UAV Person Following with Obstacle Avoidance via APF and VORTEX FIELD

Master in Artificial Intelligence and Robotics

Elective in Robotics

Students:

Francesco Barbetta

Giorgio De Santis

Roberto Passante

Marco Antonio Iossa

Academic year 2025-2026

Index

1 Introduction

Unmanned Aerial Vehicles (UAVs), and in particular quadrotor platforms, are increasingly employed in tasks that require close interaction with humans, such as surveillance, assistance, and monitoring. Among these applications, person following represents a challenging problem, as it requires the UAV to track a moving target while maintaining a safe and stable relative configuration, even in the presence of obstacles.

The main difficulty of person-following tasks lies in the simultaneous satisfaction of competing objectives**.** On one hand, the UAV must accurately follow the human target, preserving a desired relative distance and orientation to ensure observability and safety. On the other hand, the vehicle must react promptly to obstacles in the environment, avoiding collisions without inducing unstable or oscillatory behaviors that may compromise tracking performance or flight safety.

In this project, we address the problem of person following with obstacle avoidance for a quadrotor UAV operating in a simulated environment. The human target moves along a predefined trajectory, while the UAV autonomously adjusts its motion to follow the target and avoid static obstacles detected through onboard proximity sensors. The control strategy is designed to be fully reactive, without relying on global path planning or map reconstruction.

Obstacle avoidance is achieved through a vortex-based field approach, combining repulsive forces that push the UAV away from obstacles with tangential components that guide the motion around them.

The goal of this work is to develop a robust and smooth control architecture that enables stable person following while safely navigating cluttered environments.

2 System Modelling

2.1 Quadrotor Dynamic Model

The quadrotor is modeled as a rigid body moving in three-dimensional space. Its dynamics can be described by a nonlinear state–space model of the form:

$$\dot{\xi} = f(\xi) + g(\xi)u$$

where the state vector is defined as:

$$\xi = \left(x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r\right)^T$$

with (x, y, x)denoting the position of the center of mass in the inertial frame, $\left(v_x, v_y, v_z\right)$the linear velocities, $(\phi, \theta, \psi)$the roll, pitch, and yaw angles, and $(p, q, r)$the angular velocities expressed in the body frame.

The control input vector is given by:

$$u = \left(T, \tau_\phi, \tau_\theta, \tau_\psi\right)^T$$

where $T$is the total thrust generated by the propellers and $\tau_\phi, \tau_\theta, \tau_\psi$are the roll, pitch, and yaw torques.

2.2 Translational Dynamics

The translational motion of the quadrotor is described by the dynamics of its center of mass expressed in the inertial frame. Let
$\mathbf{p} = [x\ y\ z]^T$denote the position of the vehicle and
$\mathbf{v} = \dot{\mathbf{p}}$ its linear velocity.

The translational dynamics are governed by Newton's second law:

$$m\ddot{\mathbf{p}} = \mathbf{F}$$

where m is the total mass of the quadrotor and **F** represents the sum of all external forces acting on the vehicle.

The dominant forces considered in this model are:

- the gravitational force,
- the total thrust generated by the propellers.

Aerodynamic drag, wind disturbances, and ground effects are neglected.

The total thrust T is generated along the body-fixed vertical axis and must be expressed in the inertial frame through the rotation matrix $R \in SO(3)$, which

maps vectors from the body frame to the world frame. The resulting force balance is:

$$m\ddot{\mathbf{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$$

where g is the gravitational acceleration.

Expanding the equation, the translational acceleration depends on both the magnitude of the total thrust and the current attitude of the quadrotor. In particular, roll and pitch angles indirectly affect the horizontal motion by tilting the thrust vector, while the vertical motion is directly influenced by the thrust magnitude.

This formulation highlights the intrinsic coupling between translational and rotational dynamics: horizontal accelerations are achieved by reorienting the thrust direction rather than by generating independent horizontal forces.

In the implemented simulation model, the translational dynamics are evaluated implicitly by the physics engine of the simulator, which applies the thrust-generated forces to the rigid body at each simulation step.

2.3 Rotational Dynamics

The rotational motion of the quadrotor describes the evolution of its attitude, which is defined by the orientation of the body frame with respect to the inertial frame. The attitude is commonly represented using Euler angles $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^T$, corresponding to roll, pitch, and yaw.

The rotational dynamics of the quadrotor are governed by the rigid-body rotational equation:

$$J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (J\boldsymbol{\omega}) = \boldsymbol{\tau}$$

where:

- $J \in \mathbb{R}^{3\times3}$is the inertia matrix of the vehicle,
- $\boldsymbol{\omega} = [p\ q\ r]^T$is the angular velocity vector expressed in the body frame,
- $\boldsymbol{\tau} = [\tau_\phi\ \tau_\theta\ \tau_\psi]^T$represents the control torques acting on the quadrotor.

The control torques are generated by differential thrusts among the four propellers. Roll and pitch torques are produced by unbalanced thrusts between opposite propellers, while the yaw torque is generated by exploiting the reaction torques associated with the counter-rotating propellers.

The relationship between angular velocities and Euler angle rates is given by:

$$\dot{\boldsymbol{\eta}} = W(\boldsymbol{\eta})\boldsymbol{\omega}$$

where $W(\boldsymbol{\eta})$is a configuration-dependent transformation matrix.

2.4 Actuation and Force Generation

The quadrotor is actuated by four independent propellers.
Each propeller generates an upward force proportional to the square of its rotational speed.

$$T_i = k_f \omega_i^2$$

The total thrust acting on the vehicle is defined as:

$$T = \sum_{i=1}^{4} T_i$$

where $T_i$denotes the thrust produced by the i-th propeller.

