performance under conditions of significant variations in the flight path. This choice aims to gauge the drone's response to intricate and dynamically changing tracks.

And here are the results:

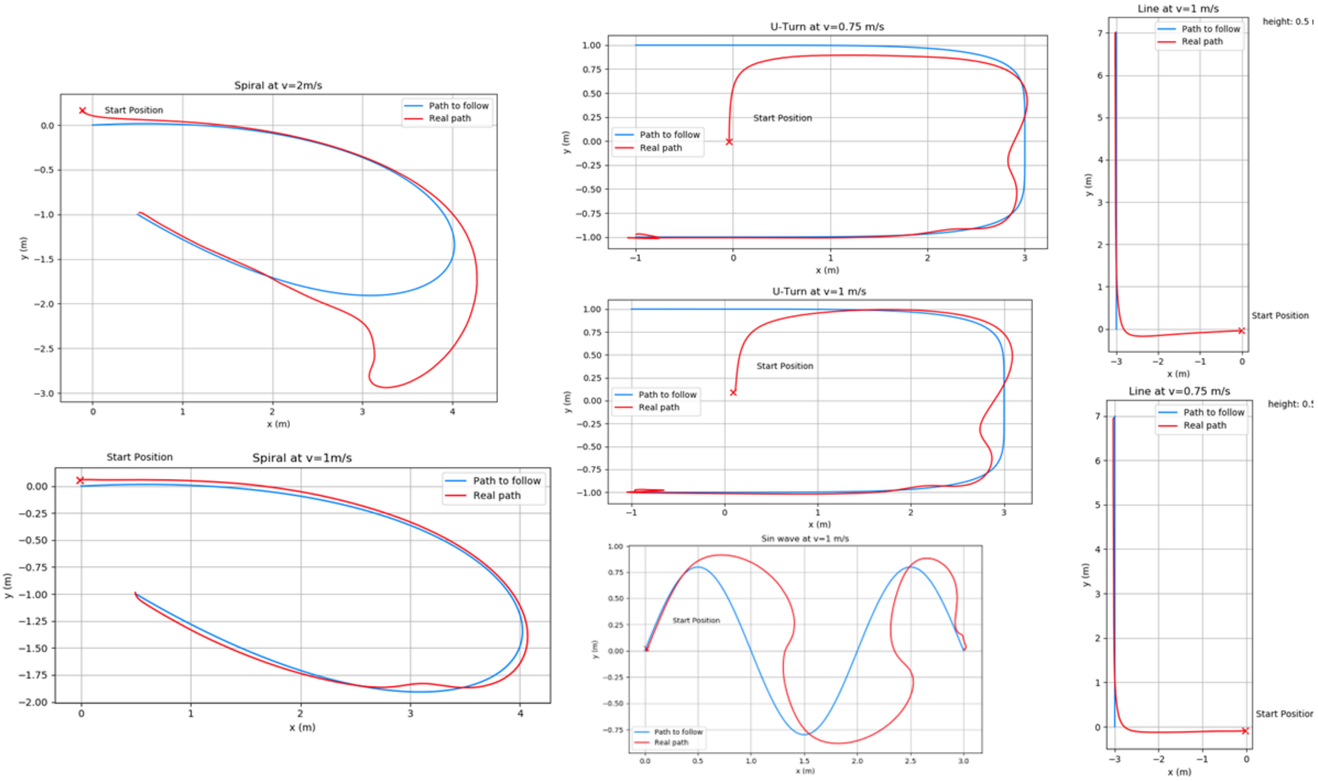
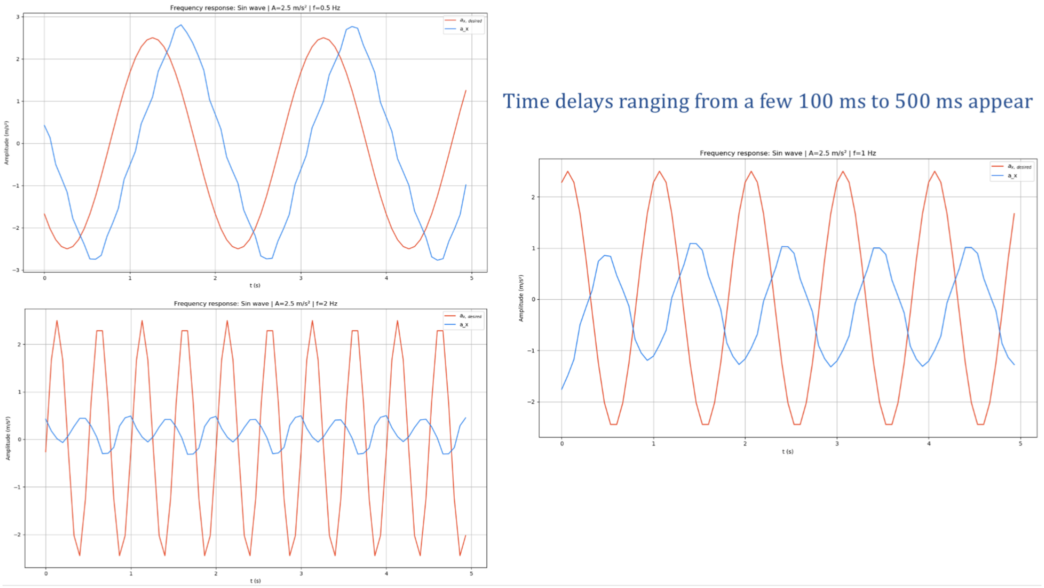


Figure 8: Results of the simulation

3.2 Analysis & explanation of the results

The results highlight two important problems linked to the controller. To begin with, the UAV overshoots at high speeds in curved turns. Given that the convergence of the algorithms was proven, the problem could only come from the commands. In the implementation, the acceleration command was bounded to 3 m/s^2 due to safety issues. However, the drone has a measured maximum acceleration of $4.5m/s^2$. Nevertheless, in a turn, the centrifugal acceleration is equal to $a_c = \kappa v^2$ with κ the curvature of the turn and v the speed of the vehicle. The spiral path has for instance a maximum curvature of 3.16 m^{-1}. A swift computation shows that with $\kappa = 3.16\ m^{-1}$ and $v = 1\ m/s$, $a_c = 1 \times 3.16 \leq 4.5\ m/s^2$.
Therefore, a frequency response test was performed on the acceleration. Since the drone is symmetric, exciting in with a sin wave in any direction in the (Oxy) plane should yield the same result. Thus, the benchmark to test the frequency response is composed of two inquiries: one is the frequency response in the (Ox) direction and the second in (Oz).
Hence are the plots of the desired command and the real command extracted from the IMU:



