SAPIENZA
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE ANTONIO RUBERTI (DIAG)

Control Barrier Function with variable obstacle velocity and spherical bounding volume

MODELING AND CONTROL OF MULTI-ROTOR UAVS

Professor:
Vendittelli Marilena

Students:
La Sala Carlo
Lubrano Valerio
Scuderi Matteo

Academic Year 2024/2025

Contents

1 Introduction

Unmanned Aerial Vehicles (UAVs) have experienced significant advancements over the last decade, driven by developments in electronics, materials, and computational power. These aerial systems have transitioned from military-exclusive technologies to widespread civilian applications, ranging from hobbyist drones to advanced autonomous platforms. Among the various configurations of UAVs, quad-rotors have emerged as one of the most widely adopted designs due to their simplicity, versatility, and high maneuverability.

Quad-rotors operate with four rotors positioned symmetrically, allowing them to hover, perform vertical takeoffs and landings, and execute agile movements. These characteristics make them suitable for a variety of applications, including aerial photography and cinematography, precision agriculture, infrastructure inspection, and environmental monitoring. In the industrial sector, quad-rotors are increasingly used for inventory management and maintenance tasks in hazardous or hard-to-reach areas. In addition, their potential for urban air mobility and delivery last-mile is actively explored.

In disaster response and search-and-rescue missions, quad-rotors play a crucial role by providing rapid situational awareness, locating survivors, and delivering essential supplies. Their compact design and ability to navigate constrained environments are particularly advantageous in such scenarios. Furthermore, in defense and security, quad-rotors are deployed for reconnaissance, surveillance, and tactical operations, often equipped with advanced sensors and communication systems.

Figure 1: DJI Quadrotor

Current research focuses on improving the autonomy and robustness of quad-rotor systems. Advances in computer vision, artificial intelligence, and sensor fusion have enabled significant progress in autonomous navigation, especially in GPS-denied environments. Technologies such as simultaneous localization and mapping (SLAM) allow quad-rotors to operate in complex and dynamic environments with minimal human intervention. Additionally, energy efficiency and flight endurance remain critical challenges, with ongoing studies exploring novel battery technologies and hybrid propulsion systems.

The future of quad-rotor technology lies in enhanced collaboration through swarm robotics, where multiple UAVs operate in coordination to achieve complex tasks. This is particularly relevant for applications such as large-scale mapping, precision spraying in agriculture, and coordinated rescue efforts. As regulations for UAV operations continue to evolve, quad-rotors are expected to play a central role in the integration of autonomous aerial systems into urban airspace, further expanding their capabilities and impact.

In summary, quad-rotors represent the forefront of UAV innovation, combining versatility, adaptability, and technological advancements. Their growing presence across diverse fields underscores their importance as key enablers of modern aerial robotics and intelligent systems,

2 The problem of trajectory tracking among obstacles for the UAVs

One of the fundamental challenges in the operation of quad-rotors is the ability to execute precise trajectory tracking in environments cluttered with both fixed and moving obstacles. Trajectory tracking refers to the capability of the UAV to follow a predefined or dynamically updated path while maintaining stability and avoiding collisions. This problem is critical in applications such as urban delivery, search and rescue, and autonomous navigation in dynamic environments.

The complexity of this problem arises from several factors:

- **Dynamic and unpredictable obstacles:** In real-world scenarios, obstacles such as vehicles, pedestrians, or other UAVs may have unpredictable movements that require real-time adaptation of the planned trajectory.
- **Fixed environmental constraints:** Static obstacles, such as buildings, trees, or power lines, create narrow corridors or constrained spaces that require highly accurate path planning and control.
- **Nonlinear dynamics:** Quad-rotors exhibit nonlinear and coupled dynamics, making it challenging to design control systems that can ensure stability and robustness while adapting to sudden changes in the environment.

- **Sensor limitations:** The performance of trajectory tracking is often constrained by the resolution, range, and latency of onboard sensors, which are responsible for detecting obstacles and providing situational awareness.

