Geometric Controller with APF method for UAV

Stefano Riccardi, Gaetano Torella, *Unina*
Repository: https://github.com/gaetanotorella/FSR_FinalProject/

***Abstract*—This project presents a geometric controller to manage the position and attitude of an Unmanned Aerial Vehicle (UAV). The developed control system aims to ensure high stability and reliability during flight. The Artificial Potential Field (APF) method was implemented for obstacle avoidance, providing a solution both in offline mode, for preemptive path planning and in online mode, for real-time adaptation using exteroceptive sensors.**

The simulations were conducted in two distinct development environments: Simulink and ROS/Gazebo. Simulink was used to model and simulate dynamic controls, allowing for a detailed analysis of system performance under controlled conditions. ROS/Gazebo provided a realistic simulation environment in which drone behavior was tested in complex scenarios that included various obstacles.

The simulation results indicate that the proposed system can maintain safe and precise navigation even in the presence of unexpected obstacles. The combined approach of geometric control and APF method has proven effective in avoiding collisions, significantly enhancing the drone's capability to operate autonomously in dynamic and potentially complex environments. This study confirms the validity of the proposed method and suggests its applicability for future implementations in autonomous drones intended for critical missions such as surveillance, inspection, and goods delivery.

I. Introduction

The aim of this project is the development of a fully autonomous unmanned aerial vehicle capable of reaching a specific position by avoiding crashes with the environment. The project starts with an analysis of the UAV dynamic model, with particular attention on quadrotors and the main control strategies like the geometric controller. In order to perform obstacle avoidance it has been chosen an artificial potential field algorithm to generate a reference trajectory for the controller both for a previous known environment and for a random environment. In the end, the control strategies presented above were implemented and validated using Matlab-Simulink and ROS.

II. UAV Dynamics

UAV Quadrotor dynamics are typically described using a body reference frame with a North-East-Down (NED) configuration where the x-axis points to the north, the y-axis points to the east and the z-axis points down in the direction of the Earth's center. The position of the quadrotor in the body frame is given by the vector:

$$p_b = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \tag{1}$$

The orientation of the quadrotor can be described both using the rotation matrix $R_b \in SO(3)$ (special orthogonal group) and through a minimal representation given by the Euler angles.

$$R_b = \begin{bmatrix} x_b & y_b & z_b \end{bmatrix} \qquad \eta_b = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{2}$$

The rotation from the body frame to the world frame is described by a series of rotations around the principal axes. The combined rotation matrix R_b is:

$$R_b(\eta_b) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3}$$

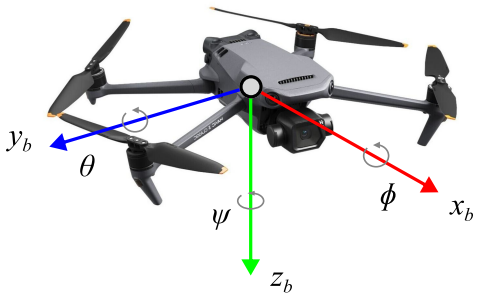


Fig. 1: Quadrotor UAV NED configuration

The input of UAV's dynamic model are the velocities of each propeller ω_i which provides a thrust $T_i > 0$; the total thrust produced by the drone is $u_T = \sum_{i=1}^{n} T_i > 0$ directed along the z_b-axis and the control torque applied on the drone to change its orientation is $\tau^b = [\tau_x \ \tau_y \ \tau_z] \in \mathbb{R}^3$. The relation between the single propeller thrust and its velocity is expressed by: $T_i = c_T \omega_i^2$ and $Q_i = c_Q \omega_i^2$ where $c_T > 0$ is the thrust constant and $c_Q > 0$ is the drag factor. To establish a relation between control torque and thrust and the rotor velocities, the *allocation matrix* is introduced:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ u_T \end{bmatrix} = \begin{bmatrix} 0 & -lc_T & 0 & lc_T \\ lc_T & 0 & -lc_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \\ c_T & c_T & c_T & c_T \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{4}$$

The dynamic model of the UAV can be expressed in the body frame using a coordinate-free model as follows:

$$\begin{cases} m\ddot{p}_b^b = mge_3 - u_T R_b e_3 + f_e \\ \dot{R}_b = R_b S(\omega_b^b) \\ I_b \dot{\omega}_b^b = -S(\omega_b^b) I_b \omega_b^b + \tau^b + \tau_e^b \end{cases} \tag{5}$$

where

- $R_b(\eta_b) \in SO(3)$ is the rotation matrix of the body frame with respect to the world frame:

$$R_b(\eta_b) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{6}$$

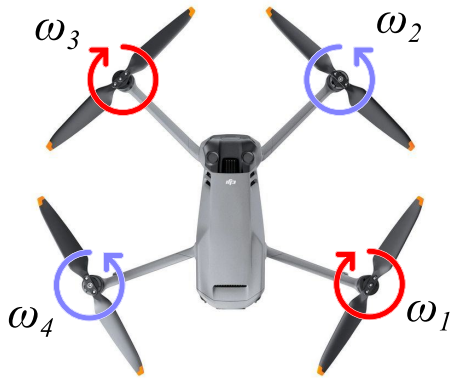


Fig. 2: Propellers configuration

- $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$;
- $I_b \in \mathbb{R}^{3\times3}$ is the diagonal inertia matrix for the angular part and is referred to the body frame;
- m is the mass of the UAV and it is positive definite;
- $f_e, \tau_e \in \mathbb{R}^r$ are external or unmodeled terms;
- u_T, τ^b are the control inputs of the UAV.

III. Geometric control

The control strategy chosen within the project is the geometric controller that uses a coordinate-free model to avoid all the singularities associated with roll-pitch-yaw (RPY) angles. This approach employs rotation matrices to represent rotations, ensuring more stable and precise control.

$$\begin{cases} m\ddot{p}_b^b = mge_3 - u_T R_b e_3 \\ \dot{R}_b = R_b S(\omega_b^b) \\ I_b\dot{\omega}_b^b = -S(\omega_b^b)I_b\omega_b^b + \tau^b \end{cases} \tag{7}$$

The translation motion of the system is determined by the total thrust u_T and the direction of thrust $R_b e_3$, where the magnitude of the total thrust u_T is directly controlled, whereas the direction of the thrust is dictated by the third body-fixed axis $z_{b,d}$. To change the direction of the total thrust, the desired system's attitude, represented by the rotation matrix $R_{b,d}$, must be adjusted accordingly.

The total thrust u_T and the desired $R_{b,d}$ are chosen to stabilize the zero equilibrium of the tracking error for translational dynamics. The desired $R_{b,d}$ is determined so that the z axis of the body-fixed frame points in the desired thrust direction. The first two columns of the rotation matrix $R_{b,d}$ represent the directions of the body-fixed axes and must follow the desired direction to maintain system stability. The control torque τ^b, is designed to follow the desired attitude resulting $R_{b,d}$ obtained by $x_{b,d}$ and $z_{b,d} = R_{b,d}e_3$

Once $z_{b,d}$ is defined, the desired $x_{b,d}$ might not be orthogonal to it. Therefore, $x_{b,d}$ must be projected into the plane orthogonal to $z_{b,d}$. So the result $R_{b,d}$ is the following:

$$R_d = \begin{bmatrix} \mathrm{S}(y_{b,d})y_{b,d} & \frac{\mathrm{S}(z_{b,d})x_{b,d}}{||\mathrm{S}(z_{b,d})x_{b,d}||} & z_{b,d} \end{bmatrix} \tag{8}$$

The controller's task involves ensuring the position p and velocity $\dot{p}$ track p_d and $\dot{p}_d$, respectively, and track the projection of x_b to the projection of $x_{b,d}$.

The tracking errors for the translational motion are defined as:

$$e_p = p - p_{b,d}, \quad \dot{e_p} = \dot{p} - \dot{p}_{b,d} \tag{9}$$

For angular motion, the tracking errors are defined in $SO(3)$:

$$e_R = \frac{1}{2}(R_{b,d}^T R_b - R_b^T R_{b,d}), \quad e_\omega = \omega_b^b - R_b^T R_{b,d}\omega_{b,d}^{b,d} \tag{10}$$

Here, e_R represents the angular error and e_ω the angular velocity error, where $(\cdot)^\vee$ is the vee map that converts a skew-symmetric matrix to a vector. The angular velocity error can be seen as the velocity of $R_{b,d}^T R_b$ in the body frame.

The control input u is given by:

$$u = -(-K_p e_p - K_v \dot{e_p} - mge_3 + m\ddot{p}_{b,d})^T R_b e_3 \tag{11}$$

The desired $z_{b,d}$ is:

$$z_{b,d} = -\frac{-K_p e_p - K_v \dot{e_p} - mge_3 + m\ddot{p}_{b,d}}{\| - K_p e_p - K_v \dot{e_p} - mge_3 + m\ddot{p}_{b,d}\|} \tag{12}$$

For the inner loop control, after retrieving $R_{b,d}$, the control torque τ^b is designed as:

$$\begin{aligned} \tau^b = &- K_R e_R - K_\omega e_\omega + S(\omega_b^b)I_b\omega_b^b \\ &- I_b\left(S(\omega_b^b)R_b^T R_{b,d}\omega_{b,d}^{b,d} - R_b^T R_{b,d}\dot{\omega}_{b,d}^{b,d}\right) \end{aligned} \tag{13}$$

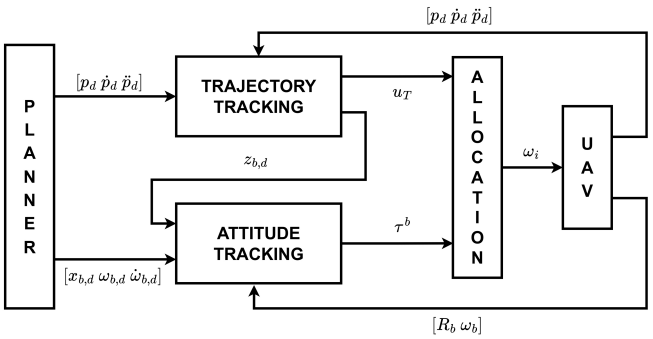


Fig. 3: Geometric control scheme

This type of control ensures the exponential stability of the closed-loop system if the initial attitude error is less than $90°$, and the converges to zero of the position tracking when there is no attitude error, and it's limited for non-zero attitude tracking errors.

IV. APF

One of the essential sub-systems in UAV is path planning which will guide and navigate the robot to reach the goal with obstacle avoidance. Some path planning methods have been proposed, such as the iteration method and the real-time method. The iteration method, or heuristic method, can give better path planning but needs more time for computing to find good path planning. Thus it will need more time for processing in a larger area. Meanwhile, the real-time method, or classical algorithm, does not need iteration and has simple mathematical equations. Thus, it will need less time for processing [2]. The

real-time method analyzed in this project is APF, it has simple mathematics equations so it needs low computation. Thus it will be suitable for real-time application [2]. The APF gives the velocity input to the planner that generates the desired trajectory to the controller, the proposed system is shown in Fig:

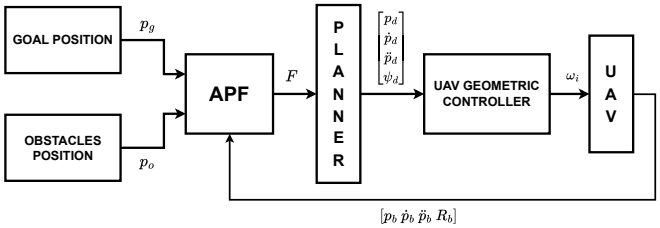


Fig. 4: APF control scheme

The APF method exploits the superimposition of two forces:

- An attractive force to the goal
- A repulsive force away from the $CO - obstacle$ region

A. Attractive Force

Attractive force is the force used to navigate to the goal. Supposing $q = [x, y, z]$ the UAV position in the space, and q_g the goal position, it's possible to compute an attractive potential as:

$$U_a(q) = \frac{1}{2} K_a e^T(q) e(q) = \frac{1}{2} K_a ||e(q)||^2 \tag{14}$$

Where:

- $e(q) = q_g - q$
- K_a is an attractive gain positive define

It is possible to retrieve the resulting force as:

$$f_a(q) = -\nabla U_a(q) = K_a e(q) \tag{15}$$

Note that with this formula when $e(q) = 0$ result $f_a = 0$ but the force increases with the error $e(q)$, therefore the value of the force could be too high for large initial errors. In the figure is shown the behavior of the attractive force considering $q_0 = [0, 0, 0]$ and $q_f = [10, 5, 0]$), choosing $k_a = 0.8$.