Time delays ranging from a few 100 ms to 500 ms appear

Figure 9: Desired commands (red) and actual $a_x$command (blue)

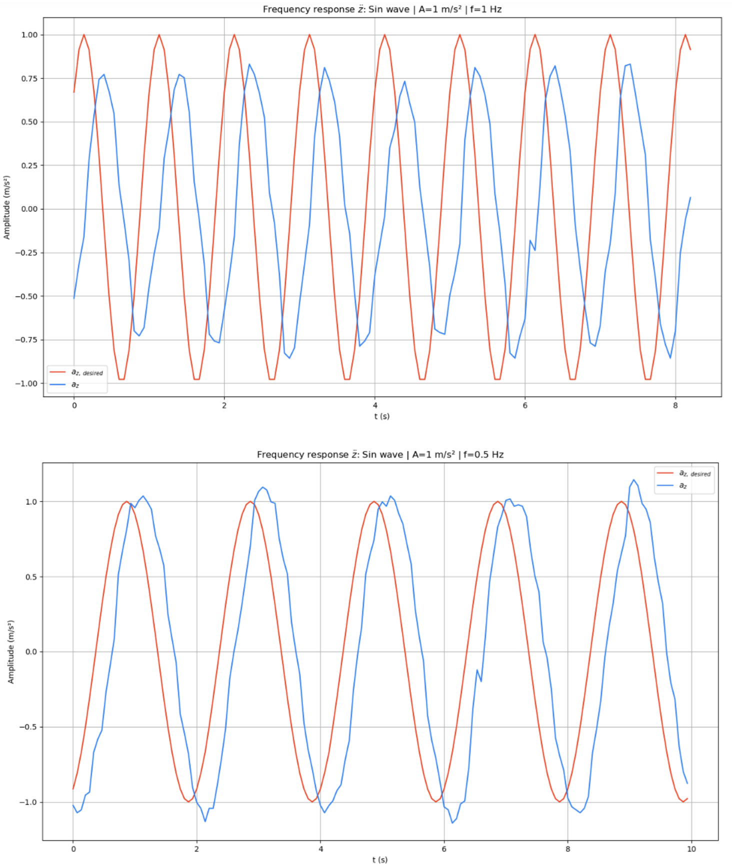


Figure 10: Desired commands (red) and actual $a_z$command (blue)

Primarily, the response in the (Oz) plane seems ordinary. However, in the (Ox) plane, a distinct delay becomes apparent, particularly at higher frequencies.

This phenomenon has a dual explanation. Initially, when a command is transmitted to the Flight Control Unit (FCU), an internal control command is employed to steer the UAV in the required direction, simultaneously aligning a slight portion of its thrust with the intended course. This is likely executed through a PID-based control law. Consequently, the accelerations $\ddot{x}$ and $\ddot{y}$ are artificially governed, while solely $\ddot{z}$ is genuinely controlled. This predicament is rooted in the fact that the drone is maneuvered using high-level commands, a characteristic intrinsic to MAVROS.

The second explanation stems from physical considerations. The UAV solely generates thrust along one axis, namely the vertical direction. For instance, when an acceleration in the forward direction is requested, the UAV needs to rotate and orient itself forward, introducing a time delay. This delay is negligible for minor accelerations, yet for sharp and swift turns, the drone must transition swiftly from nearly zero acceleration to a significant one. For instance, envision being oriented left and having to promptly switch to a rightward direction—a transition that cannot occur instantaneously, resulting in a delay τ. Consequently, the drone overshoots during turns, as it needs time to align its thrust with the new direction. An illustration of this phenomenon is presented below:

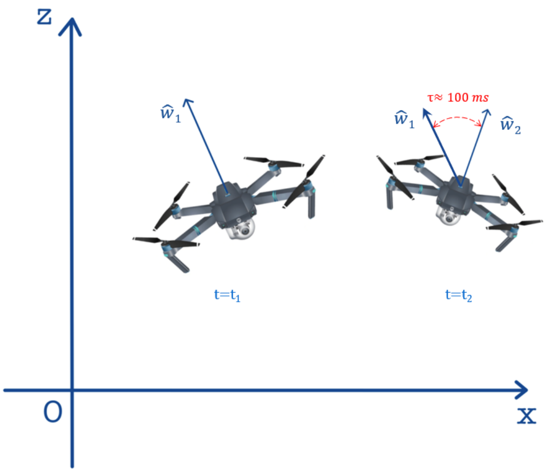


Figure 11: Drone changing the direction of thrust

A potential resolution would involve controlling the drone using both its thrust and angular acceleration. Formulating a Lyapunov-based control law for this purpose would present a formidable challenge. A practical approach could entail assuming control of the drone via its linear accelerations and angular velocities, given its high responsiveness in achieving the desired angular velocity. An analytical solution might therefore be envisioned [5].

This alternative involves controlling the drone's thrust and angular velocity. Such a topic exists in MAVROS. However, a constraint exists on the thrust's range—bounded between 0 and 1—where 0 signifies no thrust, 0.5 corresponds to hovering, and 1 signifies ”maximum thrust.” A difficulty arises due to the fact that this designated ”maximum thrust” value does not align with the simulation's actual maximum thrust. Efforts were made to ascertain this value's adjustment but proved unsuccessful. Although such discrepancies might not pose an issue on the actual drone, the limitation emerged during simulation. Notably, due to spatial constraints within the university's laboratory, it was unfeasible to test the MAVROS topic's maximum thrust in real-world conditions.
Consequently, the only feasible solution was to control the drone using ”artificial accelerations.”