Control torques are obtained by applying differential thrusts between the propellers, allowing the quadrotor to control its attitude around the three rotational axes.

In the implemented control architecture, the high-level controller computes a collective thrust and attitude correction terms. These quantities are then mapped to individual propeller commands using a standard mixing strategy. In the simulation environment CoppeliaSim, thrust commands are directly converted into forces and torques applied to each propeller through the simulator API.

The propeller dynamics are not explicitly modeled. Instead, each propeller applies:

- a force along its local vertical axis, proportional to the commanded thrust,
- a reaction torque around the same axis, accounting for the rotor spinning direction.

The controller computes three corrective terms:

- roll correction α_{Corr},
- pitch correction β_{Corr},
- yaw correction rotCorr,

which are then combined with the total thrust to generate the individual propeller commands.

The following code snippet shows how thrust commands are applied to each propeller within the simulator:

```
handlePropeller(1, thrust * (1 - alphaCorr + betaCorr + rotCorr))
handlePropeller(2, thrust * (1 - alphaCorr - betaCorr - rotCorr))
handlePropeller(3, thrust * (1 + alphaCorr - betaCorr + rotCorr))
handlePropeller(4, thrust * (1 + alphaCorr + betaCorr - rotCorr))
```

2.5 Sensors and Assumptions

The quadrotor perceives the environment through a set of eight proximity sensors mounted around the vehicle body: North, South, East, West and the four diagonal directions (NE, NW, SE, SW).
At each control cycle, all sensors are read and the returned distance is used to detect nearby obstacles within a predefined influence range d_0. For each valid detection, the measured point (initially expressed in the sensor frame) is transformed into the world frame using the sensor pose, enabling a consistent geometric interpretation of obstacle location.

To avoid false detections caused by the quadrotor itself, a self-filtering step discards readings generated by components belonging to the UAV structure (e.g., propellers and body elements). Only external objects are therefore considered for obstacle avoidance.

In our implementation we assume ideal sensing: measurements are instantaneous and noise-free, and a detected point corresponds to the closest obstacle along the sensor beam.

```
local sensorNames = {
    "/Quadcopter/proximitySensor_N",  "/Quadcopter/proximitySensor_S",
    "/Quadcopter/proximitySensor_E",  "/Quadcopter/proximitySensor_W",
    "/Quadcopter/proximitySensor_NE", "/Quadcopter/proximitySensor_NW",
    "/Quadcopter/proximitySensor_SE", "/Quadcopter/proximitySensor_SW"
}
for i=1,#sensorNames,1 do
    sensorHandles[i] = sim.getObject(sensorNames[i])
end
```

2.6 Target Modeling and Relative Pose

The target (Bill) is modeled as a kinematic point mass moving along a predefined trajectory in the environment.
Rather than tracking the absolute position of the target, the quadrotor is required to maintain a desired relative pose with respect to it.

At each simulation step, the target position $\mathrm{p}_t \in \mathbb{R}^3$ and its yaw angle ψ_t are retrieved from the simulator. The target yaw defines the heading direction and is used to specify the relative position of the quadrotor.

The desired relative position of the quadrotor is defined as a fixed offset behind the target along its heading direction, with an additional vertical displacement. The desired position is given by:

$$\mathrm{p}_d = \mathrm{p}_t - d_f \begin{bmatrix} \cos\psi_t \\ \sin\psi_t \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}$$

where:

- d_f is the desired following distance,
- h is the desired altitude offset.

This formulation ensures that the quadrotor remains behind the target while preserving a constant vertical separation.

The yaw reference for the quadrotor is chosen equal to the target yaw:

$$\psi_d = \psi_t$$

```
local targetOri = sim.getObjectOrientation(targetObj, -1)
local yaw = targetOri[3]

local follow_dist = 1.2
local lateral_offset = 0.0

local ex = math.cos(yaw)
local ey = math.sin(yaw)
local ex_lat = -math.sin(yaw)
local ey_lat =  math.cos(yaw)

local virtualTargetPos = {
    targetPos[1] - follow_dist * ex + lateral_offset * ex_lat,
    targetPos[2] - follow_dist * ey + lateral_offset * ey_lat,
    desiredZ
}
```