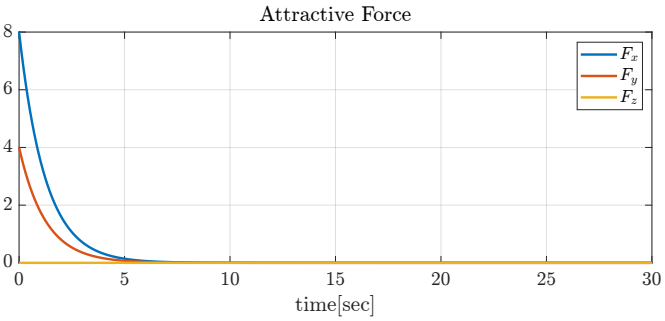


Fig. 5: Attractive Force f_a

Another choice for the computing of attractive force is considering:

$$U_b(q) = K_a ||e(q)|| \tag{16}$$

$$f_b(q) = -\nabla U_a(q) = k_a \frac{e(q)}{||e(q)||} \tag{17}$$

In this way the force does not go to infinite when $e(q)$ increases in the norm, therefore it is suitable for large initial errors, but it is important to notice that the force is not defined when $q = q_g$.

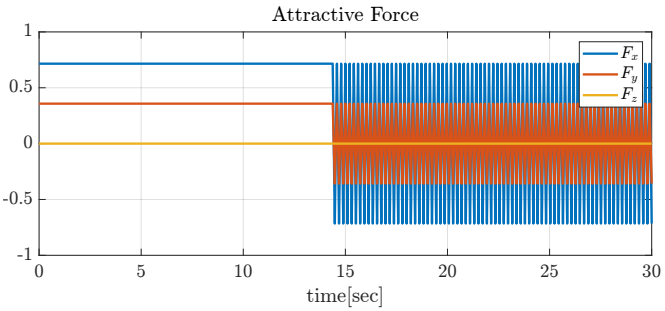


Fig. 6: Attractive Force f_b

A good choice to allow large initial errors and to have good behavior in the target position is to combine the two forces:

$$f(x) = \begin{cases} f_a, & \text{if } \|e(q)\| < 1 \\ f_b, & \text{if } \|e(q)\| \geq 1 \end{cases} \tag{18}$$

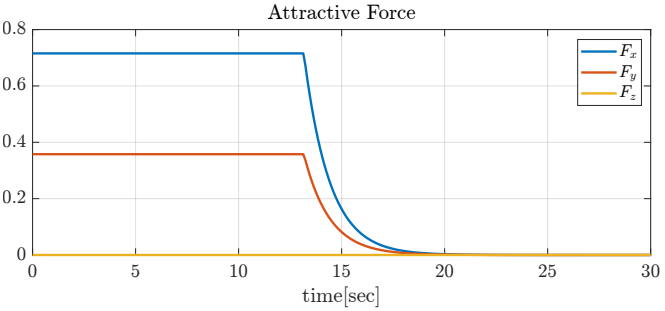


Fig. 7: Attractive Force combining f_a and f_b

B. Repulsive Force

The repulsive force is needed to avoid collisions, creating a sort of virtual barrier around the obstacles. Considering all the $C - obstacle$ region $CO_i, i = 1, ..., p$, for each CO_i a potential is associated:

$$\mathbf{U}_{r,i}(q) = \begin{cases} \frac{\mathbf{k}_{r,i}}{\gamma} \left(\frac{1}{\eta_i(q)} - \frac{1}{\eta_{o,i}} \right)^{\gamma}, & \text{if } \eta_i(q) \leq \eta_{o,i} \\ 0, & \text{if } \eta_i(q) > \eta_{o,i} \end{cases} \tag{19}$$

- $\mathbf{k}_{r,i}$ is a positive constant gain.
- $\eta_i(q) = \min_{q' \in CO_i} ||q - q_{obs}||$ is the current minimum distance from the obstacle .
- $\eta_{o,i}$ is a threshold distance beyond which the repulsive force is zero, called "*range of influence*".
- $\gamma = 2, 3$ is a factor.

The repulsive force $f_{r,i}(q)$ is defined as the negative gradient of a repulsive potential $U_{r,i}(q)$:

$$f_{r,i}(q) = -\nabla U_{r,i}(q)$$

The repulsive force is defined as:

$$f_{r,i}(q) = \begin{cases} \frac{\mathbf{k}_{r,i}}{\eta_i(q)^2}\left(\frac{1}{\eta_i(q)} - \frac{1}{\eta_{o,i}}\right)^{\gamma-1}\nabla\eta_i(q), & \text{if } \eta_i(q) \leq \eta_{o,i} \\ 0, & \text{if } \eta_i(q) > \eta_{o,i} \end{cases} \tag{20}$$

The gradient of the distance $\nabla\eta_i(q)$ is given by [2]:

$$\nabla\eta_i(q) = \frac{\mathbf{q}_{obstacle} - \mathbf{q}}{\eta_i(q)} \tag{21}$$

Represents the derivative of the distance with respect to the position. This gradient is a vector pointing from the current position $\mathbf{q}$ to the fixed point $\mathbf{q}_{obstacle}$, and its magnitude is normalized by the distance $\eta_i(q)$. The figure shows the Repulsive force with the same previous configuration, adding an obstacle $q_{obs} = [2, 2, 0]$, $k_r = 0.8$, and the factor $\gamma = 3$.

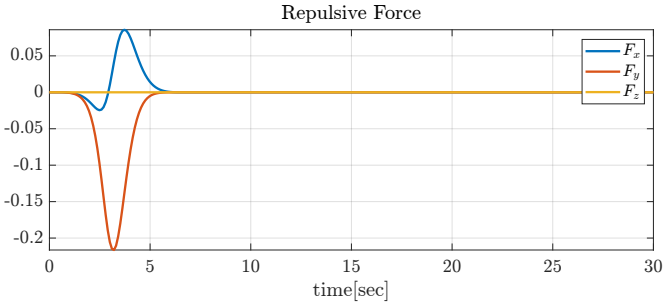


Fig. 8: Repulsive Force f_r

The difference between online and offline APF implementation lies in how the repulsive force is computed. In online scenarios, dividing the CO_i is impractical, leading to issues when dealing with closely situated obstacles. Therefore, in the online version, an assumption is often made that only one obstacle will be encountered at a time. Alternatively, the algorithm can be adjusted to calculate the repulsive force for each point retrieved by the Lidar sensor within a distance less than $\eta_{o,i}$. This approach would naturally result in a stronger and more variable repulsive force. Therefore, it necessitates a finer tuning of the gains k_a and k_r. Additionally, filtering f_r could be beneficial to reduce its discontinuity and ensure smoother operation.

C. Total Force

The total force of APF is the sum of the attractive force and the repulsive force of obstacles as:

$$F_{total}(q) = f_a(q) + \sum_{i=1}^{p} f_{r,i}(q) \tag{22}$$

The total forces are seen as the velocity vector moving the robot and become the input for the planner.

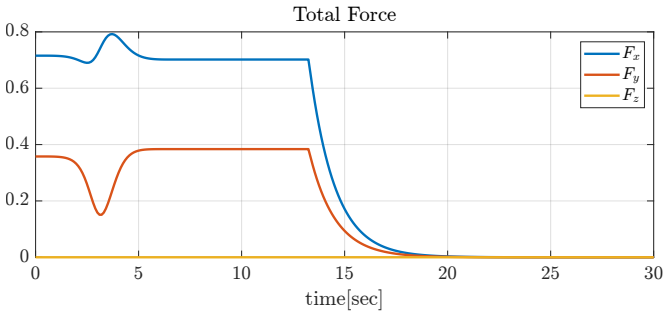


Fig. 9: Total Force F_{total}

V. Momentum Based Estimator

To improve the performances of the controller can be useful to compute the disturbances acting on the UAV during the flight; for this reason the momentum-based estimator of order *r* has been implemented. The r-order estimator can be formally written as:

$$\gamma_i(t) = K_i \int_0^t - \begin{bmatrix} \widehat{f}_e \\ \widehat{\tau}_e \end{bmatrix} + \gamma_{i-1} dt \tag{23}$$

for $i = 2...r$, starting from:

$$\gamma_1(t) = K_1 \left(q - \int_0^t - \begin{bmatrix} \widehat{f}_e \\ \widehat{\tau}_e \end{bmatrix} + \begin{bmatrix} mge_3 - u_T R_b e_3 \\ C^T(\eta_b, \dot{\eta}_b)\dot{\eta}_b + Q^T(\eta_b)\tau^b \end{bmatrix} dt \right) \tag{24}$$

Where K_i can be computed from the order r Butterworth Filter coefficient as follows (for $j = 0,\ 1,\ \ldots\ r-1$):

$$\prod_{i=j+1}^{r} K_i = c_j \tag{25}$$

$$G(s) = \frac{c_0}{s^r + c_{r-1}s^{r-1} + \ldots + c_1 s + c_0} \tag{26}$$

The momentum-based estimator can also be used to compute uncertainties in drone parameters; in this project, it has been employed to calculate uncertainty on the UAV mass as follows:

$$m_{real} = \frac{f_{ez}(t_{end})}{g} + m \tag{27}$$

VI. Matlab/Simulink Implementation

The UAV chosen for the simulations is a quadrotor with a mass $m = 1.52\ kg$ with the following inertia matrix:

$$I_b = \begin{bmatrix} 0.0347563 & 0 & 0 \\ 0 & 0.0458929 & 0 \\ 0 & 0 & 0.0977 \end{bmatrix} [kg \cdot m^2] \tag{28}$$

and with a thrust coefficient $c_T = 1.6 \times 10^{-2}$ and a drag coefficient $c_Q = 8.55 \times 10^{-6}$.

The resulting allocation matrix is:

$$A = \begin{bmatrix} -5.44 \cdot 10^{-7} & 5.44 \cdot 10^{-7} & 5.44 \cdot 10^{-7} & -5.44 \cdot 10^{-7} \\ -5.44 \cdot 10^{-7} & 5.44 \cdot 10^{-7} & -5.44 \cdot 10^{-7} & 5.44 \cdot 10^{-7} \\ -1.37 \cdot 10^{-7} & -1.37 \cdot 10^{-7} & 1.37 \cdot 10^{-7} & 1.37 \cdot 10^{-7} \\ 8.55 \cdot 10^{-6} & 8.55 \cdot 10^{-6} & 8.55 \cdot 10^{-6} & 8.55 \cdot 10^{-6} \end{bmatrix} \tag{29}$$

A first simulation was done with the UAV starting from the position $q_i = [0\ 0\ 0]$ and with goal position $q_f = [10\ 10\ 10]$ keeping a value of yaw $\psi = 0°$ and the parameters of the outer and inner loop of the geometric control chosen are the following:

Params	x	y	z
K_P	6	6	6
K_V	4.7	4.7	4.7

Params	ϕ	θ	ψ
K_R	2	2.3	0.15
K_W	0.4	0.52	0.18

TABLE I: Geometric control gain

A. *Offline APF*

First of all, the offline APF algorithm was implemented according to this chart:

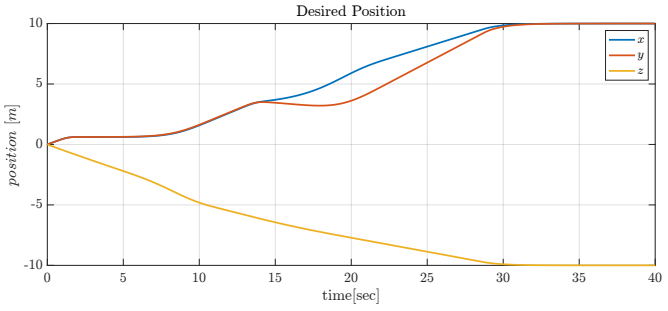
- the obstacles were discretized into a series of points belonging to their outer surface;
- starting from the initial position, the components F_{att} and F_{rep} were computed as seen previously;
- a new position value was computed according to the relation $q = q + F * Ts$ with $T_S = 0.01s$
- the position value obtained is substituted for the current position and the process is iterated creating a vector of timed waypoints with a period of T_S until the final position is reached;

A trajectory with minimum *jerk* was generated using Matlab's *minjerkpolytraj* function starting from the obtained waypoints. In the first simulation, the values of the APF gains are $k_a = 0.8$ and $k_r = 0.8$ and two cylindrical obstacles described by the characteristics shown in (table: II):