3.3 Solution to deal with overshooting & delay

So as to solve the UAV's overshoots, it is necessary to look ahead to prepare for highly curved turns. The UAV has the necessary thrust to overcome a given turn in the test tracks, but it must first align the axis of thrust in time, before arriving in the previously mentioned corner. For instance, if the drone has to make a U-turn, it would suffice that it changes orientation slightly from neutral to slightly tilted backward a few moments before the U-turn comes, as depicted here:

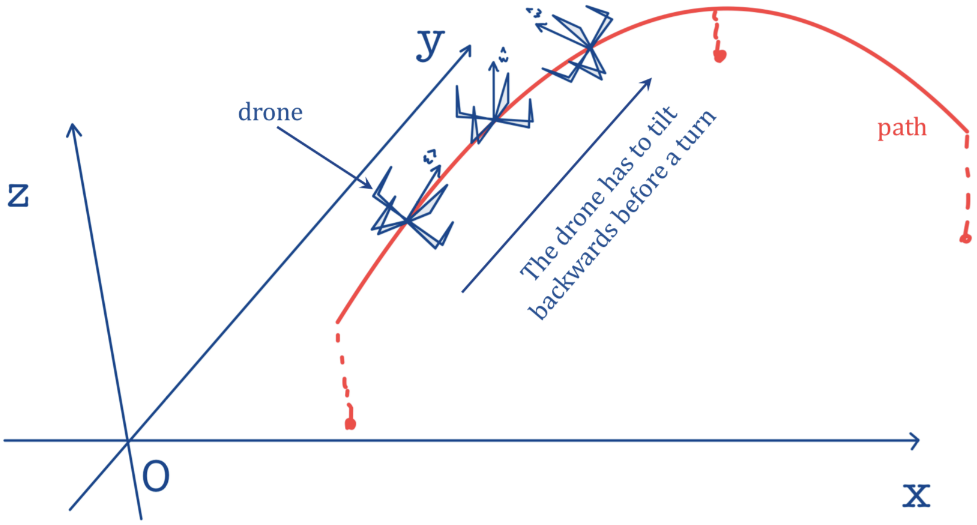


Figure 12: Drone preparing in advance to take a highly curved turn

Still, how would we achieve to change the tilt of the drone when the yaw is the sole orientation variable accessible using the MAVROS acceleration topic? After careful observation, it was noticed that when the drone slows down it tilts backward. Therefore, would could add a term to our forward acceleration that governs the UAV to slow down when the curvature and the speed are high. This term could then be bound for stability issues.

Therefore, let this term be denoted by t. Multiple formulas were tested empirically. The simplest form of expression amounting to a multiplication was of course tested: $t = -k.\kappa.v$ ($k \in \mathbb{R}_+^*$).

After careful consideration, analysis of tests showed that a term that is null for small curvatures and grows rapidly with a given threshold when the curvature is high is required. This should remind us of a quadratic expression "$y = (\frac{x}{x_0})^n$". If n has a high value, then, x values between 0 and x_0 get attenuated substantially and those greater than x_0 amplified significantly:

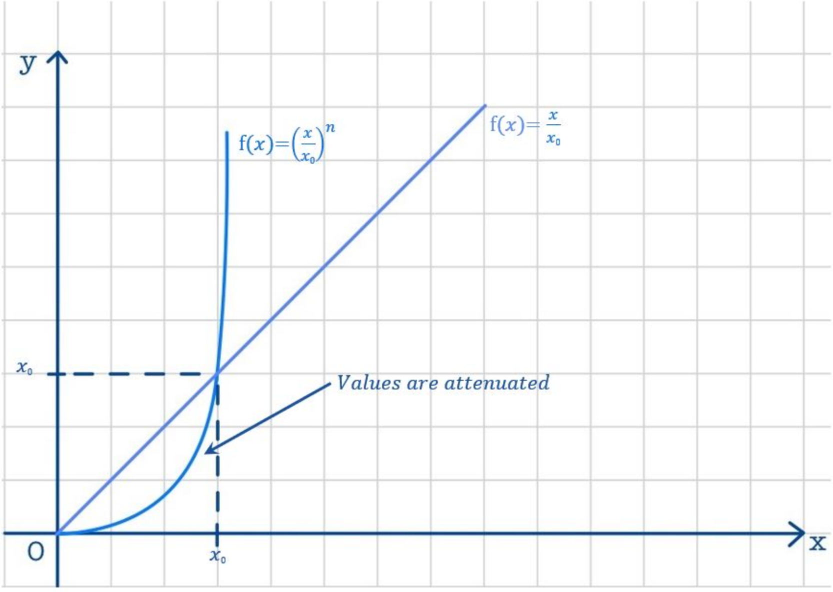


Figure 13: Quadratic curve

Given that the centrifugal acceleration is $a_c = \kappa.v^2$, it hints that a natural form of expression to test is $t = -k.\kappa.v^2$. In practice, when the curvature was overly high, the drone would stop almost completely with this expression. Therefore, a saturation was required on the curvature with a given threshold as such: $t = -k.tanh(\frac{\kappa}{\kappa_0}).v^2$. We also want this term to not affect the drone when it is far away from the path. Thus $t = -k.tanh(\frac{\kappa}{\kappa_0}).v^2.(1 + \frac{d_{path}}{d_0})^{-1}$. The coefficients were found empirically and the final expression of t is therefore:

$$t = -k.tanh(\frac{\kappa}{\kappa_0}).v^2.(1 + \frac{d_{path}}{d_0})^{-1}, k = 5, \kappa_0 = 6\ m^{-1}, d_0 = 1\ m \tag{7}$$

The incorporation of this term has effectively eliminated the issue of overshooting. However, there is still a minor inconvenience that requires our attention. Indeed, at very high speeds the drone overshoots. This comes from physical limitations. Indeed, the drone has a maximum thrust. In order to avoid such scenarios we need to limit the speed of the drone to a threshold given by the curvature and the maximum acceleration the drone is capable of producing. In turns, the driving force comes from the centrifugal acceleration as $a_c = \kappa.v^2$. The drone has a maximum acceleration a_{max}. Thus the velocity needs to satisfy the following:

$$v \leq q.\sqrt{\frac{a_{max}}{\kappa}} \tag{8}$$

q is a safety factor. In practice $q = 0.7$ (30% of margin).