To address these challenges, state-of-the-art approaches combine advanced motion planning algorithms with robust control techniques. Motion planning methods, such as Rapidly-exploring Random Trees (RRT) and optimization-based approaches, are often used to generate collision-free trajectories. Meanwhile, control strategies such as Model Predictive Control (MPC) and adaptive controllers ensure the quad-rotor can accurately follow the planned trajectory despite dynamic disturbances.

Moreover, real-time obstacle detection and tracking are facilitated by advanced sensors, such as LiDAR, stereo cameras, and depth sensors, as well as techniques like simultaneous localization and mapping (SLAM). Recent research also incorporates artificial intelligence, enabling UAVs to predict obstacle behavior and plan trajectories that account for future movements.

The trajectory tracking problem in dynamic environments remains an active area of research, with ongoing efforts focused on improving computational efficiency, reducing reliance on external positioning systems, and enhancing the UAV's ability to handle highly dynamic scenarios.

3 Control Barrier Functions

In this section we will see the concept of **Control Barrier Functions** (CBFs from now on) and their application to **Obstacle Avoidance**. We'll later extend the discussion to a scenario with moving obstacles.

3.1 The concept of CBFs

To address the problem, we must first introduce the concept of CBFs, as they are essential in implementing controllers to solve this task.

Control Barrier Functions (CBFs) are a set of functions used in the design of controllers to ensure that dynamic systems operate under safe conditions. Formally, given a dynamic system described by the following differential equation:

$$\dot{x} = f(x, u) \tag{1}$$

where $x \in D \subset \mathbb{R}^n$ represents the state vector of the system, $u \in U \subset \mathbb{R}^m$ is the input vector, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear function describing the system's evolution, we can define a CBF as a continuous and differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ such that:

$$\begin{aligned} h(x) > 0,\ \forall x \in C, \\ \dot{h}(x) + \alpha(h(x)) \geq 0,\ \forall x \in C \end{aligned} \tag{2}$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ is a class-$\mathcal{K}$ function (i.e., a continuous, strictly increasing, and positive function with $\alpha(0) = 0$), and $C \subset \mathbb{R}^n$ is the desired safety set. Specifically, the safety set can be formally defined as:

$$C = \{x \in D \subset \mathbb{R}^n : h(x) \geq 0\} \tag{3}$$

These conditions ensure that the barrier function $h(x)$ remains positive within the defined safety set. Importantly, this framework assumes that obstacles are fixed and do not have their own velocity, which simplifies the safety constraints.

In practice, if a controller is designed to guarantee that the barrier function $h(x)$ remains positive throughout the evolution of the system, it can be guaranteed that the system never violates the safety constraints.

As for the UAV domain of application, the CBF considered as a starting point is the same described in [1]:

$$h(p(t), \dot{p}(t)) = (p(t) - \bar{p})^T (p(t) - \bar{p}) + \mu (p(t) - \bar{p})^T \dot{p}(t) \tag{4}$$

where $\bar{p}$ defines the **constant** spacial position of the obstacle.

In the next section, we will explore how to modify the CBF framework to accommodate more complex scenarios.

3.2 Extension for moving obstacles

If we consider scenarios in which the obstacles' position is not fixed, we have to modify the CBF to guarantee safety, as the direction of motion of the obstacles might violate the safety constraints. Thus, considering the theory explained in (Section 3.1) we can construct another CBF which takes into account the velocity of the obstacle:

$$h(p(t), \dot{p}(t), \bar{p}(t), \dot{\bar{p}}(t)) = (p(t) - \bar{p}(t))^T (p(t) - \bar{p}(t)) + \mu (p(t) - \bar{p}(t))^T (\dot{p}(t) - \dot{\bar{p}}(t)) \tag{5}$$