Obstacle	x_c	y_c	r	h
obs_1	$4.5m$	$5m$	$0.3m$	$10m$
obs_2	$2m$	$2m$	$0.2m$	$3m$

TABLE II: Obstacles parameters

The desired trajectory values obtained are shown in the figure (fig: 10) and the desired path is shown in (fig: 11)



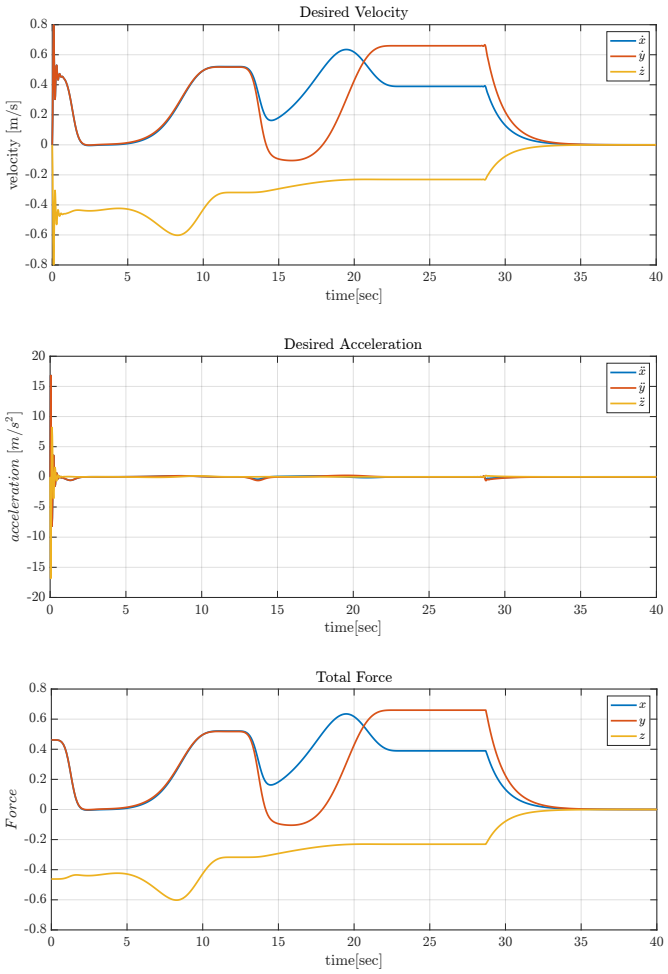


Fig. 10: Desired trajectory values

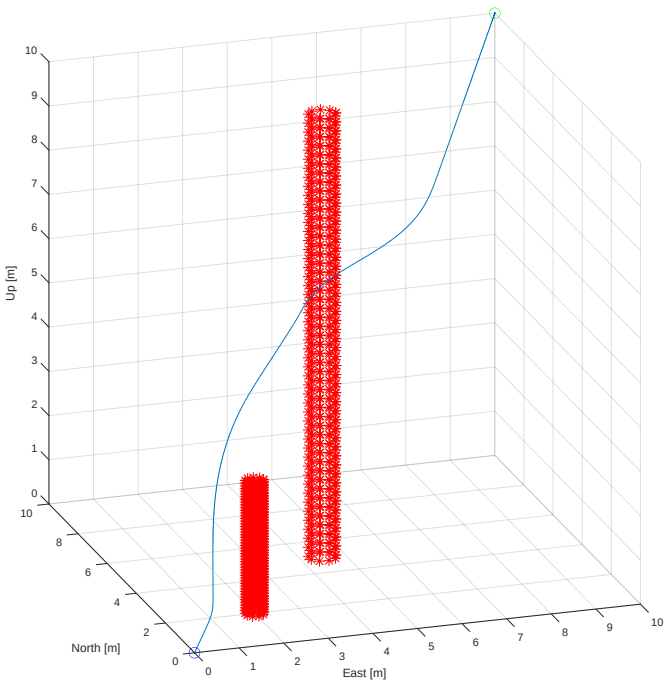
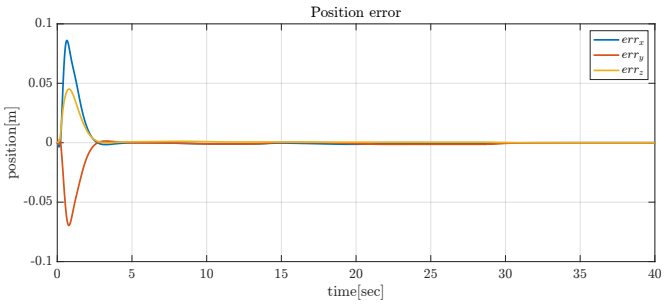
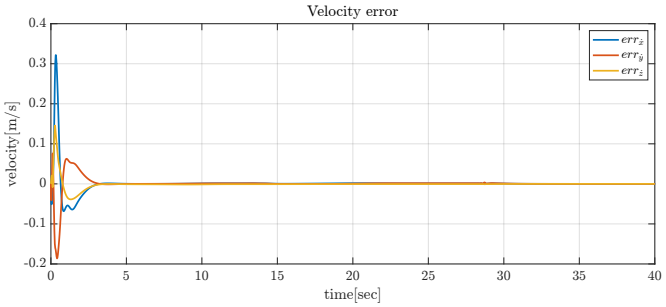
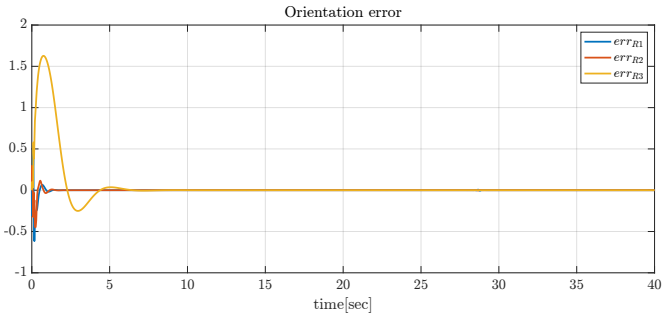


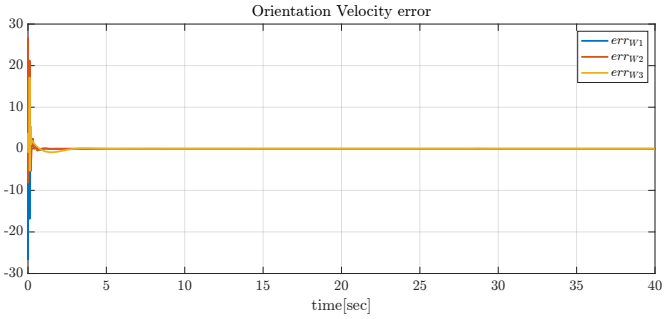
Fig. 11: Desired trajectory

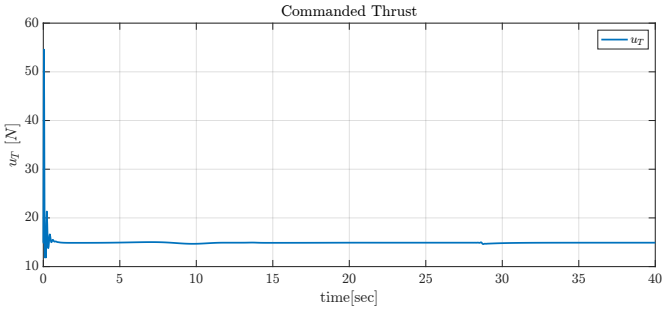
This reference trajectory is passed as a planner to the geometric controller on Simulink; the results of this simulation are shown in the figure (fig: 12)











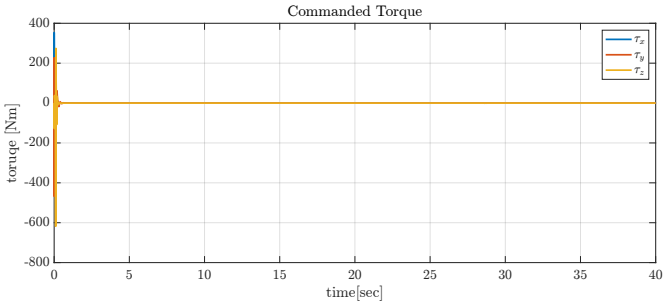


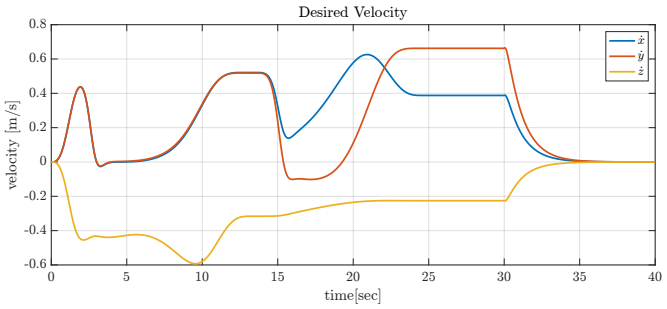
Fig. 12: Trajectory values

From the simulation results, it can be seen that the UAV follows the desired trajectory to the goal position avoiding the obstacles. Nevertheless, it is possible to notice a huge tracking error in both position and speed in the first time interval because the total force computed from the APF method is different from zero in the first instant of time, for this reason, both velocity and acceleration computed by *minjerkpolytraj* function are huge and oscillatory. To avoid this behavior, it has been implemented a moving average filter with a time-variant coefficient α:

$$F_{filtered}(k) = (1-\alpha)F_{filtered}(k-1) + \alpha F(k) \qquad (30)$$

where α is time variant.

The results obtained filtering the force of APF are shown in the figures (fig: 13)



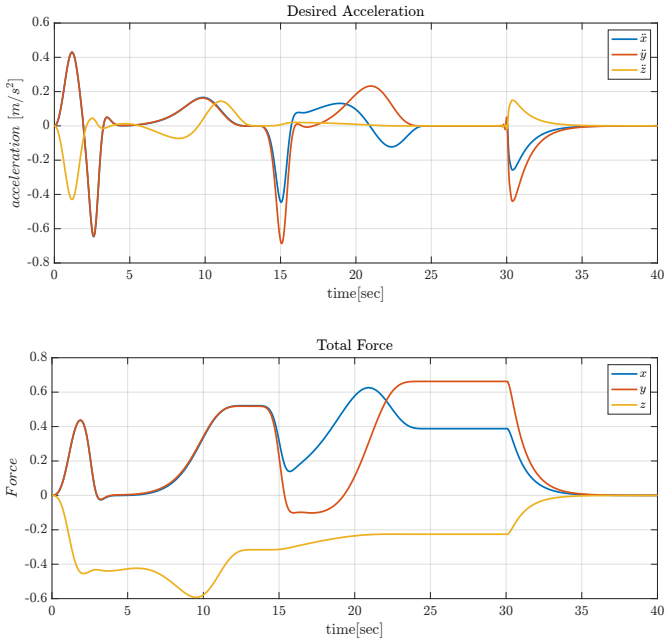
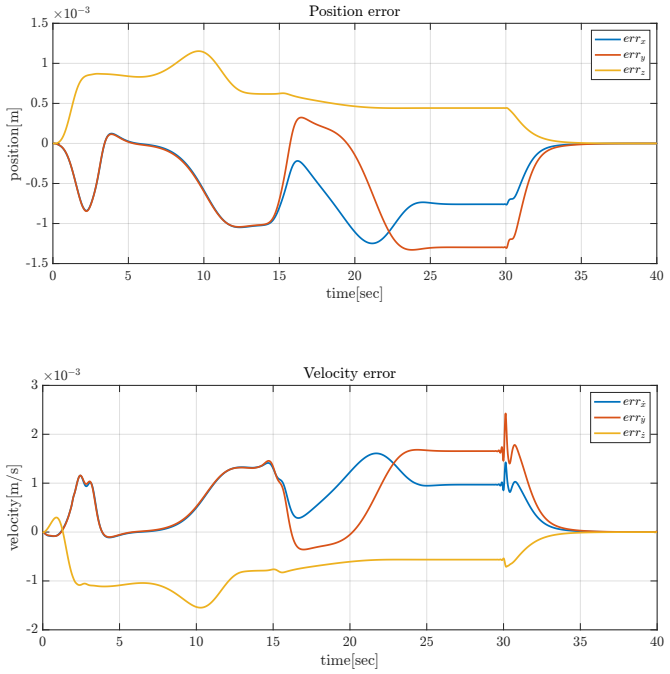
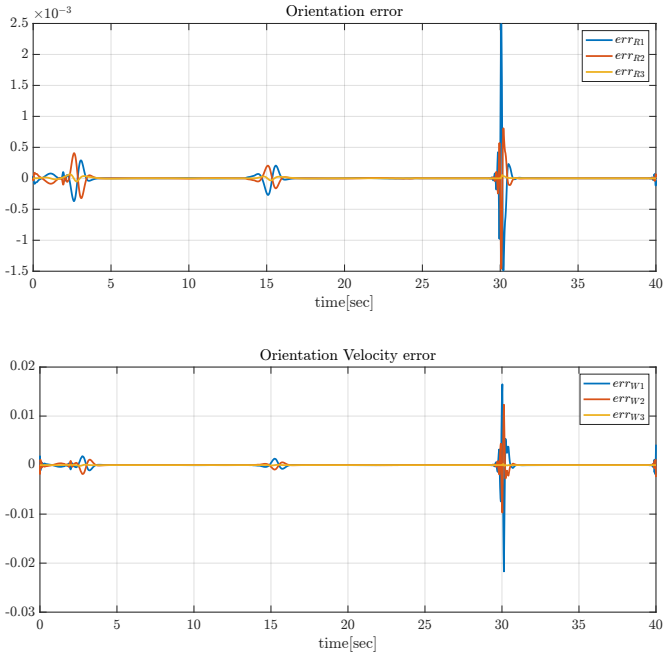


Fig. 13: Desired trajectory values with filter

It results that the path is the same but the desired velocity and acceleration values are smoother. Therefore, the tracking error and the commanded torque are reduced as the figures (fig: 14) show.