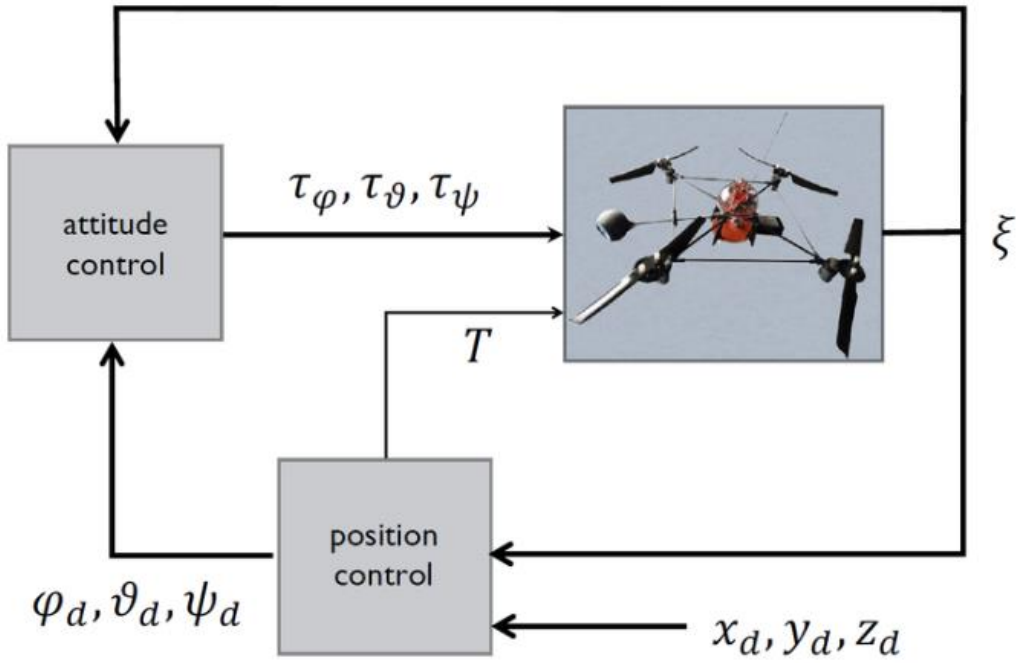
3 Control Architecture Overview

The quadrotor's flight is managed by a hierarchical control suite that translates position errors into the motor mixing terms ($\alpha_{\mathrm{Corr}}, \beta_{\mathrm{Corr}}$) defined in Section 2.4. The quadrotor is controlled using a hierarchical cascaded control architecture, where high-level position errors are converted into low-level attitude and thrust commands.

The overall structure is composed of an outer position loop and an inner attitude loop, with the physical dynamics fully handled by the CoppeliaSim simulator. The control system is organized into three main components:

- Altitude control (outer loop, vertical axis)
- Horizontal position control (outer loop → attitude setpoints)
- Attitude and heading control (inner loop)

This separation allows slow position dynamics to be handled independently from fast attitude stabilization, improving robustness and stability.



3.1 Altitude Control (Height Control)

The vertical dynamics are controlled by a PID (Proportional-Integral-Derivative) regulator. This controller computes the total thrust required to maintain the UAV at a desired height $+1.5$.

$$T = g + K_p e_z + K_i \int e_z \, dt + K_d \Delta e_z$$

The term 9.81 represents the feedforward gravity compensation, while l[3]*vParam provides the derivative action based on the actual vertical velocity retrieved from the simulator.

```
local thrust = 9.81 + pParam*e + iParam*cumul + l[3]*vParam
```

This specific configuration is crucial for person-following, as it minimizes the "bouncing" effect when the human target changes elevation abruptly.

3.2 Rotational Controller (PD Heading)

The rotational loop ensures the quadrotor faces the target at all times. It uses a PD (Proportional-Derivative) logic to compute the yaw correction term (referred to as rotCorr in the code).

```
local euler = sim.getObjectOrientation(d, targetObj)
local rotCorr = euler[3]*0.1 + 2*(euler[3] - prevEuler)
prevEuler = euler[3]
```

- **Proportional term:** Reduces the angular gap between the UAV's heading and the target's orientation.
- **Derivative term:** Acts as an angular damper, reducing the rotational velocity as the error approaches zero to prevent overshoot.

3.3 Horizontal Controller and Setpoint Mapping

Horizontal motion is not controlled directly through forces or accelerations. Instead, the outer position loop computes desired roll and pitch angles by mapping position errors into attitude setpoints**.**

The relative target position is first expressed in the drone's body frame**:**

$$sp = T_{WD}^{-1} \cdot p_{target}$$

Using a small-angle linear approximation, the position error is converted into desired tilt angles:

$$\theta_d = k_p\, sp_y + k_d\, \dot{sp}_y$$
$$\phi_d = k_p\, sp_x + k_d\, \dot{sp}_x$$

This approach allows the UAV to translate horizontal position errors into roll and pitch commands, which are then tracked by the inner attitude controller.
No explicit dynamic model is used at this level; the controller relies on attitude modulation to generate horizontal motion.

```
alphaCorr = alphaCorr + sp[2]*0.01 + 0.5*(sp[2] - psp2)
betaCorr  = betaCorr  - sp[1]*0.01 - 0.5*(sp[1] - psp1)
```

3.4 Attitude Control (Inner Loop – Roll & Pitch Stabilization)

The inner loop is responsible for stabilizing the quadrotor around the desired roll and pitch angles generated by the outer position loop.

The attitude controller computes correction terms based on angle errors:

$$e_\theta = \theta_d - \theta, e_\phi = \phi_d - \phi$$
$$\alpha_{corr} = K_{p\theta} e_\theta + K_{d\theta} \dot{e}_\theta$$
$$\beta_{corr} = K_{p\phi} e_\phi + K_{d\phi} \dot{e}_\phi$$

This fast inner loop ensures accurate tracking of the desired attitude and rejects disturbances caused by external forces or rapid target motion.

```
local alphaE = (vy[3] - m[12])
local alphaCorr = 0.25*alphaE + 1.5*(alphaE - pAlphaE)
local betaE  = (vx[3] - m[12])
local betaCorr = -0.25*betaE - 1.5*(betaE - pBetaE)
pAlphaE = alphaE; pBetaE = betaE
```

3.5 Target Trajectory and Follow Strategy

This chapter analyzes the generation of the target's movement (Bill) and the control logic used by the quadrotor to maintain its relative pose during the mission. Unlike a fixed trajectory, the system described here involves a target that autonomously navigates the environment while avoiding static obstacles.

3.6 Path Generation via OMPL (RRT-Connect)

Bill's movement is not a simple pre-recorded animation but the result of a motion planning process performed via the OMPL (Open Motion Planning Library).