where $\bar{p}(t)$ represents the obstacle's position, $\dot{\bar{p}}(t)$ represents the obstacle's velocity and μ is a positive constant. We can then define the **safe set** as $S_{safe} := h(p(t), \dot{p}(t), \bar{p}(t), \dot{\bar{p}}(t)) > \delta$, where δ is the obstacle clearance. To ensure safety, the system's evolution must remain in S_{safe}, even if the reference trajectory would enter the unsafe set. For this reason, the controller switches dynamically between the two modes depending on whether $(p, \dot{p}) \in D_{\text{track}}$ or $(p, \dot{p}) \in D_{\perp}$, defined as:

$$D_{\text{track}} = D_{u^*} \cup D_{\delta_1}, \quad D_{\perp} = \mathbb{R}^{3n} \setminus D_{\text{track}}, \tag{6}$$

where D_{u^*} and D_{δ_1} are subsets ensuring safe navigation. In summary, this framework enables UAVs to navigate efficiently while prioritizing safety in obstacle-rich environments.

For sake of notation, we now define two projection operators. Their purpose is to "filter-out" the components of the acceleration that would drive the UAV towards the unsafe set. This is achieved by projecting the UAV's acceleration onto the plane orthogonal to the direction pointing towards the obstacle:

- $\prod_{p(t)-\bar{p}(t)} = (p(t) - \bar{p}(t))[(p(t) - \bar{p}(t))^T(p(t) - \bar{p}(t))]^{-1}(p(t) - \bar{p}(t))^T$;
- $\prod^{\perp}_{p(t)-\bar{p}(t)} = I - \prod_{p(t)-\bar{p}(t)}$.

We can now define the safety-ensuring controller

$$\hat{u} = -\frac{2}{\mu}\prod_{p-\bar{p}}(\dot{p} - \dot{\bar{p}}) + \prod_{p-\bar{p}}^{\perp} u^* + \ddot{\bar{p}}(t) \tag{7}$$

Where u^* is the nominal control input that ensures trajectory tracking in the safe set and whose implementation will be detailed in (Section 4).

We can now prove that the modified controller $\hat{u}$ guarantees that the evolution of the system stays in S_{safe}.

3.2.1 Proof: Safety guarantee

Following the same reasoning as [1], we need to prove that the modified controller $\hat{u}$ makes $\dot{h} \geq -\alpha(h(x))$. In other words, the defined controller makes the safe set S_{safe} forward invariant $\forall t \geq 0$ as long as the initial conditions $(p(0), \dot{p}(0)) \in S_{safe}$.

For brevity of notation we drop all time dependencies and, reminding that $\ddot{p} = u$, we can define $A := p(t) - \bar{p}(t)$, $\dot{A} := \dot{p}(t) - \dot{\bar{p}}(t)$, for which our CBF rewrites as :

$$h = A^T A + \mu A^T \dot{A}$$

By differentiation we obtain:

$$\dot{h} = 2A^T\dot{A} + \mu(\dot{A}^T\dot{A} + A^T\ddot{A}) = A^T(2\dot{A} + \mu u - \mu\ddot{\bar{p}})) + \mu\dot{A}^T\dot{A} \tag{8}$$

Finally, substituting $u = \hat{u}$ the equation simplifies to $\dot{h} = \mu\dot{A}^T\dot{A} \geq 0$ $(\geq -\alpha(h(x))$ *assuming* $(p(0), \dot{p}(0)) \in S_{safe})$

4 Implementation

As described in the previous section, the UAV can be modeled as a double integrator system using hierarchical control and a change of input (3).

$$\begin{cases} \ddot{x} = u_1 \\ \ddot{y} = u_2 \\ \ddot{z} = u_3 \end{cases} \tag{3}$$

Where x,y,z describe the spatial position of the quadrotor and u_1, u_2, and u_3 are the acceleration inputs.