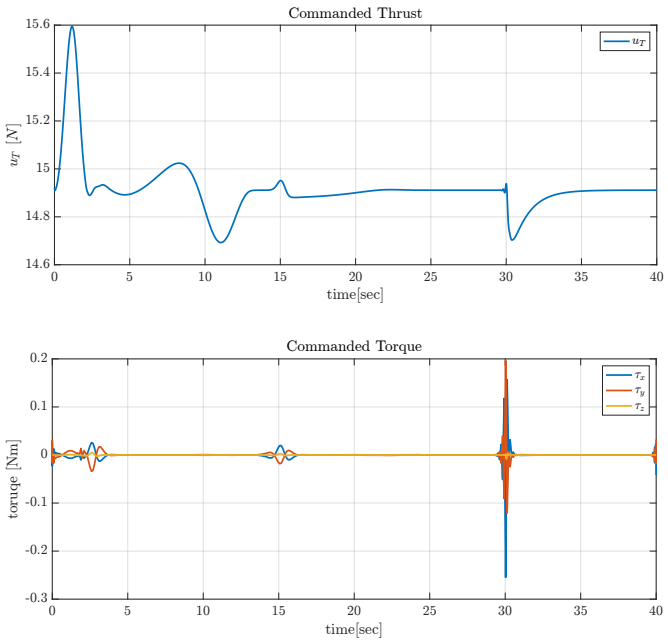


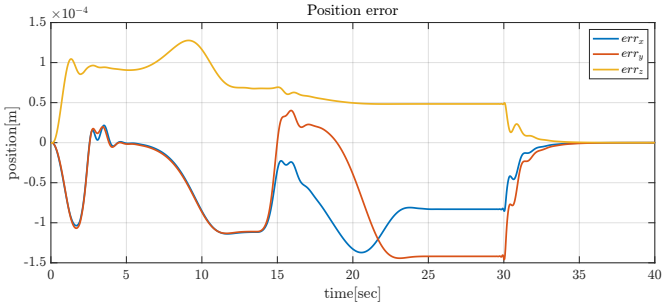
Fig. 14: Trajectory values with filter

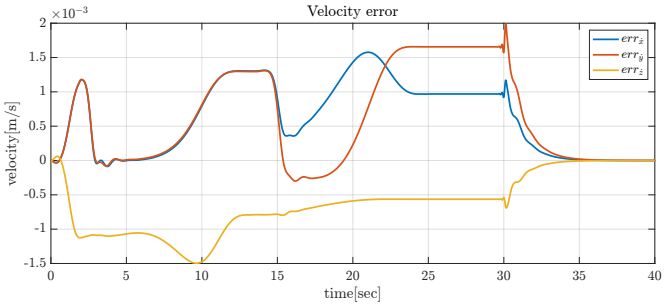
To further reduce tracking errors, geometric control gains have been modified as follows:

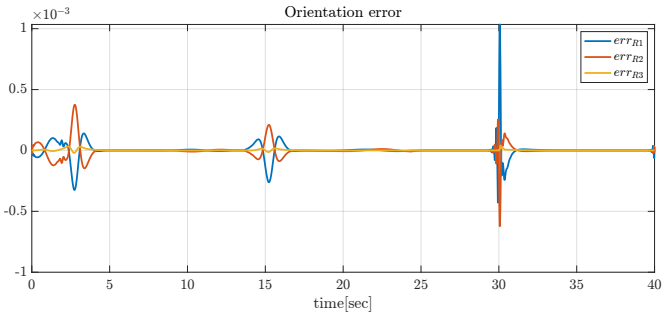
Params	x	y	z
K_P	70	70	70
K_V	6	6	6

Params	ϕ	θ	ψ
K_R	10	10	1.5
K_W	4	5.2	1.8

TABLE III: Geometric control gain







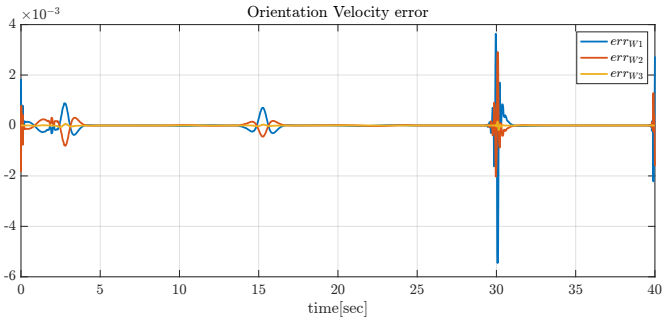


Fig. 15: Trajectory values

The figure (16) shows the trajectory obtained in the 3D space with the real obstacles

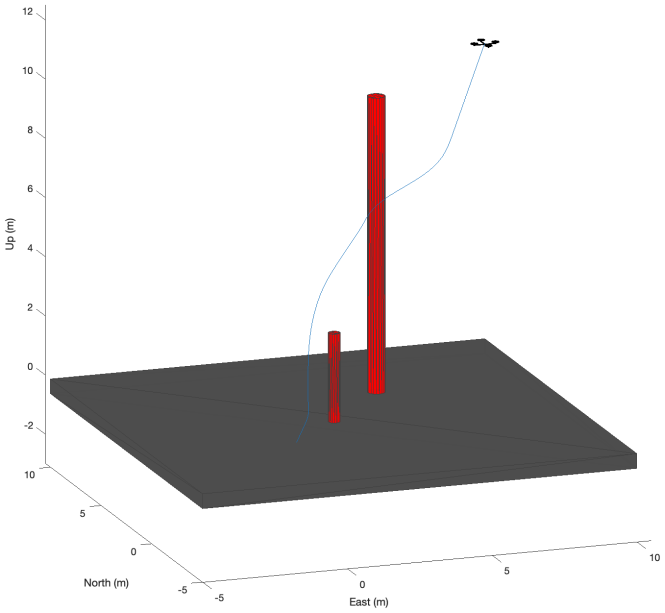
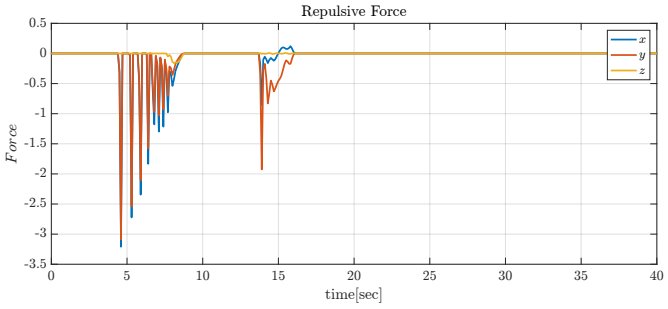
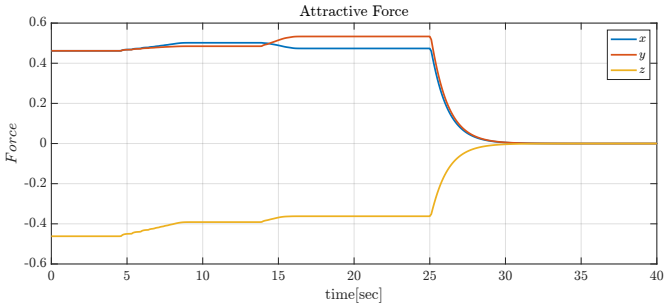


Fig. 16: Simulation trajectory

Some simulations were carried out to explore the behavior of the APF by changing the parameter $\eta_{o,i}$ and gains. An initial simulation was performed by keeping the control gains and the APF gains as previously set while decreasing the influence coefficient $\eta_{o,i}$ from 2.5 to 0.5. It was found that the UAV reaches the goal while avoiding obstacles, but the force generated by the APF exhibits strong variations near the obstacles(fig: 17). The desired path is shown in the figure (fig: 18).





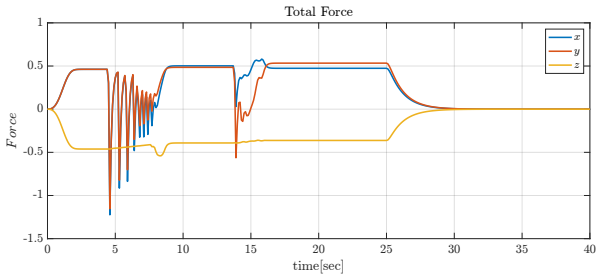


Fig. 17: Forces values

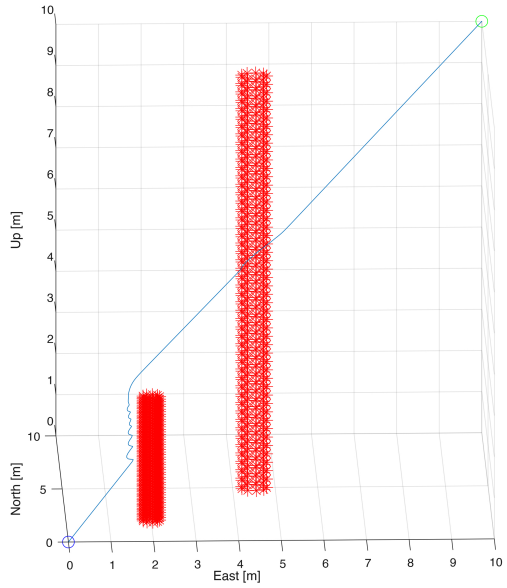


Fig. 18: Desired Trajectory

These issues in generating the desired reference for control result in altered values of τ^b and u_T, as shown in the figure (fig: 19).

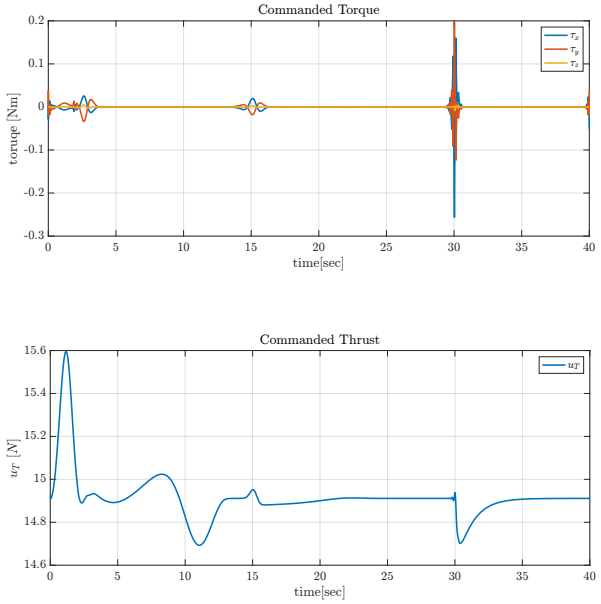


Fig. 19: Command values

Another simulation was performed aiming to reach the desired goal in a shorter time compared to the previous ones. Thus, a greater attractive force gain $k_a = 3.8$ was imposed while keeping the other parameters and gains unchanged.

As expected, the attractive force is greater, while the repulsive force exhibits some peaks (fig: 20), resulting in alterations of τ^b and u_T (fig: 21).