The script utilizes the RRT-Connect algorithm, a variant of *Rapidly-exploring Random Trees* designed to quickly connect two configurations (start and goal). To ensure test repeatability and data validation, a fixed seed was introduced for the random number generator:

```
local fixedSeed = 5678
math.randomseed(fixedSeed)
```

```
local t=simOMPL.createTask('t')
simOMPL.setAlgorithm(t,searchAlgo)
local ss={simOMPL.createStateSpace('2d',simOMPL.StateSpaceType.position2d,BillHandle,{minX,minY},{maxX,maxY},1)}
```

The task calculates a collision-free path by considering Bill's bounding volume relative to the collection of obstacles present in the scene.

3.7 Target Kinematics and Smooth Animation

Once the path is obtained as a series of waypoints, the script transforms it into a continuous trajectory. To prevent jerky movements, **Quadratic Bezier Interpolation** is used to smooth the corners of the planned path:

```
local p=sim.getPathInterpolatedConfig(path,pathLengths,dist,{type='quadraticBezier',strength=0.25},{0,0})
```

In addition to translation, the script manages Bill's orientation so that it remains consistent with the direction of motion (tangent to the path), calculating the angle using the arctangent function:

```
local dx=p[1]-pp[1]
local dy=p[2]-pp[2]
if math.sqrt(dx*dx+dy*dy) > 0.01 then
    sim.setObjectOrientation(BillHandle,{0,0,math.atan2(dy,dx)})
end
```

Finally, the moveBody(dist) function synchronizes the kinematic joints (legs, arms) in proportion to the distance traveled, ensuring the visual realism of the simulation.

3.8 Quadrotor Follow Strategy

The drone must react in real-time to Bill's movements. The following strategy is based on the generation of a Virtual Setpoint located at a constant distance (d = 1.1 m) and a fixed height (h = 1.5m) relative to the target.

As Bill follows the path planned by RRT-Connect, the drone continuously calculates its desired position:

```
local ex = math.cos(yaw)
local ey = math.sin(yaw)
local ex_lat = -math.sin(yaw)
local ey_lat =  math.cos(yaw)

local virtualTargetPos = {
    targetPos[1] - follow_dist * ex + lateral_offset * ex_lat,
    targetPos[2] - follow_dist * ey + lateral_offset * ey_lat,
    desiredZ
}
```

3.9 Coordinate Transformation in the Local Frame

The fundamental step for the drone's stability is the transformation of the setpoint from the World Frame (global) to the Body Frame (local). This allows the PID

controllers (described in Chapter 3) to operate on errors relative to the drone's physical axes:

The resulting vector sp contains the longitudinal error (sp[1]) and the lateral error (sp[2]). If Bill performs a sharp turn to avoid an obstacle, the mDrone matrix instantly captures the variation, translating it into a Roll or Pitch command for the drone. This allows the vehicle to "slide" laterally or accelerate to maintain its position behind the target.

```
local mDrone = sim.getObjectMatrix(d, -1)
sim.invertMatrix(mDrone)
local sp = sim.multiplyVector(mDrone, virtualTargetPos)
```

4 Obstacle Avoidance Vortex Field & Heuristics

4.1 Obstacle Avoidance via Repulsive Vortex Fields

Obstacle avoidance is achieved through a potential-field-based approach that modifies the reference motion of the quadrotor whenever obstacles are detected in its vicinity.
Differently from classical Artificial Potential Field (APF) methods, no attractive force toward the goal is used. The quadrotor follows the nominal reference trajectory, while obstacle avoidance is handled exclusively through repulsive forces.

Let $p \in \mathbb{R}^3$be the quadrotor position and $p_o \in \mathbb{R}^3$the position of an obstacle. The distance between them is defined as:

$$d(p, p_o) = \| p - p_o \|$$

A repulsive field is activated when $d(p, p_o) \leq d_0$, where d_0 represents the influence radius of the obstacle.
The total repulsive force is defined as:

$$\mathbf{F}_r = \mathbf{F}_{rt} + \mathbf{F}_{rr}$$

where:

- $\mathbf{F}_{rt}$is the translational repulsive component,
- $\mathbf{F}_{rr}$is the rotational (vortex) repulsive component.

The translational component pushes the quadrotor directly away from the obstacle and is defined as:

$$\mathbf{F}_{rt}(p) = \begin{cases} k_{rt}\left(\frac{1}{d(p,p_o)} - \frac{1}{d_0}\right)\frac{1}{d(p,p_o)^3}(p - p_o), & d \le d_0 \\ 0, & d > d_0 \end{cases}$$

This term alone, however, may lead to oscillatory behavior and local minima, especially when obstacles lie along the nominal direction of motion.

4.2 Rotational Vortex Field and Local Minima Avoidance

To overcome the limitations of purely translational repulsive fields, a rotational component is introduced, generating a vortex-like behavior around obstacles.

The rotational repulsive force is defined as:

$$\mathbf{F}_{rr}(p) = \begin{cases} k_{rr}\left(\frac{1}{d(p,p_o)} - \frac{1}{d_0}\right)\frac{1}{d(p,p_o)^3}\, R\,(p - p_o), & d \le d_0 \\ 0, & d > d_0 \end{cases}$$

where:

- $k_{rr}$is the rotational gain,
- $R \in \mathbb{R}^{2\times 2}$is a planar rotation matrix that determines the direction of rotation (clockwise or counterclockwise).

The direction of rotation is chosen based on the relative angle between:

- the quadrotor velocity direction φ,
- the direction toward the obstacle centroid ρ.