This simplification allows the model to track both feasible (realizable) and non-feasible (unrealizable) trajectories using a combined feedforward + PD control (9).

$$u^* = \ddot{p_d} + K_d(\dot{p_d} - \dot{p}) + K_p(p_d - p), \tag{9}$$

where $K_d \in \mathbb{R}^{3\times3}$ and $K_p \in \mathbb{R}^{3\times3}$ are gain matrices, $p_d \in \mathbb{R}^3$ is the desired trajectory and $p \in \mathbb{R}^3$ is the UAV position, and $u^* \in \mathbb{R}^3$.

In our simulations several types of reference trajectories are implemented, including:

1. Straight-line trajectory
$$P_d = \begin{bmatrix} O_x - f \cdot t \\ O_y \\ O_z \end{bmatrix} \tag{10}$$
where O_x,O_y,O_z are the initial positions, f is a positive constant, and t is the time variable.

2. Helical trajectory
$$P_d = \begin{bmatrix} O_x + R \cdot \cos(f \cdot t) \\ O_y + R \cdot \sin(f \cdot t) \\ O_z + v \cdot t \end{bmatrix} \tag{11}$$
where R is the helix radius and v is the velocity along the $z - axis$.

3. Three-dimensional periodic trajectory
$$P_d = \begin{bmatrix} O_x + a \cdot \sin(f \cdot t) \\ O_y + b \cdot \sin(f \cdot t) \cdot \cos(f \cdot t) \\ O_z + c \cdot \sin(0.5 \cdot f \cdot t) \end{bmatrix} \tag{12}$$
where a, b and c are constants.

4. Right-angled trajectory This trajectory is composed of three distinct segments. For $t < 10$ seconds:
$$P_d = \begin{bmatrix} O_x + f \cdot t \\ O_y \\ O_z \end{bmatrix} \tag{13}$$
For $10 \leq t < 20$ seconds:
$$P_d = \begin{bmatrix} O_{x1} \\ O_y + f \cdot t - 10 \\ O_z \end{bmatrix} \tag{14}$$

For $t \geq 20$ seconds:

$$P_d = \begin{bmatrix} O_{x1} \\ O_{y1} \\ O_z + f \cdot t - 20 \end{bmatrix} \tag{15}$$

where O_{x1} and O_{y1} are initial positions.

4.1 Moving obstacles, known velocities

In this section we will cover the implementation for scenarios with moving obstacle whose velocities are known. This controller is developed using the **Switching Programming** technique.

Switching Programming for mobile robots leverages the **CBFs** technique to ensure safety during navigation. This approach defines a control framework with two modes: a nominal control law (9) for trajectory tracking and a safety control law u to avoid obstacles.

The system dynamically switches between these modes based on the environment. The safety control law relies on the CBF illustrated in (Section 3.2)

Thus, our final controller is defined as:

$$u = \begin{cases} u^* & if \quad (p, \dot{p}) \in D_{track} \\ (-2/\mu) \prod_{p-\bar{p}} \dot{p} + \prod_{p-\bar{p}}^{\perp} u^* + \ddot{\bar{p}}(t) + u^{\perp} & if \quad (p, \dot{p}) \in D_{\perp} \end{cases} \tag{16}$$

Where:

```
u_perp = 0;
if rank([p-pOB, p_dot, pOB_dot)],  0.1) == 1
    u_perp = ([-(p-pOB)(2); (p-pOB)(1); 0]);
end
```

This part of code means that if the velocity of the obstacle and the velocity of the UAV are parallel with opposite versus, a component orthogonal to the velocities direction will be added to the control input. This situation has to be treated in directed way beacuse this is one of the singularity of the theory of CBF.

4.2 Moving obstacles, unknown velocities

Now we can consider the case in which the UAV doesn't know the velocity of the obstacles. We can assume that on the UAV is mounted a camera system which provide to the quadrotor the real time position of the obstacles. Thus, we need to estimate the obstacle velocity and acceleration to use them in the CBF framework. We implemented

a Kalman Filter and we consider a process and a measurement affected by a white Gaussian noise.