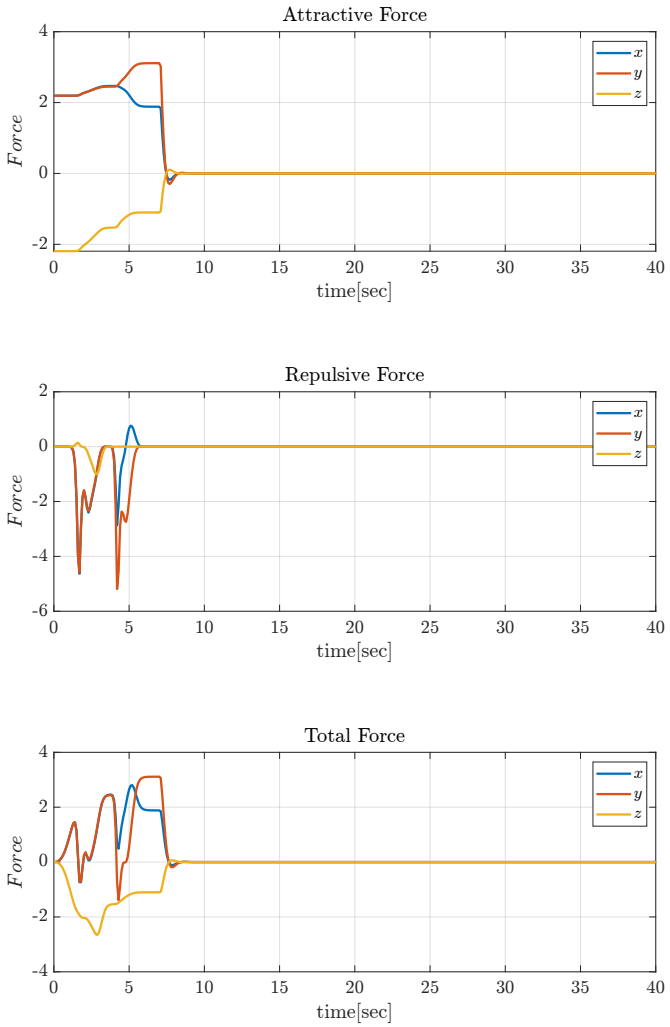
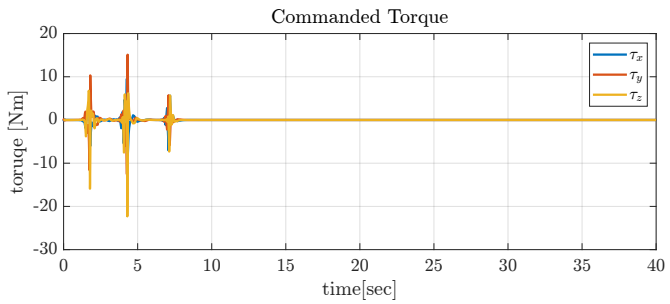


Fig. 20: Force values



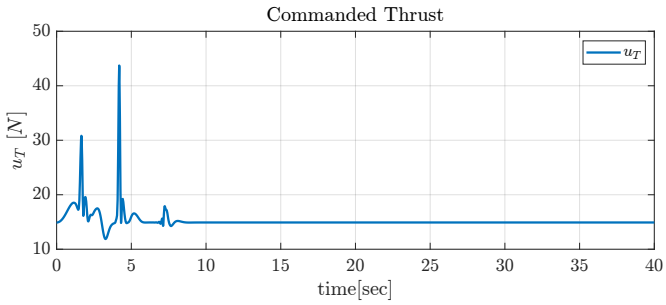


Fig. 21: Command values

To avoid the peaks in F_{rep}, the coefficient $\eta_{o,i}$ was increased to 6.5. In this case, the total force is smoother compared to the previous simulation (fig: 22), resulting in better values in the commanded thrust and torque (fig: 23).

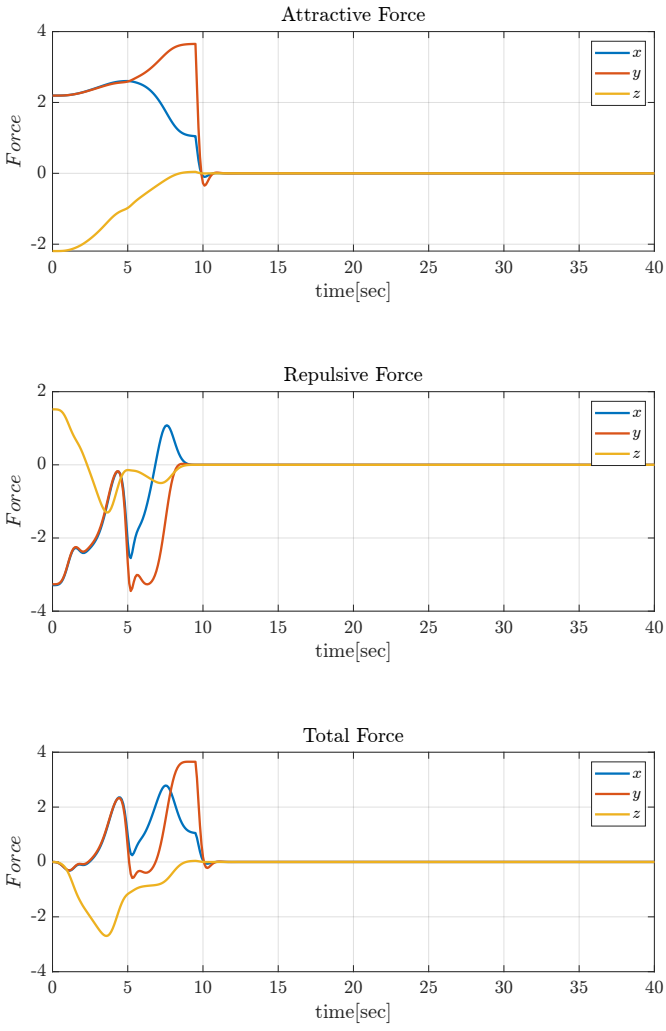


Fig. 22: Command values

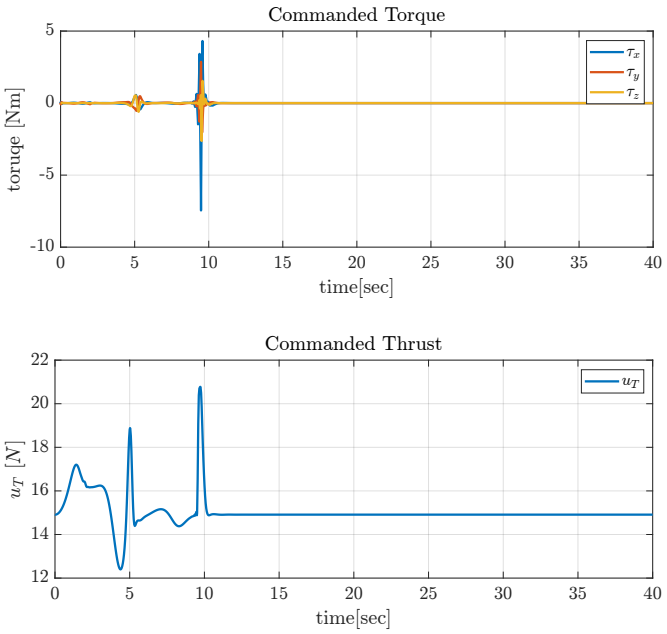


Fig. 23: Command values

An alternative approach to the one just illustrated is to use the force generated by the APF directly as the desired velocity. By using numerical integration, the desired position can be obtained, and by differentiation, the acceleration can be derived. In this case, the planner, after calculating F_{tot}, becomes:

$$\mathbf{v} = \begin{bmatrix} v_{x,d} \\ v_{y,d} \\ v_{z,d} \end{bmatrix}$$

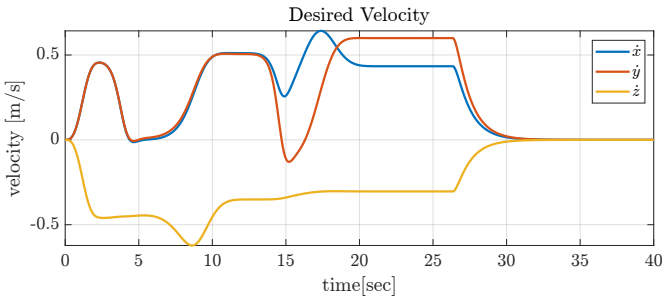
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{v} \cdot T_s$$

$$\phi_k = \phi_{k-1} + \omega_\phi \cdot T_s$$

$$\mathbf{a} = \frac{\mathbf{v} - \mathbf{v}_{k-1}}{T_s}$$

$$a_\phi = \frac{\omega_\phi - \omega_{\phi_{k-1}}}{T_s}$$

A simulation was performed keeping the previous scenario unchanged. By choosing $k_a = 0.8$ and $k_r = 0.03$, it results that the behavior is similar to the previous one. The results of the outputs from the planner are shown in the figure (fig: 24).



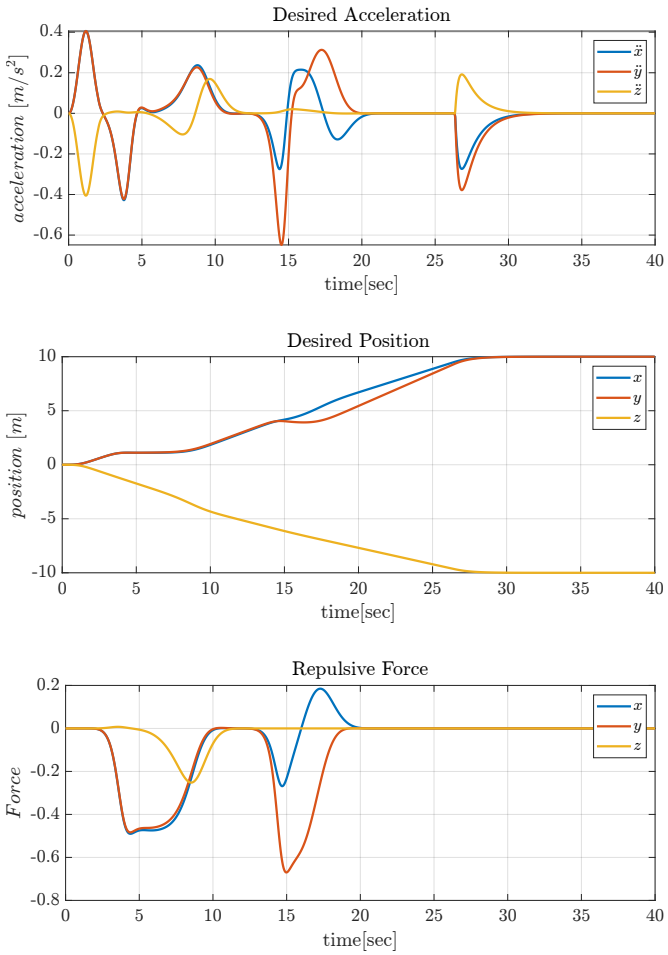


Fig. 24: Desired values with Dynamic APF

This alternative method does not appear to offer advantages in terms of results, but it facilitates the conversion of the algorithm to an online version. For this reason, this method was implemented also in ROS simulation.

VII. ROS Implementation

The simulations carried out on Simulink yielded excellent results regarding implementing the APF method in the offline version. However, it was decided to implement and validate this control strategy using the ROS middleware.

The ROS package adopted for this part of the project is *RotorS Simulator* [1] which allows the simulation of drones inside Gazebo, in particular, the UAV chosen is the *ArDrone*, a quadrotor with propellers on the same plane (fig: 25)

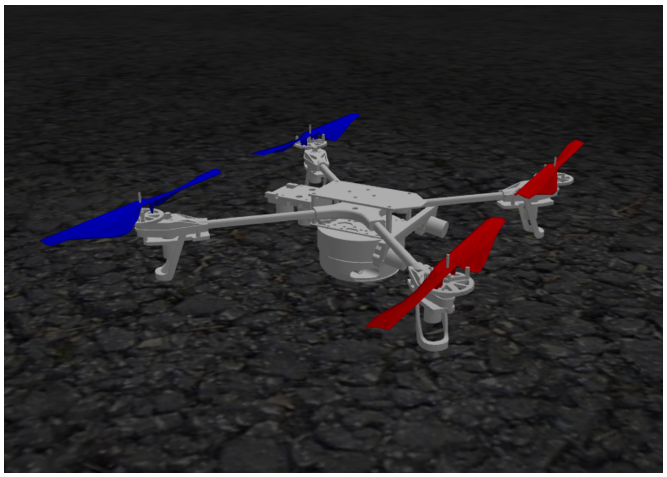


Fig. 25: ArDrone

The inertia parameters of the Ardrone are the same as the Simulink simulation (28), and the allocation matrix matches the one seen previously (29). RotorS package offers the possibility to control drones either through individual rotor speeds or through the desired roll, pitch, yaw, and thrust commands; it was decided within the project to control the UAV via RPY and thrust, which represent the outputs of geometric control. Furthermore, for the online implementation of the APF, a Lidar sensor was added to the base of the drone, assuming negligible mass and inertia (fig: 25).