Subsequent sections will detail the outcomes of the tests conducted in both simulation and on the actual drone.

3.4 Final results in simulation

Hence are the final results after the implementation of the previous solutions:

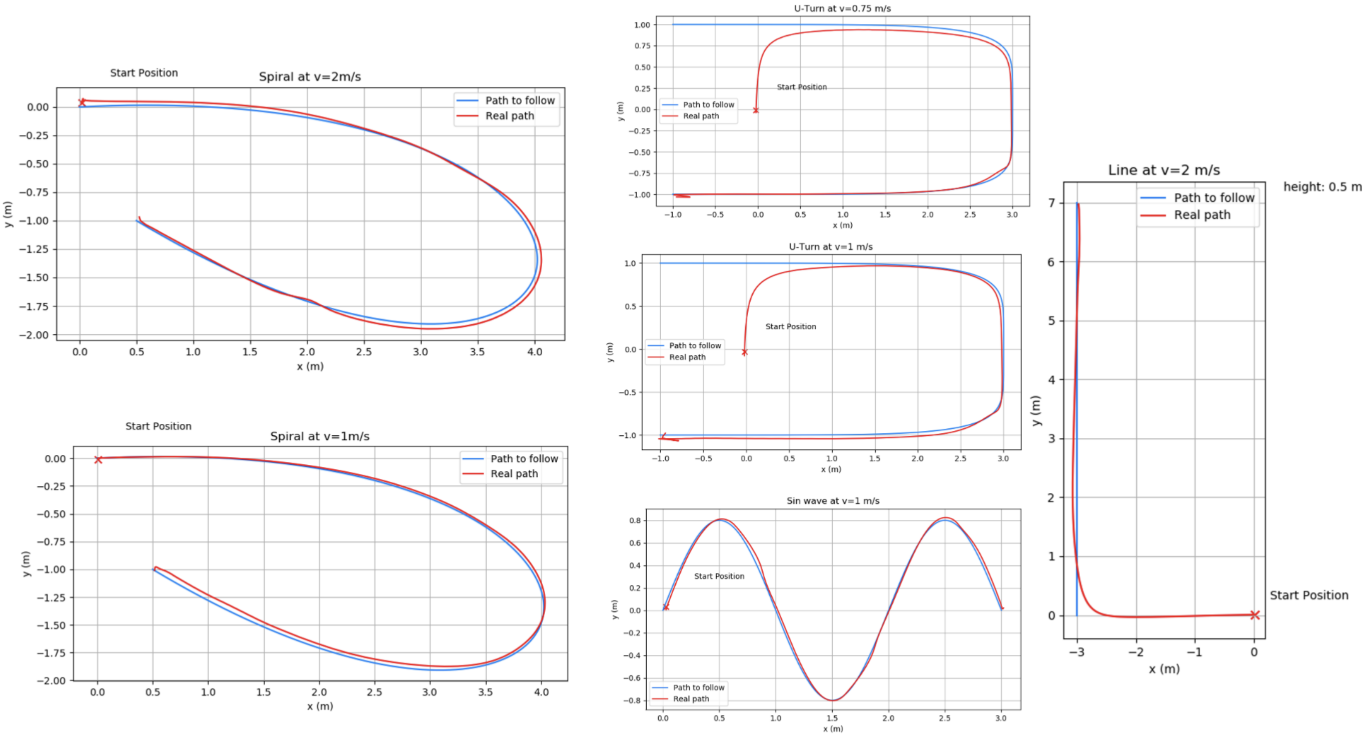


Figure 14: Drone changing the direction of thrust

We employ the infinity norm to quantify the error, which is determined by the difference between the current position and the point at the curvilinear parameter s along the path. Consequently, as the Euclidean sphere 2 (unit sphere) is encompassed within the infinity sphere, the actual real error is smaller. The ensuing error graphs illustrate the aforementioned tracks:

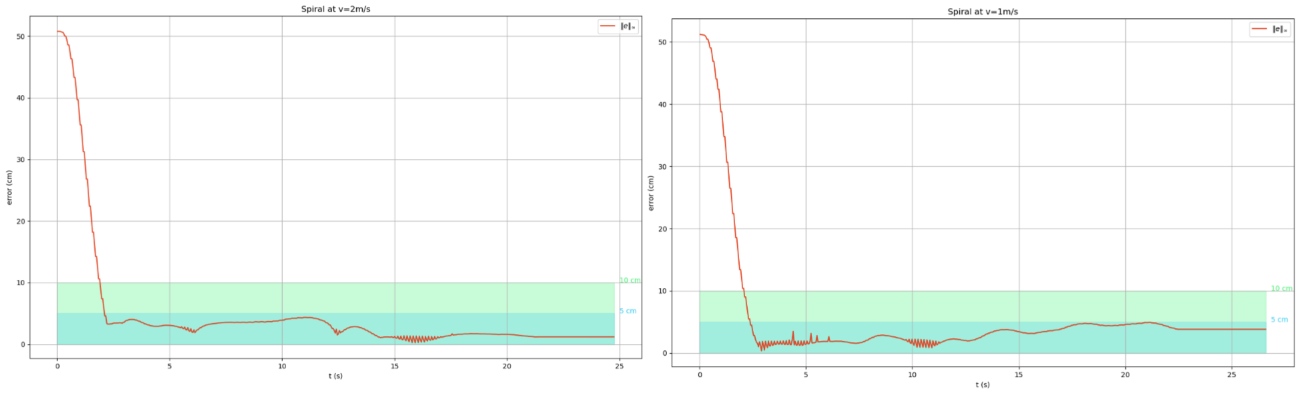


Figure 15: $\|.\|_\infty$ of spiral

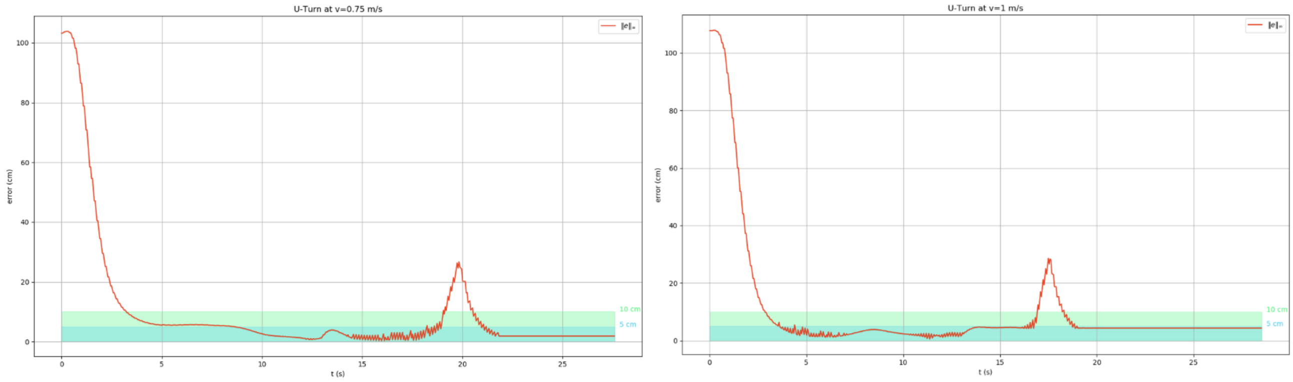


Figure 16: $\|.\|_\infty$ of u-turn

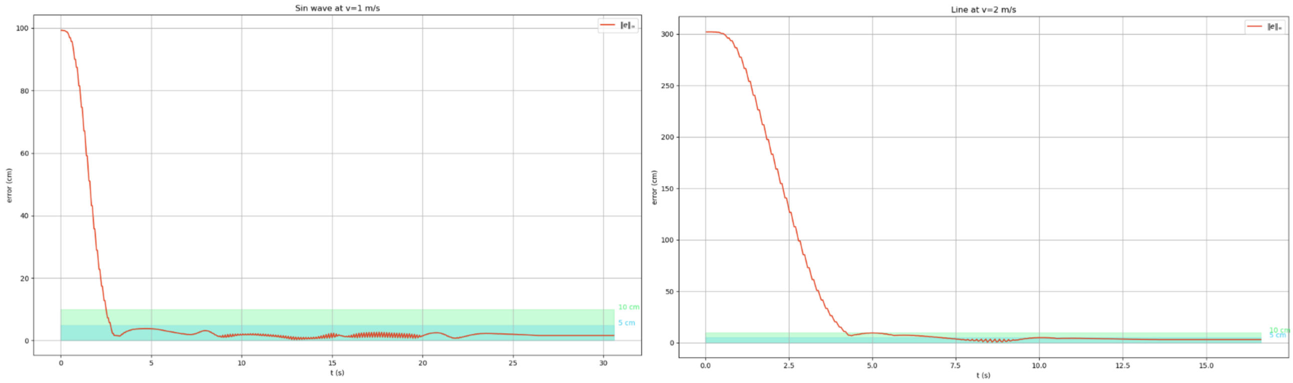


Figure 17: $\|.\|_\infty$ of line and sine wave

The u-turn trajectory exhibits a notable error spike at approximately $t \simeq 19\ s$. This spike, however, is a false alert, arising from the registration of rosbags that introduce a delay in the control loop update. However, it is worth highlighting that while the error stays below the required $10\ cm$, once it enters this region (light green zone), it can be observed that t is predominantly confined within the range of under $5\ cm$ (blue-green zone) on most occasions.

Nevertheless, it is worth wondering about the stability of this controller. The added term is empirical and there is no guarantee that divergences might not occur. The only guarantee is that in testing the common scenarios, we did not encounter any issues.

3.5 Tests and results on the real drone

The real UAV at our disposal is not the most efficient system. It has a considerable size and emits a substantial volume of noise, not to mention that the spinning blades produce strong winds. It consumes around $1500W$ (as much as an oven).
Testing this system is thus very dangerous. Taking into account the reduced size of the laboratory, we were not able to test it while registering rosbags. Indeed, if lag emerges, the drone might remain stuck on the last commands for a long period and crash into a wall.
Fortunately, we did two measurement campaigns: one in Mosbach (South of Germany) and one in Vienna. Having substantially more space, in the unfortunate case of an issue, the remote might be acted to manually fly the UAV.
Moreover, due to time constraints as well, we were only able to test straight lines in Mosbach. Depicted below are the results of the aforementioned tests:

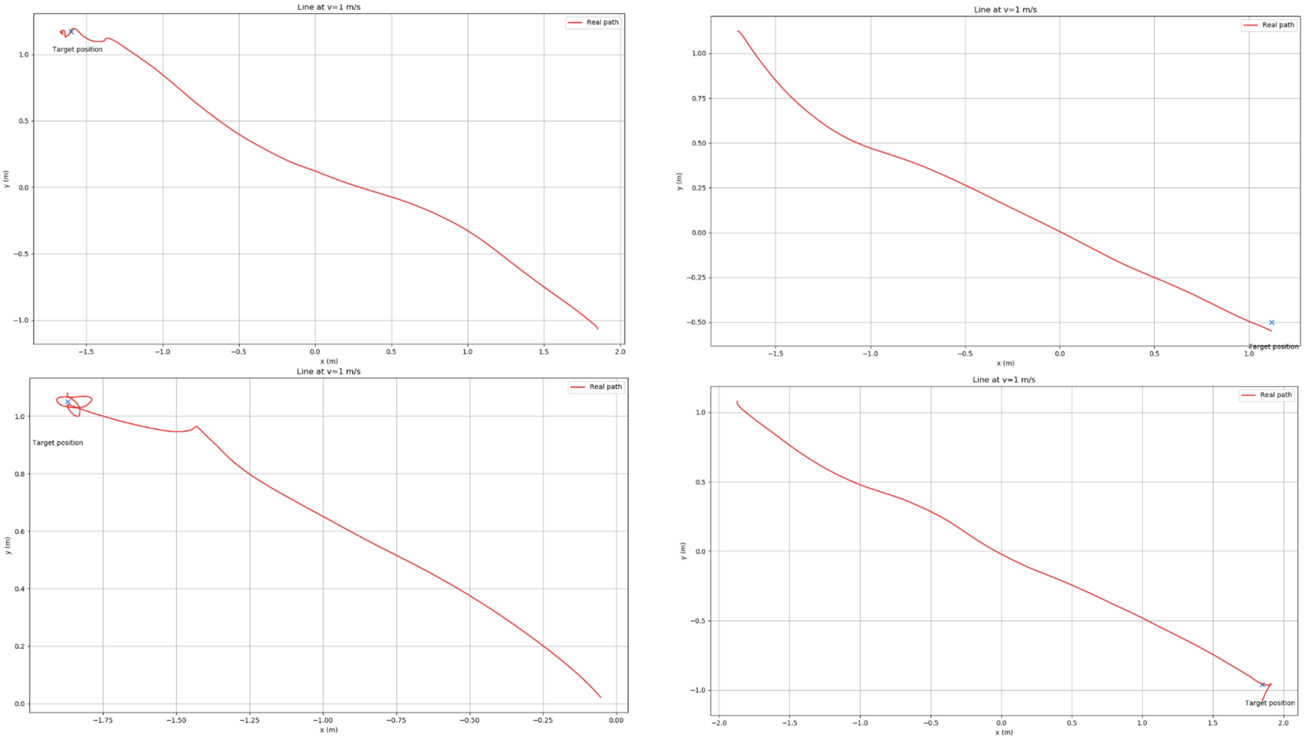


Figure 18: Straight lines test on the real drone

Those tests allowed us nevertheless to adjust the real gains. When implemented on the actual drone, the gains were approximately halved.

3.6 Obstacle Avoidance

Following the research stage, the implementation phase was initiated. Rigorous testing was executed through simulation scenarios, aiming to validate the functionality and efficacy of the local obstacle avoidance system. RRT was implemented in Python and was heavily optimized by vectorization of the operations, allowing the use of numpy arrays, and therefore faster computation.

Nevertheless, two problems arose: the algorithm did not generate optimal enough paths within a minor number of iterations and was slightly slow even after optimization. The paths exhibited were uneven and erratic trajectories, and therefore required a smoothing process. A promising process for path smoothing was found in [6]. This algorithm alone was insufficient because it was required to reduce the length of the paths to an acceptable size as well. Therefore, a hybrid model was deployed combining [6] and another process. The algorithm starts by erasing points that make the path erratic. Thereafter, it also adds new points halfway between existing points. Ultimately, it tries to tie those new points to neighbors resulting in a shorter total path length. The algorithm has one advantage and one inconvenient:

The problem arises from the fact that some scenarios get the algorithm to be stuck in a fixed point configuration. After reaching this stage, new iterations do not change the path.

The advantage is that this fixed point configuration happens solely when the path is getting close to the shortest path. For paths that are extremely large, plots showed that the length of the path diminishes exponentially with the number of iterations up to a threshold which occurs when getting close to the minimal length. Therefore, RRT can generate very jagged and long paths for a minor number of iterations which would then be smoothened with this process that is very efficient in the first iterations.

However, due to constraints in terms of time and resources, the final phases of the project, involving testing in both simulation and real-world environments the OA algorithm, were not fully validated and require further experimentation.

3.7 Conclusion

The finality of the project is the demonstration at the military base in Vienna. The path following algorithm was implemented and respected the stringent constraint of $\leq 10\ cm$ of error.

The scenario was the following: the drone starts at the entrance of the building and then takes off. It scans and sends feedback to a control station. The control station sees the environment and generates obstacle-free paths that the drone has to follow. The drone follows them and in the exploration phase, it encounters victims. Once it is done exploring, it lands.

The demonstration was a success, but due to confidentiality issues, it cannot be shown.

4 Second phase: building a rigid-body transform minimal contractor for SLAM purposes

The last 3 weeks of the internship were directed towards an active field of research: interval analysis. More specifically, the task at hand is the following:

We register with a depth camera a point cloud from two different perspectives. Can we use interval analysis to bound (in a guaranteed manner) and contract (minimally) the transformation tying those two point clouds?

Another scenario is SLAM. between two consecutive measurement frames, a vehicle could see the same point cloud from different perspectives. Using an algorithm like SIFT or SURF might allow us to associate those points. Therefore, finding the transformation will allow it to find and update its current pose. This problem was already encountered and a solution can be found in [7].

After exchanging with my supervisor *Aaronkumar Ehambram*, the focus was shifted towards the rotation because it was the variable not getting contracted minimally.

Time constraints were very present and therefore efficiency and organization were necessary. After exchanging and brainstorming, we came to the hypothesis that the problem might be translatable into a 3D polar constraint. Therefore, if a minimal contractor can be built for the 3D case then the problem would be solved.

4.1 The polar equation

The polar equation can be expressed as such:

$$X = R \begin{pmatrix} \rho \\ 0 \\ 0 \end{pmatrix} \tag{9}$$

$$R \in SO(n),\ X \in \mathbb{R}^n,\ \rho \in \mathbb{R}^{+*}$$

When $n = 2$, it amounts to finding the angle θ as such:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \rho \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} \mid x, y, \rho, \theta \in \mathbb{R} \tag{10}$$

We want to solve this problem using interval analysis. The 2D case was already solved in [8].

4.2 Minimal contractor for the 3D polar constraint

Using a similar approach as in [8], the 3D case could be solved. Indeed, the idea in [8] is to start from a restrained set on which a minimal contractor is available and thereafter extend it to a more general domain.