Let:

$$\theta = \varphi - \rho$$

The rotation matrix is defined as:

$$R = \begin{cases} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, & \theta \ge 0 \\ \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, & \theta < 0 \end{cases}$$

This construction ensures that the repulsive force induces a tangential motion around the obstacle, preventing the quadrotor from:

- getting stuck in local minima,

- oscillating along a single axis,
- repeatedly entering and exiting the repulsive region.

```
-- repulsione
local F_rep = {-u_obs[1]*k_rep*strength, -u_obs[2]*k_rep*strength}

-- verso di rotazione
local vec_to_target = {virtualTargetPos[1]-q[1], virtualTargetPos[2]-q[2]}
local cross_product = (vec_to_obs[1]*vec_to_target[2]) - (vec_to_obs[2]*vec_to_target[1])
local rot_dir = 1
if cross_product < 0 then rot_dir = -1 end

-- vortice (perpendicolare)
local F_vor = {-u_obs[2]*k_vortex*strength*rot_dir, u_obs[1]*k_vortex*strength*rot_dir}

local F_total_world = {F_rep[1]+F_vor[1], F_rep[2]+F_vor[2], 0}
```

In the implemented system, the vortex field is computed at each control step by:

- evaluating the distance to detected obstacles,
- computing the translational and rotational repulsive components,
- summing them to obtain the total repulsive force.

This force is then used to modify the reference velocity vector, ensuring smooth obstacle circumnavigation while preserving trajectory continuity.

The gains k_{rt}, k_{rr} and the activation radius $d_0$are tuned empirically to balance safety margins and trajectory deviation.

4.3 Repulsive Field Formulation

To prevent collisions, each obstacle generates a repulsive force acting on the quadrotor whenever the distance from the obstacle falls within a predefined influence region.
The repulsive field is designed to push the vehicle away from obstacles while remaining smooth and bounded to avoid instability.

Let $\mathbf{q} \in \mathbb{R}^2$ be the horizontal position of the quadrotor and $\mathbf{q}_o \in \mathbb{R}^2$the position of the detected obstacle.
The relative distance vector is defined as:

$$\mathbf{d} = \mathbf{q}_o - \mathbf{q}, d = \| \mathbf{d} \|$$

The corresponding unit direction vector is:

$$\mathbf{u}_{obs} = \frac{\mathbf{d}}{d}$$

The repulsive force magnitude is activated only when the obstacle lies inside the influence radius d_0.

To avoid singularities near the obstacle, the effective distance is saturated by a minimum threshold $d_{\min}$.

A smooth activation function $\sigma(d) \in [0,1]$ is introduced:

$$\sigma(d) = \frac{d_0 - d}{d_0 - d_{\min}}$$

The repulsive force is then defined as:

$$\mathbf{F}_{rep} = -k_{rep}\, \sigma(d)^3\, \mathbf{u}_{obs}$$

The cubic dependence ensures a progressive growth of the force as the quadrotor approaches the obstacle, improving numerical stability and avoiding abrupt accelerations.

4.4 Vortex (Tangential) Field Formulation

While a pure repulsive field guarantees collision avoidance, it often leads to oscillatory behavior or local minima when navigating around obstacles.
To overcome this limitation, a vortex field is added, generating a tangential force around the obstacle.

The vortex force is orthogonal to the repulsive direction and induces a circular motion that guides the quadrotor smoothly around obstacles.

Given the obstacle direction unit vector $\mathbf{u}_{obs} = [u_x, u_y]^T$, the tangential direction is obtained by a $90°$ rotation:

$$\mathbf{u}_{tan} = \begin{bmatrix} -u_y \\ u_x \end{bmatrix}$$

The vortex force is defined as:

$$\mathbf{F}_{vor} = k_{vor}\, \sigma(d)^3\, \rho\, \mathbf{u}_{tan}$$

where $\rho \in +1, -1$determines the direction of rotation (clockwise or counter-clockwise).

The rotation direction is chosen using the sign of the planar cross product between:

- the obstacle direction vector,

- the direction toward the virtual target.

This choice ensures that the quadrotor bypasses the obstacle on the side that best preserves motion toward the target, as proposed in the reference paper.

The final obstacle-induced force acting on the quadrotor is obtained by combining the repulsive and vortex components:

$$\mathbf{F}_{obs} = \mathbf{F}_{rep} + \mathbf{F}_{vor}$$

This formulation preserves safety through repulsion while ensuring smooth obstacle circumnavigation via the tangential component.

```
local F_total_world = {F_rep[1]+F_vor[1], F_rep[2]+F_vor[2], 0}
```

4.5 Smooth Activation and Blending of the Vortex Field

In the proposed obstacle avoidance strategy, the vortex field is not activated abruptly. Instead, a continuous blending mechanism is adopted to smoothly combine target tracking and obstacle avoidance actions. This approach prevents discontinuities in the commanded forces and significantly reduces oscillatory behaviors.

A blending coefficient $w \in [0,1]$ is computed as a function of the minimum distance from the closest detected obstacle. Two thresholds are defined: an activation distance d_{on} and a deactivation distance d_{off}, with $d_{\text{off}} > d_{\text{on}}$. Within this range, the avoidance contribution is progressively increased.

The raw blending factor is computed as:

$$w_{\text{raw}} = \frac{d_{\text{off}} - (d_{min} + d_{\text{soft}})}{d_{\text{off}} - d_{\text{on}}}$$

and then saturated to the interval $[0, 1]$:

$$w = \text{clamp}(w_{\text{raw}}, 0,1)$$

To further smooth the activation, a cubic shaping is applied:

$$w \leftarrow w^3$$

This nonlinear shaping ensures a gradual growth of the avoidance effect when approaching an obstacle, while preserving smooth behavior at the boundaries.