This is the setup for the Kalman Filter:

- Sampling interval: dt
- Transition Matrix: $A_k = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
- Observation matrix: $H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
- Process Covariance matrix: $Q_k = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$
- Measurements Covariance matrix: $J_k = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$

Once we have the estimate of the obstacle velocity and acceleration from the Kalman Filter, we can use them in the controller based on Switching Programming.

5 Results and simulations

In this section we will see the main results of the simulations. Simulations are in MATLAB and UAV is represented by a sphere-shaped object and also the obstacles are spheres. The 3D animation is developed using Simulink and its UAV Toolbox

which provided us the 3D model of the UAV and the scenarios. This is the whole Simulink block scheme for the most challenging simulation:

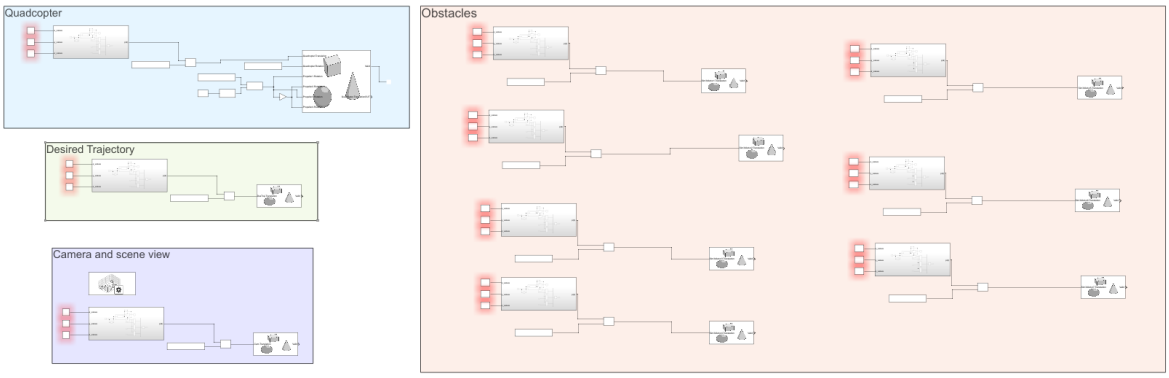

Figure 2: Simulink block scheme

5.1 Moving obstacles, known velocities

In this challenging scenario the UAV (bigger blue sphere) (safe radius $r = 5$) follows the 3D trajectory reference (12) (bigger orange sphere) and during its trajectory it runs into several obstacles of radius $R_{ob} = 1$. The obstacles have different behavior:

- blue: follows a circular trajectory;
- red and yellow: follow a straight forward and backward trajectory;
- green: fixed obstacles. The UAV can pass safety beetween them;
- cyan: fixed obstacles. The UAV can't pass safety between them, so the controller treats the two obstacles as a big obstacle and goes around it.