A. Offline APF

Three ROS nodes were developed for the implementation of the APF method offline soon:

- *APF_method_offline* to compute the attractive and repulsive forces;
- *outer_loop_apf* for position tracking;
- *inner_loop* for attitude tracking;

In this section of the project, the obstacles are assumed as previously known and they were discretized into a series of points as seen in the previous chapter, and the attractive and the repulsive forces are computed during the execution. A first simulation has been conducted with the following parameters:

Params	x	y	z
K_P	6	6	6
K_V	4.7	4.7	4.7

Params	ϕ	θ	ψ
K_R	2	2.3	0.15
K_W	0.4	0.52	0.18

TABLE IV: Geometric control gain

Obstacles parameters are shown in (table: VI)

Params	value
k_a	0.5
k_r	0.7
$\eta_{o,i}$	2.5

TABLE V: APF parameters

Obstacle	x_c	y_c	r	h
obs_1	$3m$	$3m$	$0.7m$	$3m$
obs_2	$5m$	$5.5m$	$0.5m$	$7m$
obs_3	$8m$	$8m$	$0.2m$	$10m$

TABLE VI: Obstacles parameters

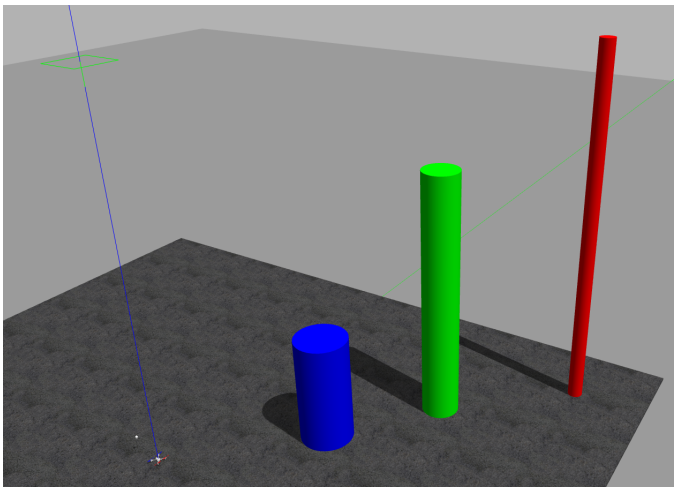
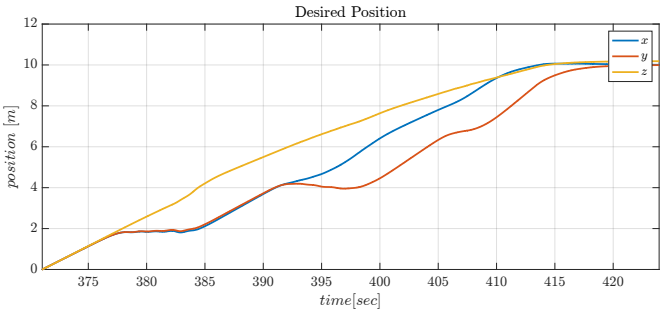
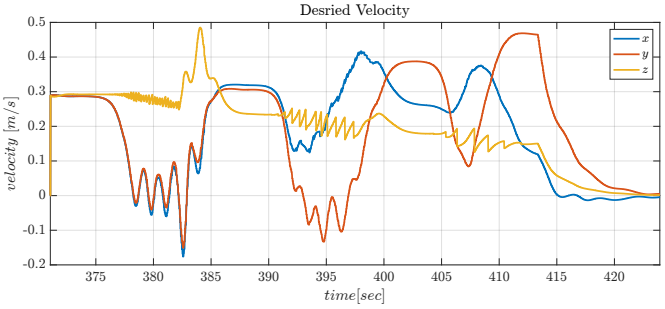


Fig. 26: Gazebo scenario

In the first simulation, the trajectory of the UAV started from $q_i = [0\ 0\ 0]$ and with goal position $q_f = [10\ 10\ 10]$ keeping a value of yaw $\psi = 0°$; the results of this simulation are shown in (fig: 27)





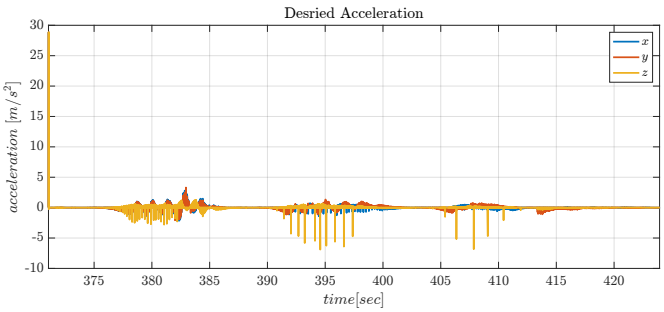


Fig. 27: Desired trajectory values

It can be seen that the desired values of velocity and acceleration present some spikes due to repulsive force as can be noticed in (fig: 28)

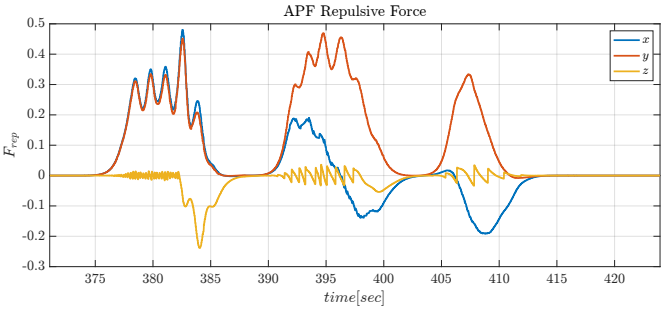
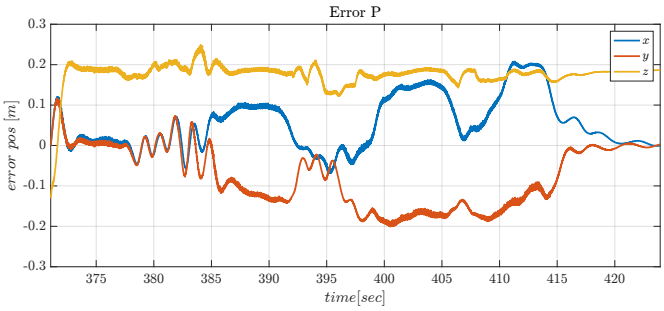
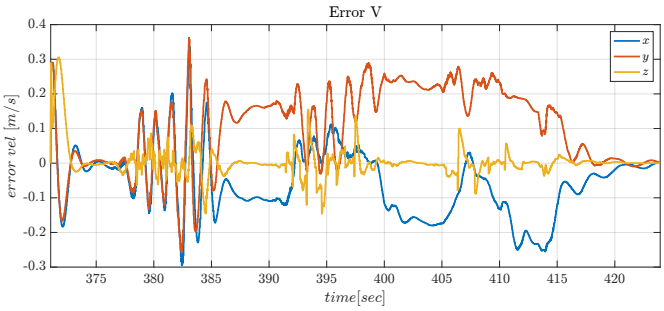


Fig. 28: APF repulsive force

Furthermore, it can be seen that the UAV reaches the desired position avoiding obstacles with a position error $e_z = 0.2m$ due to an orientation error (fig: 29), and for this reason convergence of the error to zero is not guaranteed.





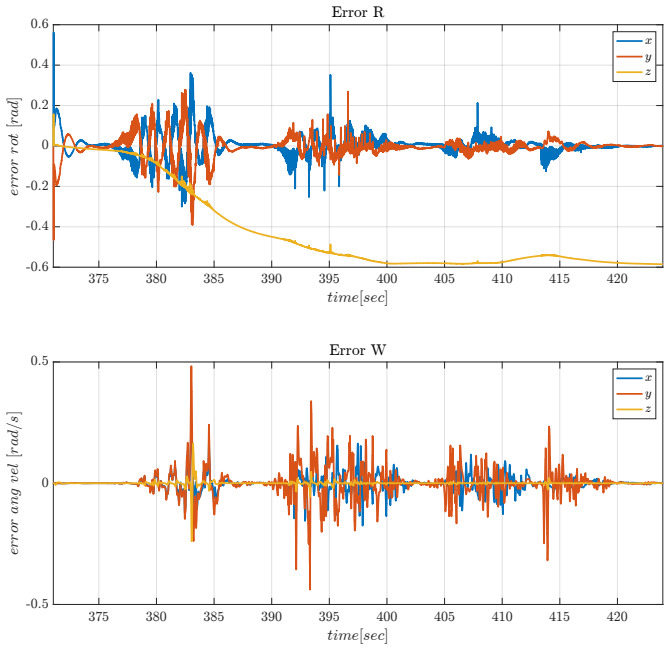
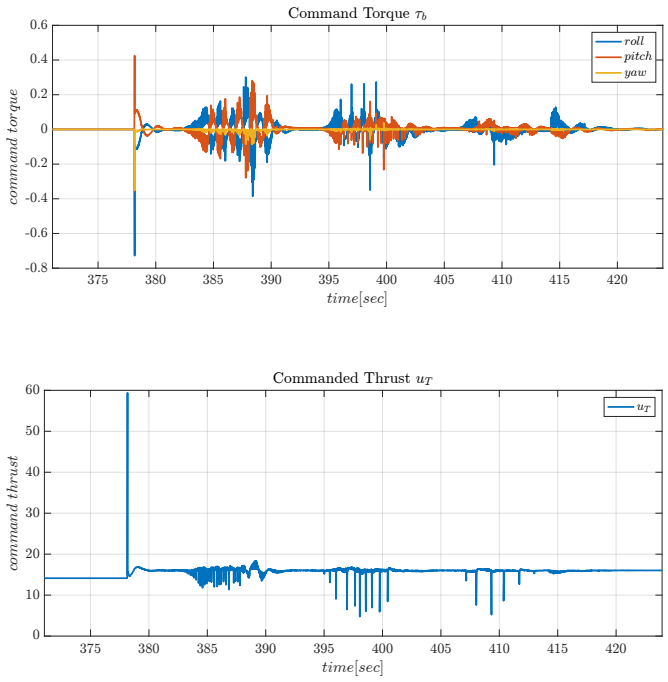


Fig. 29: Trajectory tracking errors

In the figure, the commands of u_T and τ^b and the velocity commands sent to the rotors are shown (fig: 30).



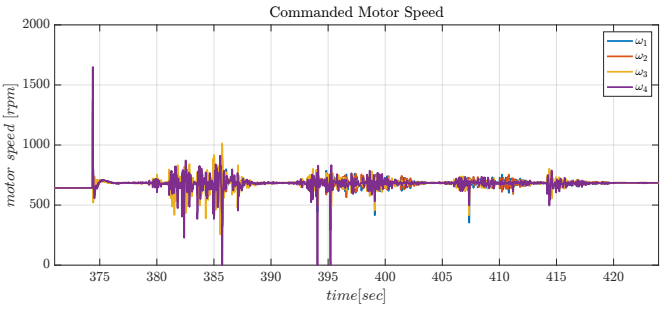


Fig. 30: Command

To reduce spikes into repulsive force and consequently into the desired velocity and acceleration profiles, a further simulation was performed, reshaping the computation of repulsive force, considering all points belonging to the obstacles and not only the closest to the drone. In this simulation, the repulsive force coefficient chosen is $k_r = 0.05$ with the same k_a, $\eta_{i,o}$ and same q_i and q_f. The results show a reduction of the spikes in the desired acceleration (fig: 31). This happens because, by considering all the points belonging to the obstacle, the F_{rep} is smoother than before (fig: 32).