The blending coefficient is also used to attenuate the target-tracking setpoint in the body frame of the quadrotor, preventing the tracking objective from conflicting with the avoidance action:

$$s_p \leftarrow (1 - k_w\, w)\, s_p$$

This design allows the quadrotor to maintain stable motion around obstacles while preserving the desired relative configuration with respect to the target.

```
-- blending continuo (0..1) basato su distanza
local w = 0.0
if obstacle_found then
    local d_soft = 0.4      -- banda di ingresso morbida (m)
    local w_raw = (d_off - (closest_dist_found + d_soft)) / (d_off - d_on)
    w = clamp(w_raw, 0.0, 1.0)
    w = w^3
end
```

4.6 Stabilization Mechanisms: Smoothing, Saturation, and Hysteresis

Although vortex fields provide effective obstacle circumnavigation, they may induce large lateral accelerations and oscillatory motion if not properly regulated. For this reason, several stabilization mechanisms are integrated into the control architecture.

Force smoothing
The lateral forces generated by the vortex field are filtered using an exponential smoothing scheme, equivalent to a discrete-time low-pass filter:

$$F_k = (1 - \alpha)\, F_{k-1} + \alpha F_k^{raw}$$

This filtering reduces sensitivity to sensor noise and rapid variations in obstacle measurements.

```
-- raw force: migliore ostacolo
local raw_Fx = bestFx
local raw_Fy = bestFy

-- smoothing
last_Fx = (last_Fx * (1.0 - smoothing_factor)) + (raw_Fx * smoothing_factor)
last_Fy = (last_Fy * (1.0 - smoothing_factor)) + (raw_Fy * smoothing_factor)
```

Force saturation.
To ensure bounded control inputs, the filtered forces are limited to a maximum admissible magnitude:

$$\boldsymbol{F} \in [-F_{max}, F_{max}]$$

This prevents excessive accelerations and improves overall stability.

```
-- saturazione forza
last_Fx = clamp(last_Fx, -max_force, max_force)
last_Fy = clamp(last_Fy, -max_force, max_force)
```

Hysteresis logic.
To avoid frequent switching of the avoidance mode when the obstacle distance is close to the activation threshold, a hysteresis mechanism is introduced. The avoidance mode is activated only when the obstacle distance falls below d_{on}, and it is deactivated only when the distance exceeds d_{off}. This separation between activation and deactivation thresholds significantly reduces chattering effects.

Together, these mechanisms ensure that the vortex-based obstacle avoidance remains smooth, stable, and compatible with the target-following task, even in cluttered environments.

```
-- isteresi avoidMode
if avoidMode then
    if (not obstacle_found) or (closest_dist_found > d_off) then avoidMode = false end
else
    if obstacle_found and (closest_dist_found < d_on) then avoidMode = true end
end
```

5 Simulation Results and Discussion

5.1 First scenario with no Obstacles

The first simulation scenario evaluates the nominal behavior of the person-following controller in an obstacle-free environment. This setup allows the assessment of tracking performance and control stability without interference from the avoidance logic.

In this case, the target motion is not generated through the OMPL planner. Instead, the trajectory is **explicitly imposed** in order to test the controller response under **linear** and **sinusoidal** trajectories.

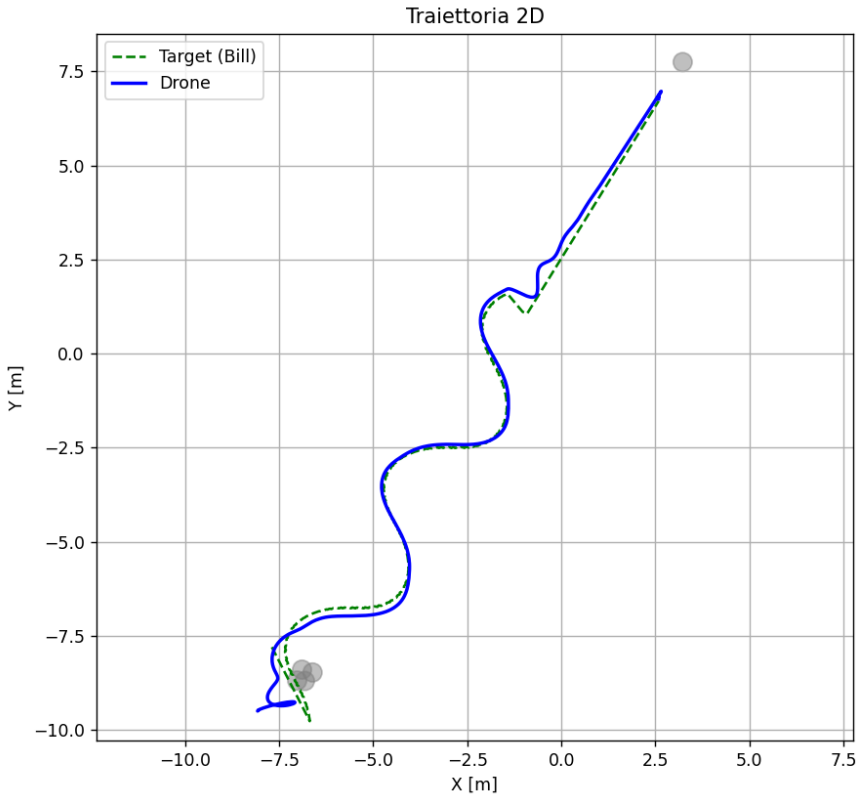
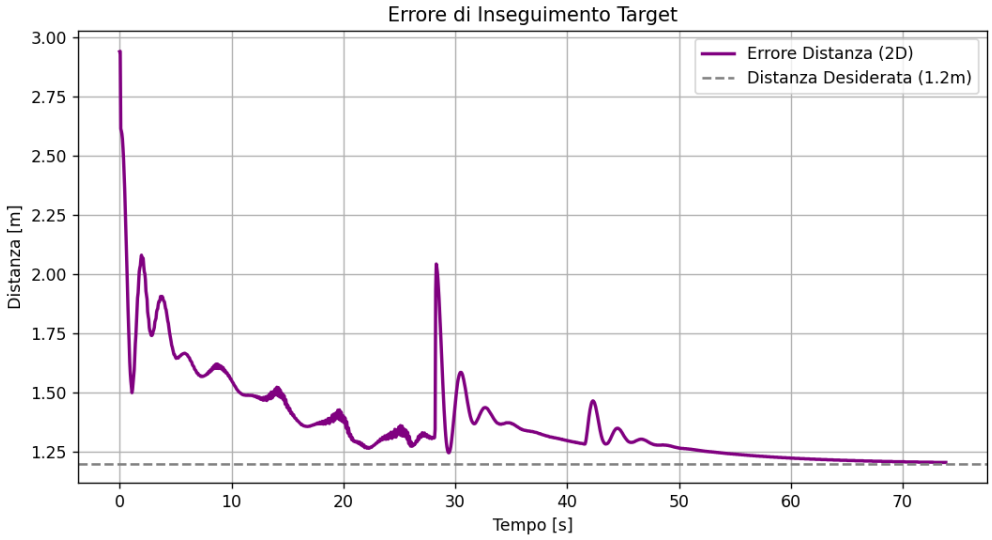