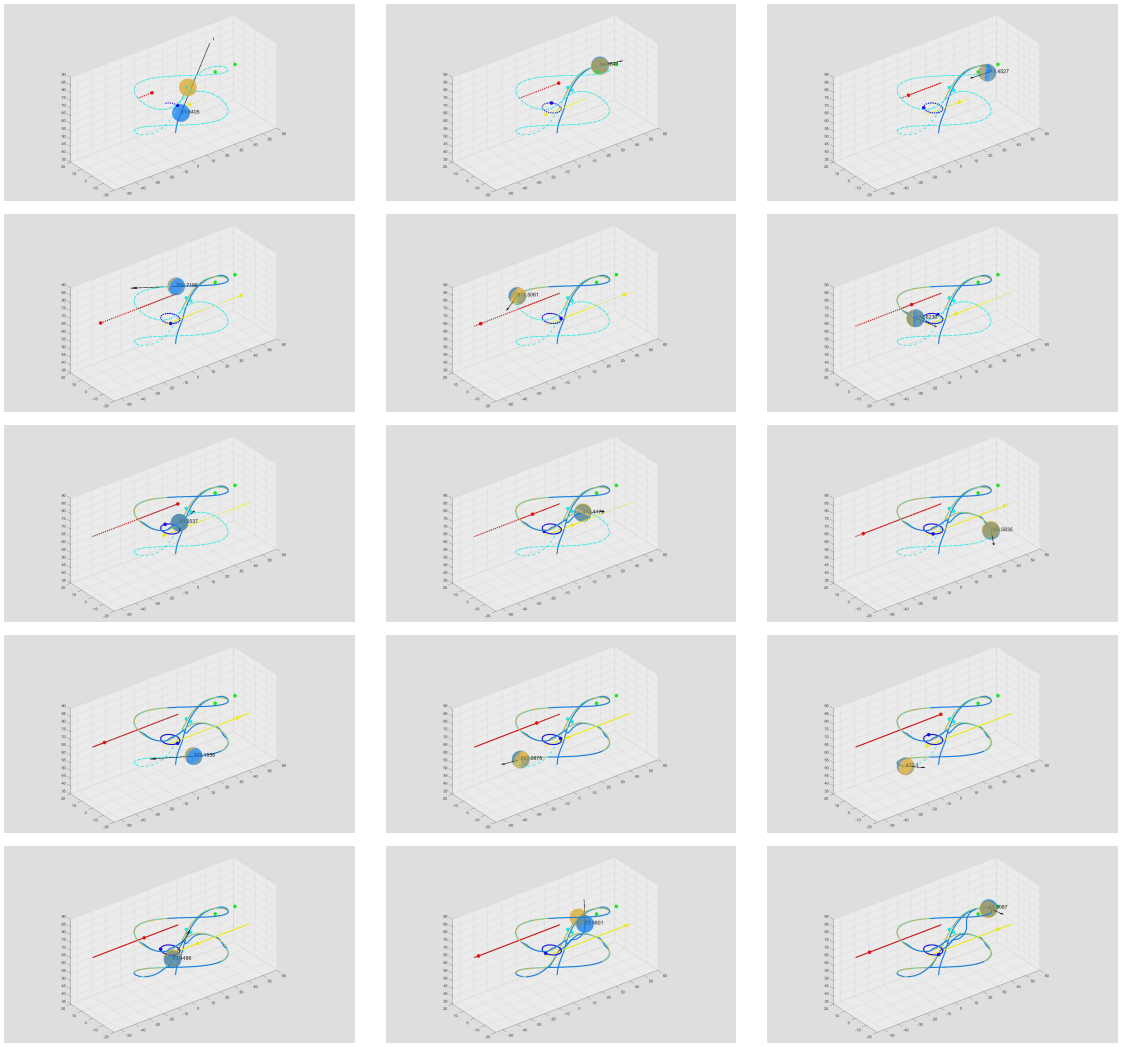

Figure 3: Several frame of the simulation

As we can seen in (Figure 3) the UAV follows the desired trajectory very well thanks to the nominal controller and when it runs into the obstacles the non-nominal controller works in a right way. In fact, using a controller based on switching programming and CBF the UAV flies in the safe zones and minimizes as much as possible the tracking error.

Figure 4: Screenshot of the Simulink 3D animation for this scenario

5.2 Moving obstacles, unknown velocities

In this simulation the UAV doesn't know the velocity and the acceleration of the obstacle a priori and using the Kalman Filter as described in section (Section 4.2) the UAV has the real-time estimation of the two measures. In this scenario, the UAV follows a straight-line trajectory at height of $z = 20$ and in the same plane there is an obstacle (of radius $R_{ob} = 2$) which moves on a circular trajectory of radius $R = 10$. The UAV's safety radius is $r = 5$. The Kalman Filter has estimated very well the obstacle velocity and acceleration as we can see in the following figure:

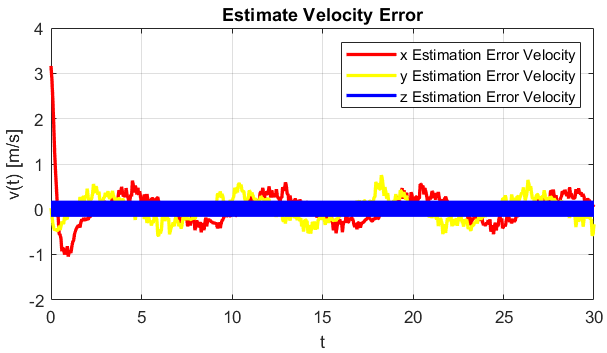

Figure 5: Estimation error on velocity

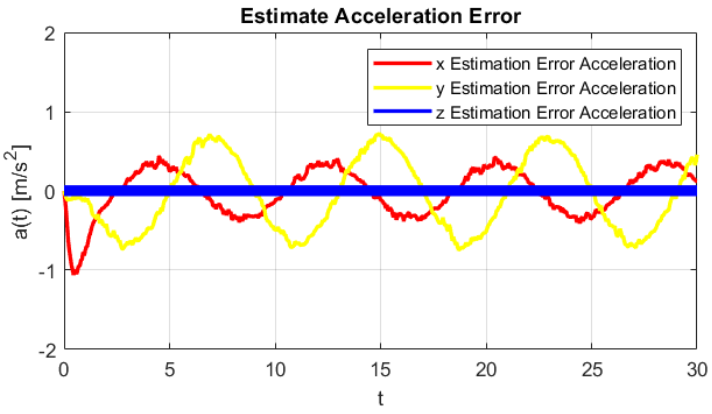

Figure 6: Estimation error on acceleration

In (Figure 5) and in (Figure 6) we can see how the Kalman filter works to estimate both the velocity and the acceleration of the obstacle. The velocity was reconstructed faithfully, with minimal error in the maximum and minimum amplitudes of the sine and cosine waves related to the velocities on the x- and y-axes. Same thing for the acceleration, where the noise is more evident, but the result is quite good.

As we can see in (Figure 7) the UAV follows the trajectory and when it runs into the obstacle the switching controller allows the UAV to change trajectory and avoid the obstacle

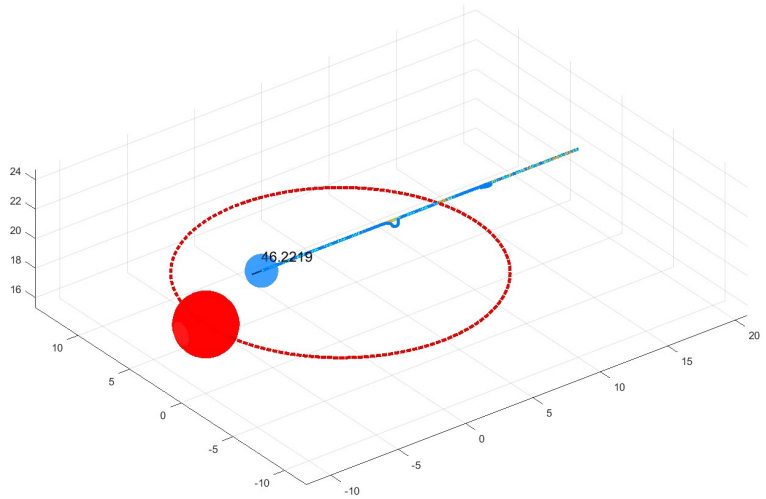

Figure 7: Final frame of the simulation with Kalman Filter

References

[1] Andrea Cristofaro, Marco Ferro, and Marilena Vendittelli. Safe trajectory tracking using closed-form controllers based on control barrier functions. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3329–3334, 2022.

[2] A. Cristofaro, M. Ferro, F. Galasso, M. Mizzoni, A. Pacciarelli, and M. Vendittelli. Combining 3d planning and control barrier functions for safe motion of quadrotor uavs among obstacles. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 2445–2450, 2023.

[3] Carlo Bruni and Carlo Ferrone. *Metodi di stima e filtraggio e l'identificazione dei sistemi.* Aracne Editrice, Roma, Italia, 2008.