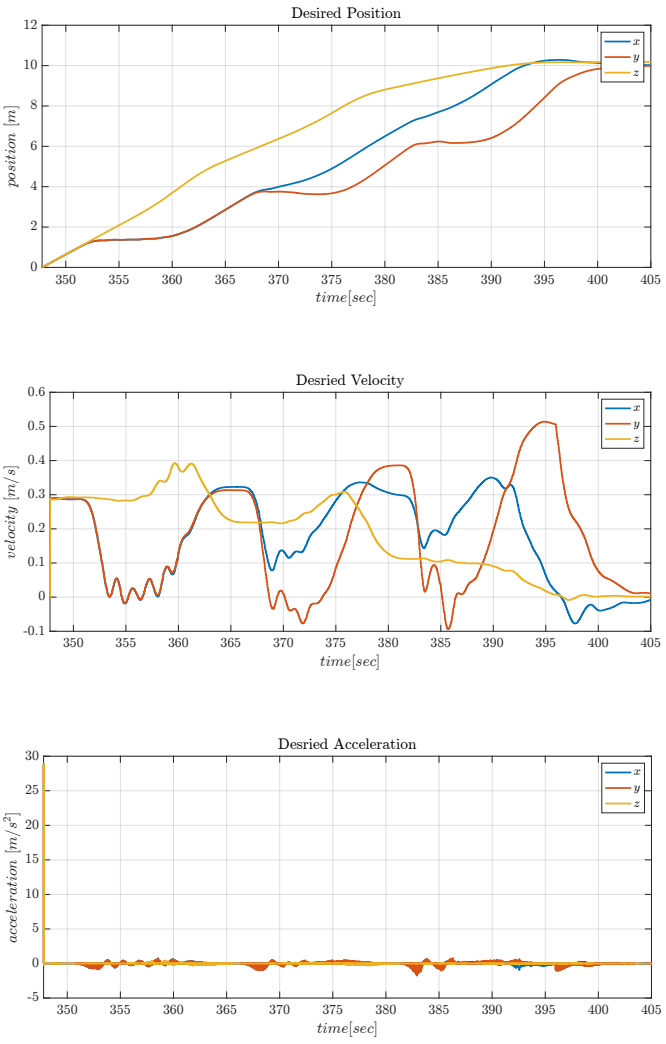


Fig. 31: Desired trajectory values

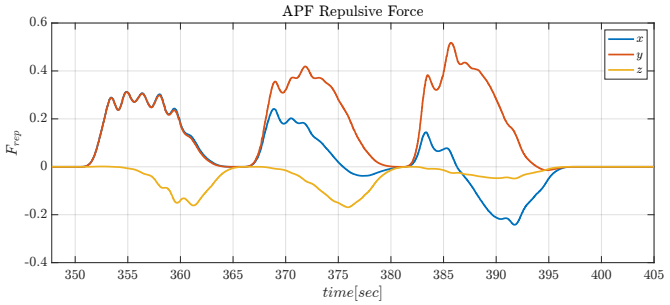


Fig. 32: APF repulsive force

With these changes, there are also smother commanded thrust and torque (fig: 33).

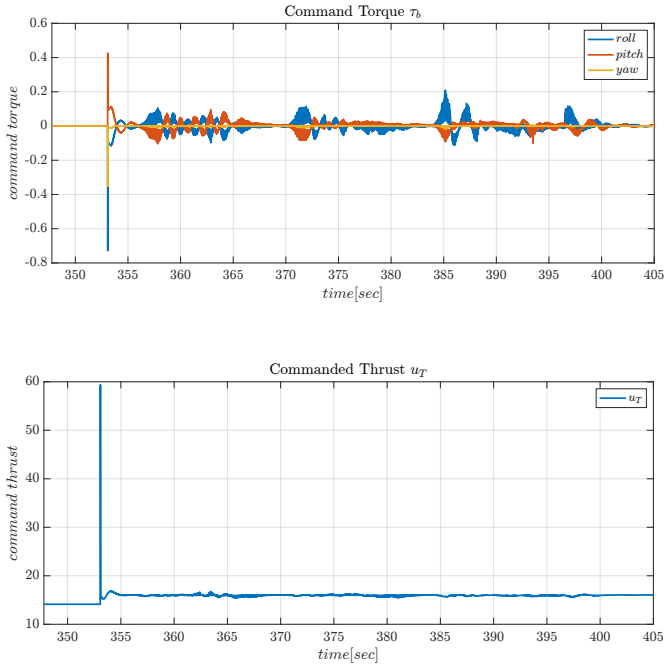


Fig. 33: Commands

B. Online APF

In the last part of the project, the APF method was implemented in the online version, using exteroceptive sensors, i.e. lidar. The lidar sensor was implemented using the "*libgazebo_ros_laser*" plugin offered by Gazebo with the specifications presented in (tab: VII)

Specification	Value
Update rate	$40Hz$
Samples	720
Angle range	$[-\pi, \pi]$
Radius range	$[0.25m, 30.0m]$
Resolution	$0.01m$

TABLE VII: Lidar specifications

In the online version of the APF, the informations about the obstacles are retrieved from the lidar scan as point clouds (fig: 34), in which are contained the coordinates of the points of the obstacles with respect to the position of the drone. These coordinates are passed as input of the APF method function that computes the repulsive force from the obstacles.

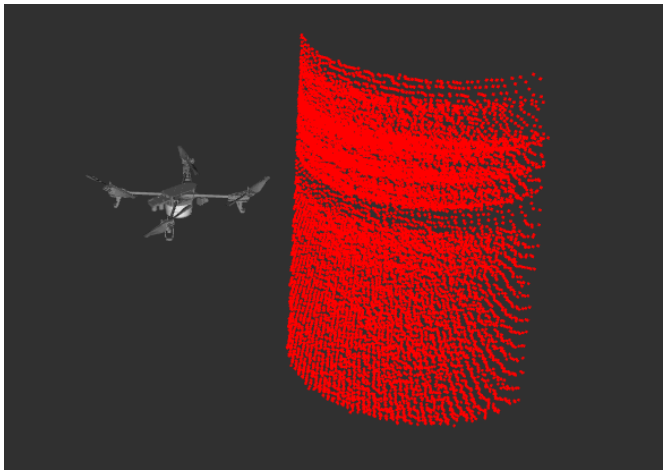


Fig. 34: Lidar laser scan

The following simulation was conducted with the online version of the APF with $k_r = 0.07$ and with $k_a = 0.5$ in the same Gazebo scenario previously seen.

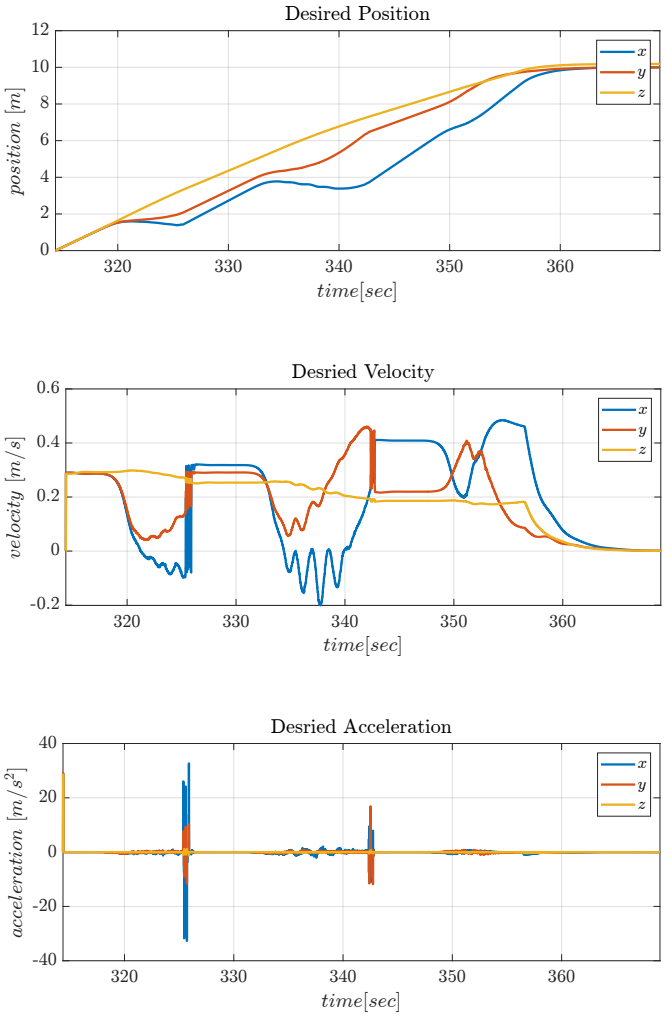


Fig. 35: Desired trajectory values

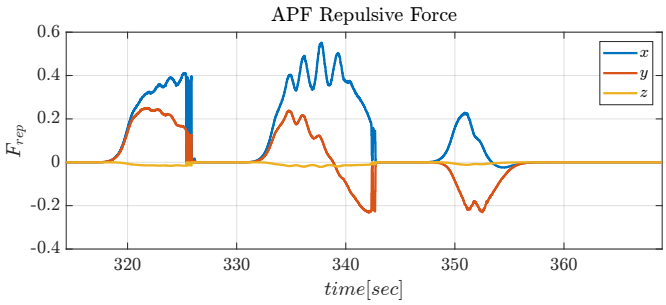
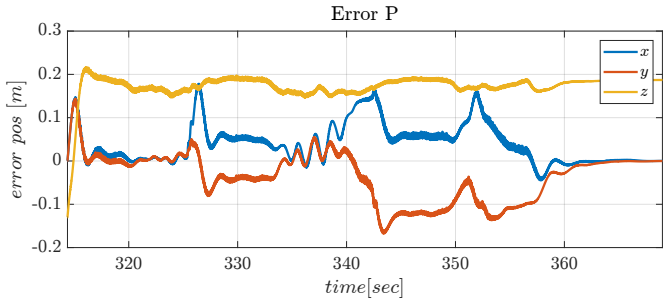


Fig. 36: APF repulsive force



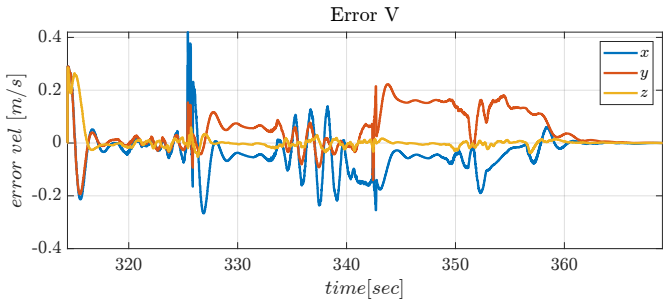
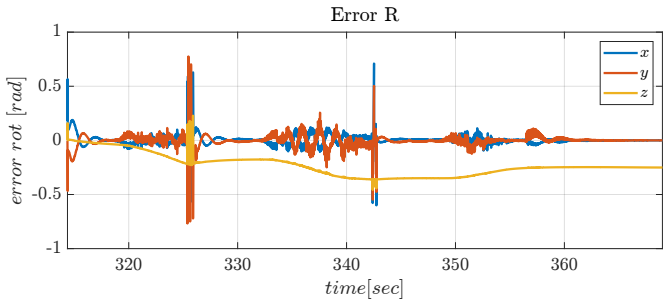


Fig. 37: Position tracking errors



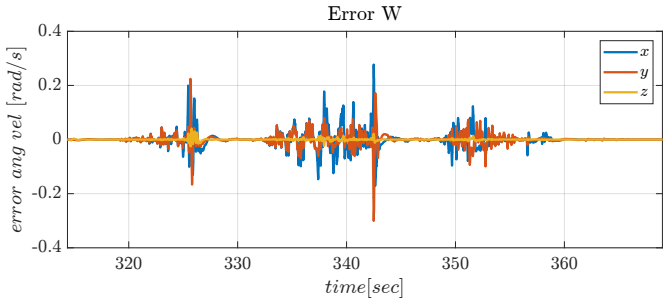
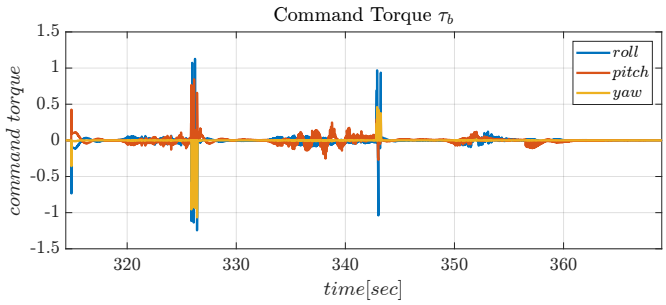
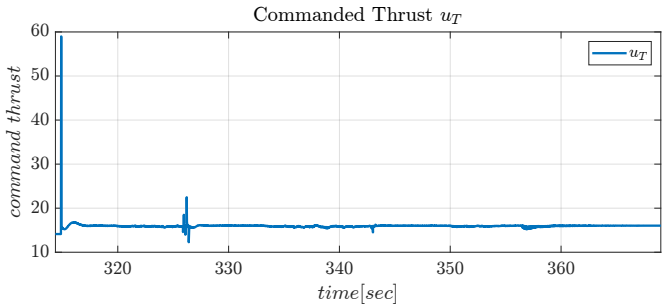


Fig. 38: Attitude tracking errors





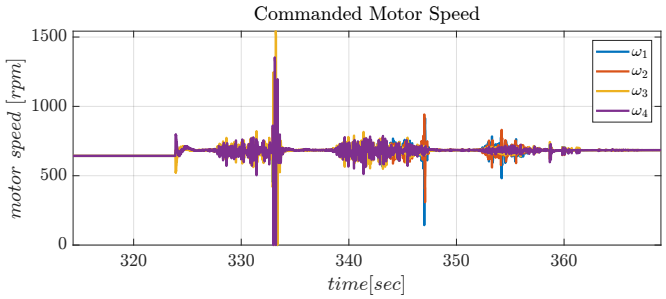


Fig. 39: Commands

The results obtained from this simulation show that the UAV successfully reaches the desired position while avoiding obstacles despite not knowing a priori the position of these in the environment; despite this, it can be observed that there are spikes in the desired velocity and acceleration profiles caused by the strong oscillations present in the repulsive force when the UAV overcomes an obstacle by climbing over it, whereas such problem does not arise where the obstacle is circumvented. This issue arises because, when the UAV rises above the height of the obstacle, due to the oscillations of the drone, the lidar recognizes points on the upper surface of the obstacle that, being very close to it, cause spikes in the repulsive force.