Figure **(Trajectory 2D)** shows that the quadrotor successfully follows the target along the entire planned path, maintaining the desired relative configuration. After an initial transient due to the initial distance mismatch, the drone trajectory closely overlaps the target trajectory, even during curved segments, indicating stable and accurate tracking.



The evolution of the 2D distance error is reported in Figure **(Tracking Error)**. The controller rapidly reduces the initial error and converges smoothly toward the

desired following distance of 1.2 m. Minor residual oscillations are observed, mainly caused by curvature changes in the target motion rather than instability of the control law.

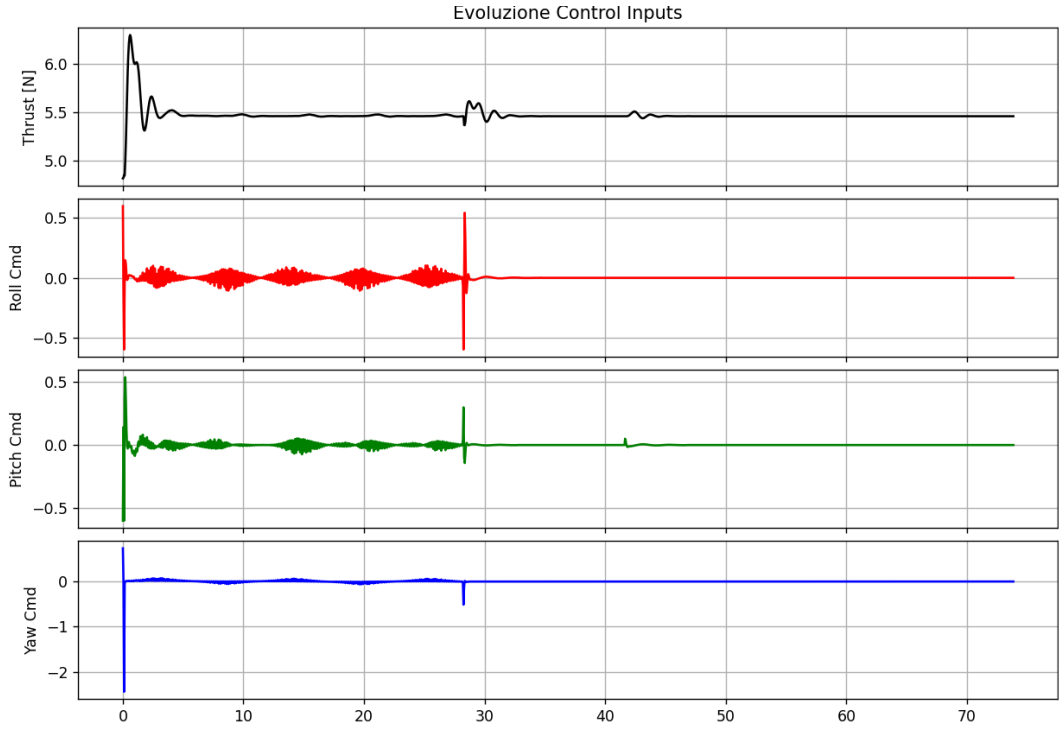


Figure **(Control Inputs Evolution)** illustrates the behavior of the control commands. The thrust exhibits a short initial transient and then stabilizes around a constant value, ensuring altitude regulation. Roll and pitch commands remain bounded and smooth, reflecting moderate lateral and longitudinal corrections required for trajectory tracking. The yaw command quickly converges to zero, confirming effective alignment with the target orientation.