Let X be a point in space and let it be written in polar form. An illustration of the constraint is depicted below:

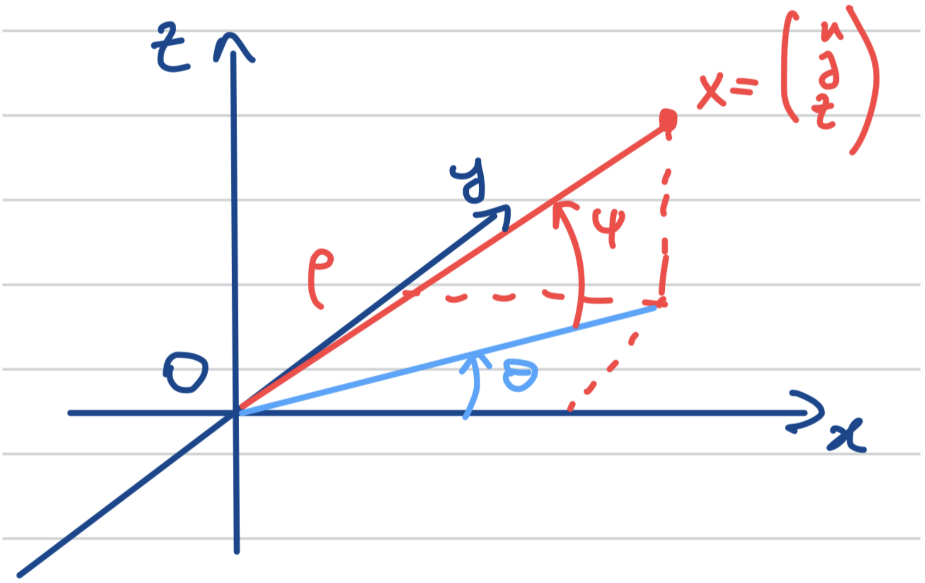


Figure 19: Polar equation in 3D

$$X = \rho \begin{pmatrix} \cos(\psi)\cos(\theta) \\ \cos(\psi)\sin(\theta) \\ \sin(\psi) \end{pmatrix}$$

$$X \in \mathbb{R}^3, \rho \in \mathbb{R}_+^*, \phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \theta \in [-\pi, \pi]$$

A comparable approach to the one utilized in the description of the 2D polar contractor will be used. The process will begin with a confined set denoted as Π_0, which will subsequently be expanded through box conservative operations. These operations primarily encompass symmetries and translations:

$$\Pi \hat{=} \left\{ (x, y, z, \rho, \theta, \psi) \in \mathbb{R}^6 / X = \mathbf{\Pi} \begin{pmatrix} \rho \\ \theta \\ \psi \end{pmatrix} \right\}$$

With $\mathbf{\Pi} \begin{pmatrix} \rho \\ \theta \\ \psi \end{pmatrix} = \rho \begin{pmatrix} \cos(\psi) cos(\theta) \\ cos(\psi) sin(\theta) \\ \sin(\psi) \end{pmatrix}$

And With $\mathbf{\Pi^{-1}} \begin{pmatrix} [\rho] \\ [\theta] \\ [\psi] \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 + y^2 + z^2} \\ arctan(\frac{y}{x}) \\ arctan(\frac{z}{\sqrt{x^2+y^2}}) \end{pmatrix}$

Let $\Pi_0 = [p_0] \cap \Pi$ with $[p_0] = \mathbb{R}^{+4} \times [0, \frac{\pi}{4}] \times [0, \frac{\pi}{2}]$

Theorem 1 (Minimal Contractor for Π_0):

The minimal contractor consistent with Π_0 is:

$$C_{\Pi_0} \begin{pmatrix} [x] \\ [y] \\ [z] \end{pmatrix} = \begin{pmatrix} [x] \times [y] \times [z] \cap \mathbf{\Pi} \begin{pmatrix} [\rho] \\ [\theta] \\ [\psi] \end{pmatrix} \\ [\rho] \times [\theta] \times [\psi] \cap \mathbf{\Pi^{-1}} \begin{pmatrix} [x] \\ [y] \\ [z] \end{pmatrix} \end{pmatrix}$$

Theorem 2 (Minimal Contractor for Π)

Let us define the following functions:

$$\begin{aligned}
\sigma_1 &: (x,y,z,\rho,\theta,\psi) \longmapsto \left(y,x,z,\rho,\frac{\pi}{2}-\theta,\psi\right) \\
\sigma_2 &: (x,y,z,\rho,\theta,\psi) \longmapsto (x,-y,z,\rho,-\theta,\psi) \\
\sigma_3 &: (x,y,z,\rho,\theta,\psi) \longmapsto (-x,y,z,\rho,\pi-\theta,\psi) \\
\sigma_4 &: (x,y,z,\rho,\theta,\psi) \longmapsto (x,y,-z,\rho,\theta,-\psi) \\
\sigma_5 &: (x,y,z,\rho,\theta,\psi) \longmapsto (x,y,z,-\rho,-\theta,-\psi) \\
\gamma_1 &: (x,y,z,\rho,\theta,\psi) \longmapsto (x,y,z,\rho,\theta+2\pi,\psi) \\
\gamma_2 &: (x,y,z,\rho,\theta,\psi) \longmapsto (x,y,z,\rho,\theta,\psi+\pi)
\end{aligned}$$

Therefore, the minimal contractor consistent with Π is:

$$C_\pi = \left[\bigsqcup_{j\in\mathbb{Z}} \gamma_2^j \circ \left(\bigsqcup_{i\in\mathbb{Z}} \gamma_1^i \circ \bigcirc_{1\leq k\leq 5}\left(I \sqcup \sigma_k\right)\right)\right](C_{\pi_0})$$

Proof:

The starting point will be the consistency of C_{π_0}:

$$\begin{aligned}
& C_{\pi_0} \sim \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \left[0,\tfrac{\pi}{4}\right] \times \left[0,\tfrac{\pi}{2}\right] \\
& \Rightarrow (I \sqcup \sigma_1)\,(C_{\pi_0}) \sim \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \left[0,\tfrac{\pi}{2}\right] \times \left[0,\tfrac{\pi}{2}\right] \\
& \Rightarrow (I \sqcup \sigma_2) \circ (I \sqcup \sigma_1)\,(C_{\pi_0}) \sim \mathbb{R}^+ \times \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}^+ \times \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right] \times \left[0,\tfrac{\pi}{2}\right] \\
& \Rightarrow \bigcirc_{1\leq k\leq 3}\,(I \sqcup \sigma_k)\,(C_{\pi_0}) \sim \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}^+ \times [-\pi,\pi] \times \left[0,\tfrac{\pi}{2}\right] \\
& \Rightarrow \bigcirc_{1\leq k\leq 4}\,(I \sqcup \sigma_k)\,(C_{\pi_0}) \sim \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \times [-\pi,\pi] \times \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right] \\
& \Rightarrow \bigcirc_{1\leq k\leq 5}\,(I \sqcup \sigma_k)\,(C_{\pi_0}) \sim \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times [-\pi,\pi] \times \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right] \\
& \Rightarrow \left[\textstyle\bigsqcup_{i\in\mathbb{Z}} \gamma_1^i \bigcirc_{1\leq i\leq 5}\left(I \sqcup \sigma_k\right)\right](C_{\pi_0}) \sim \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \left[-\tfrac{\pi}{2},\tfrac{\pi}{2}\right] \\
& \Rightarrow \left[\textstyle\bigsqcup_{j\in\mathbb{Z}} \gamma_2^j \circ \left(\bigsqcup_{i\in\mathbb{Z}} \gamma_1^i \circ \bigcirc_{1\leq k\leq 5}\left(I \sqcup \sigma_k\right)\right)\right](C_{\pi_0}) \sim \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}
\end{aligned}$$

The contractor is consistent with Π. The minimality comes from the fact that all functions and operations are box conservative.

5 Conclusion

In conclusion, to sum up my work: it involved the successful implementation of a path following algorithm using the Lyapunov method. The outcomes yielded an error rate consistently below 10 cm, with a majority of instances displaying an error of inferior to 5 cm. In addressing the challenge of local obstacle avoidance, the chosen algorithm was the Rapidly-exploring Random Trees (RRT), selected for its inherent simplicity. However, the initial utilization of RRT resulted in paths that exhibited jagged trajectories. To mitigate this issue, a hybrid approach was developed, aimed at not only reducing path length but also minimizing jaggedness.

The final validation of the developed methods was carried out in Vienna, culminating in a successful demonstration. In the subsequent and last phase of the internship, the focus shifted to the development of a 3D polar contractor utilizing interval analysis techniques. This initiative was undertaken with the intention of resolving the rigid body transformation equation encountered when examining the same point cloud from diverse viewpoints.

Reflecting on this internship, I have gained invaluable insights across both technical and interpersonal domains. The experience has underscored the real nature of engineering, where adeptly addressing challenges extends beyond technical expertise. Proficiently managing issues is a fundamental aspect of an engineer's role. Equally important is the ability to effectively communicate intentions, as uncertainties arise when one's objectives are not clearly expressed, whether originating from my own perspective or from those engaging in conversation with me. This period of professional growth has fortified my understanding of engineering and its intricate interplay with communication skills.

References

[1] A. J. Hanson and H. Ma, "Parallel transport approach to curve framing," 1995.

[2] L. Lapierre and D. Soetanto, "Nonlinear path-following control of an auv," *Ocean Engineering*, vol. 34, no. 11, pp. 1734–1744, 2007.

[3] D. Agarwal and P. S. Bharti, "A review on comparative analysis of path planning and collision avoidance algorithms," *International Journal of Mechanical and Mechatronics Engineering*, vol. 12, no. 6, pp. 608 – 624, 2018.

[4] X. Wang, S. Williams, D. Angley, C. Gilliam, T. Jackson, R. Ellem, A. Bessell, and B. Moran, "Rrt* trajectory scheduling using angles-only measurements for auv recovery," 07 2019.

[5] V. Cichella, I. Kaminer, V. Dobrokhodov, E. Xargay, N. Hovakimyan, and A. Pascoal, "Geometric 3d path-following control for a fixed-wing uav on so(3)," 08 2011.

[6] R. Song, Y. Liu, and R. Bucknall, "Smoothed a* algorithm for practical unmanned surface vehicle path planning," *Applied Ocean Research*, vol. 83, pp. 9–20, 2019.

[7] A. Bethencourt and L. Jauli, "3d reconstruction using interval methods on the kinect device coupled with an imu," *International Journal of Advanced Robotic Systems*, vol. 10, p. 1, 02 2013.

[8] B. Desrochers and L. Jaulin, "A minimal contractor for the polar equation: Application to robot localization," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 83–92, 2016.

ENSTA Bretagne	**ASSESSMENT REPORT**
	This report should be signed, endorsed with company stamp by the internship supervisor and returned into aurion “Mise à jour de PFE”

Host organisation	Leibniz Universität Hannover, Real-Time Systems Group
Dates of the internship	March 6th - August 31st
Name of the intern	Internship: Path-following and OA algorithm for an UAV & Minimal rigid-body transform contractor using interval approaches

	Please tick the corresponding boxes					
	F	E	D	C	B	A
Personal and interpersonal skills						x
Adaptability						x
Availability						x
Corporate culture						x
Work commitment						x
Language skills (oral/written)						x
Project management						**x**
Task identification						x
Organization / distribution of tasks over internship period						x
Respect of deadlines (deliverables, reports, etc.)						x
Proactivity						x
Potentially: team work						x
Internship report						x
Format, style, etc.						x
Content						x
Usability for company/laboratory						x
Appreciation of ENSTA Bretagne Training						
Scientific and technical skills meet my expectactions						x
The methodlogical skills meet my expectations						x
On what subject did the student have to be trained before he became autonomous ?	Danut POP was already at the beginning a very well organized and autonomous student.					
Which skills or training content should be reinforced?	Obstacle avoidance in mobile robotics and optimization algorithms (Newton, Levenberg-Marquardt) should be considered in more detail					

Overall appreciation
We experienced Danut POP as a very sympathetic, intelligent, and focused colleague who was highly appreciated by our team. From the first day on, Danut POP integrated very well into our team. Due to his broad skillset, he was able to provide very good ideas and implementations in Python and C++ based on ROS. We highly appreciate his work, which combines theoretic and practical tasks which he has fulfilled beyond our expectations. We wish Danut POP all the best for his future.

In the event of a position becoming available, would you consider employing this student?	X YES ☐ NO

Name of the supervisor: Aaronkumar Ehambram **Date: 18.08.2023**

Title/Position: Research Associate **Signature:** A. Ehambram