To confirm the functioning of the previously described algorithms in a more complex scenario, another simulation was conducted, changing the $k_r = 0.05$ and $\eta_{i,o} = 5.5$ to have smoother forces. The scenario chosen for this simulation is *test_city* (fig: 40), which features obstacles of various shapes that are difficult to represent mathematically.

Fig. 40: Gazebo world *test_city*

In this simulation, starting from the configuration $q = [0, 0, 0]$, the goal is first set to $q_{f,1} = [0, 0, 1]$, then to $q_{f,2} = [20, -15, 1]$, $q_{f,3} = [20, -13, 1]$, and finally $q_{f,4} = [30, -13, 1]$. In figure (fig: 41) the plots of the desired trajectory are shown.

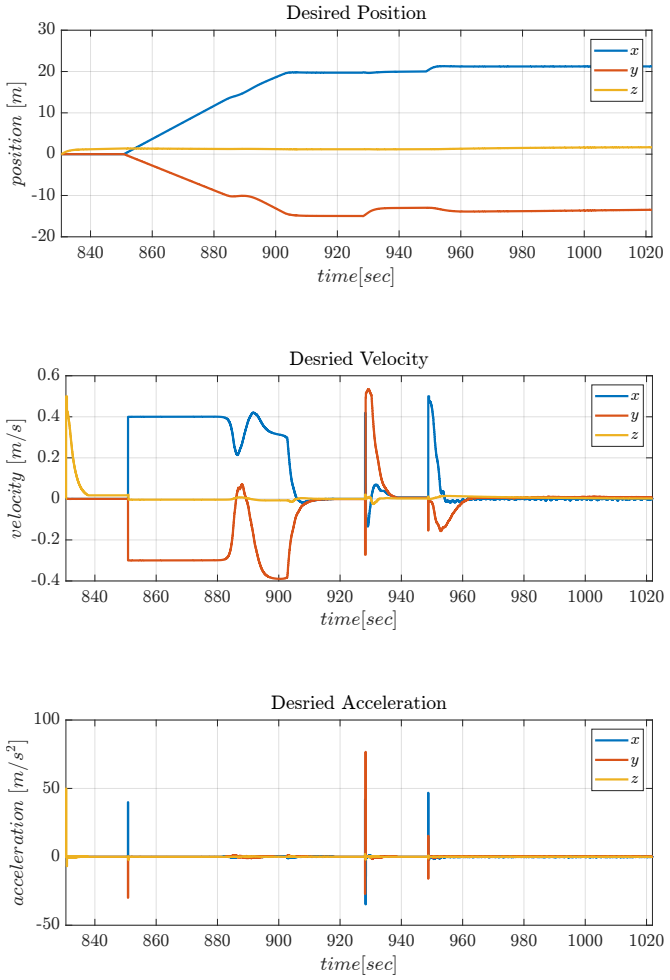


Fig. 41: Desired trajectory values

It is noted that the desired position generated by the APF never reaches the values $[30, -13, 2]$. This is due to the total force, which almost cancels out, generating an almost null desired velocity. The reason the total force is zero despite not reaching the desired configuration is the presence of a *local minima*. Indeed, by analyzing f_r, it is different from zero (fig 42). The final desired configuration is beyond the obstacle (fig 43), which prevents the UAV from advancing, despite being attracted to the goal.

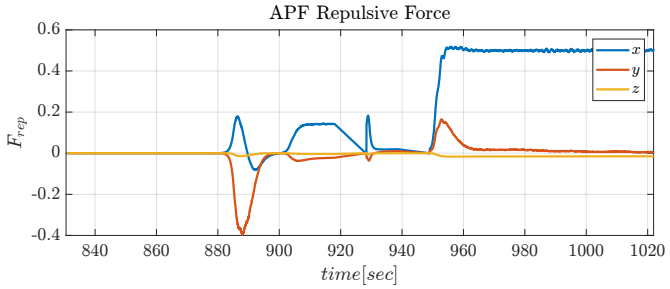


Fig. 42: Repulsive Force

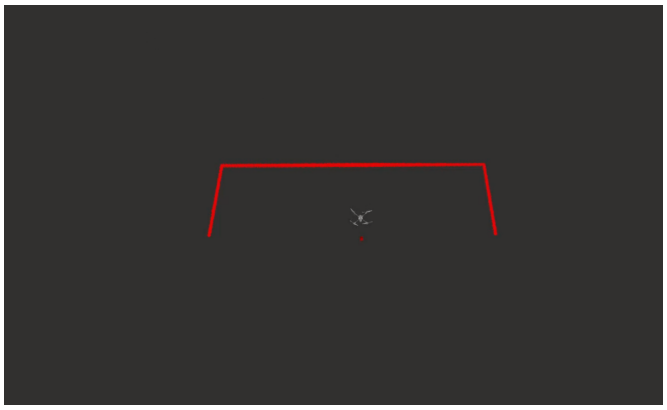


Fig. 43: Local minimal configuration in Rviz

From the analysis of tracking error (fig: 44), it's possible to see that also in this case there are constant errors along the z-axis.

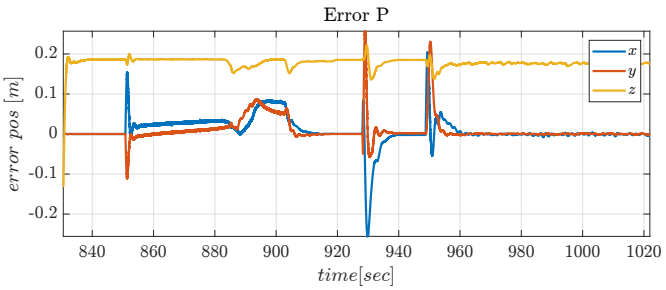


Fig. 44: Position tracking errors

To reduce this error in all ROS simulations, the real mass of the UAV was computed offline by analyzing the ROS bag file via Matlab, and the real mass was computed through the estimation of the external forces along the z-axis f_{ez}. Computing the real mass was conducted a simulation with the same gains and parameters of the last simulation and $q_0 = [0, 0, 0]$, $q_f = [0, 0, 2]$ and was estimated the external wrench via Matlab, the results are shown in figure (fig: 45).

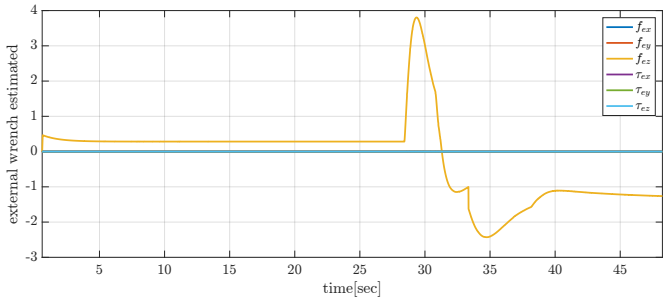


Fig. 45: Estimated wrench with $m = 1.5$

From the computing of the real mass through f_{ez} results $m_{real} = 1.6288$. Doing another ROS simulation by substituting the value of the computed mass, and analyzing (fig: 46)the results a new value of estimated m $m_{real} = 1.6335$.

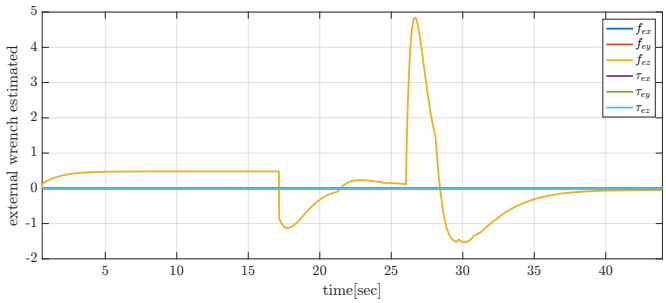


Fig. 46: Estimated wrench with $m = 1.6288$

To validate the computation of the real mass was evaluated a comparison between the results of the tracking error in the same configuration changing the mass value:

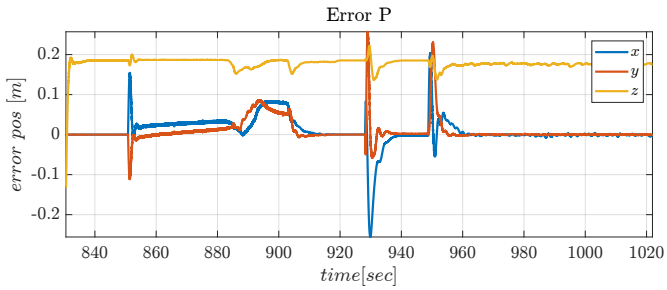


Fig. 47: Position tracking errors $m = 1.5$

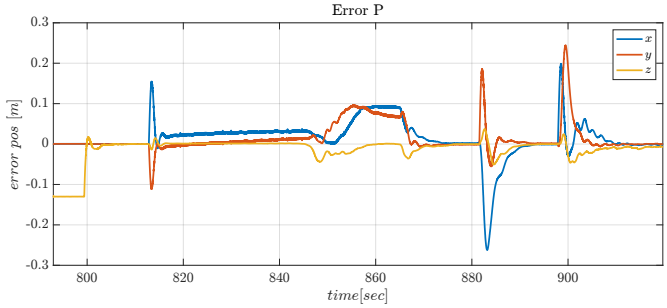


Fig. 48: Position tracking errors $m = 1.6335$

From the result shown in figure (fig: 47, 48), with the new computing mass, the steady-state error converges to zero.

VIII. Conclusion

This study presented the development and implementation of a geometric controller designed to manage both the position and attitude of an Unmanned Aerial Vehicle (UAV), with a strong emphasis on stability and reliability during flight. The integration of the Artificial Potential Field (APF) method for obstacle avoidance demonstrated a robust solution capable of addressing both preemptive path planning in offline mode and real-time adaptation using exteroceptive sensors in online mode.

The conducted simulations in two distinct environments, Simulink and ROS/Gazebo, provided comprehensive validation of the proposed system. Simulink allowed for the detailed modeling and simulation of dynamic controls, facilitating an in-depth analysis of system performance under controlled conditions. ROS/Gazebo offered a realistic and dynamic simulation environment where the UAV's behavior was rigorously tested against various obstacles, mimicking real-world scenarios.

The simulation results confirmed that the proposed control system maintains safe and precise navigation even when confronted with unexpected obstacles. The combination of geometric control and the APF method proved highly effective in collision avoidance, significantly enhancing the UAV's autonomous operation capabilities in complex and dynamic environments

References

[1] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[2] Alfian Ma'Arif, Wahyu Rahmaniar, Marco Antonio Márquez Vera, Aninditya Anggari Nuryono, Rania Majdoubi, and Abdullah Çakan. Artificial potential field algorithm for obstacle avoidance in uav quadrotor for dynamic environment. In *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pages 184–189, 2021.