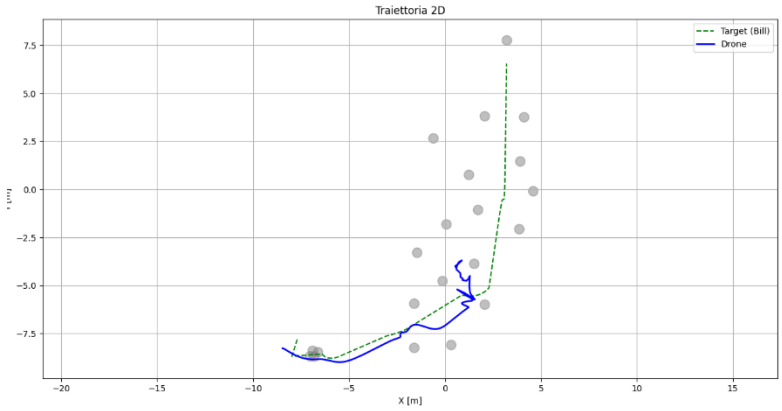
Overall, the results demonstrate that the proposed control architecture guarantees stable, smooth, and accurate person following under nominal conditions. This scenario provides a baseline reference for evaluating the impact of obstacle avoidance mechanisms in the subsequent simulations.

Block	**Parameter**	**Value**	**Meaning / Role**
Altitude Control (PID-V)	Proportional gain	10.0	Altitude error feedback
	Integral gain	0.2	Removes steady-state altitude error
	Velocity feedback gain	−1.5	Vertical damping via vertical velocity

Block	Parameter	Value	Meaning / Role
	Derivative gain	0.4	Defined but not applied (commented in thrust law)
Following Strategy	Desired distance	1.2 m	Offset behind the target
	Height offset	1.5 m	Desired altitude relative to target
	Lateral offset	0.0 m	No side-offset
Obstacle Avoidance Field	Repulsive gain	0.8	Strength of repulsive action (shaped)
	Vortex gain	0.4	Tangential component strength (shaped)
	Drone radius	0.35 m	Clearance adjustment in distance computation
	Distance shaping lower bound	0.25 m	Minimum effective distance used for shaping strength
Stabilization / Heuristics	Force smoothing factor	0.2	Low-pass filtering on repulsive/vortex components
	Force saturation (global)	1.0	Clamp on avoidance force components (Fx, Fy)
	Damping gain	0.7	Adds velocity-projection damping along obstacle direction
	Tracking attenuation factor	0.5	Tracking setpoint is reduced as avoidance weight increases (sp *= (1-0.5*w))
Horizontal Control (attitude mapping)	Attitude error derivative gain	2.5	Roll/pitch damping from attitude-related error terms
	Position gain (body-frame)	0.01	Mapping of lateral/longitudinal error into tilt command
	Position derivative gain	2.0	Damping on setpoint variation (sp - psp)
	Max tilt	0.6	Saturation limit on roll/pitch

Block	Parameter	Value	Meaning / Role
		rad	commands
Yaw Control (PD)	Proportional gain	0.01	Heading alignment
	Derivative gain	1.0	Yaw rate damping

5.2 Scenario 2 – Obstacles, *No Heuristic* (Unstable behavior)



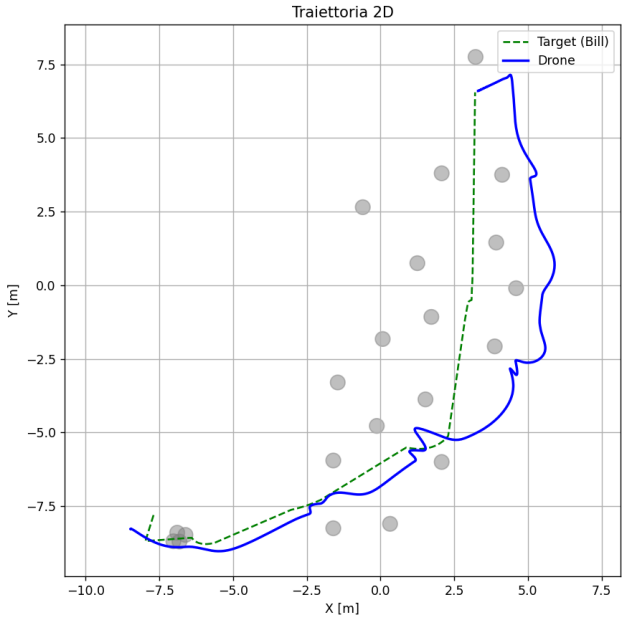
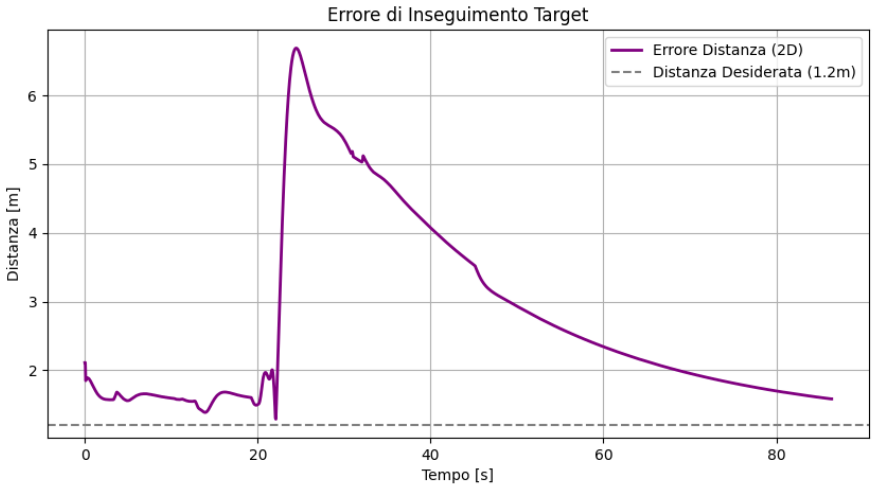


Figure **(2D Trajectories)** reports two different executions of the same scenario, performed with identical control parameters and obstacle configuration. The

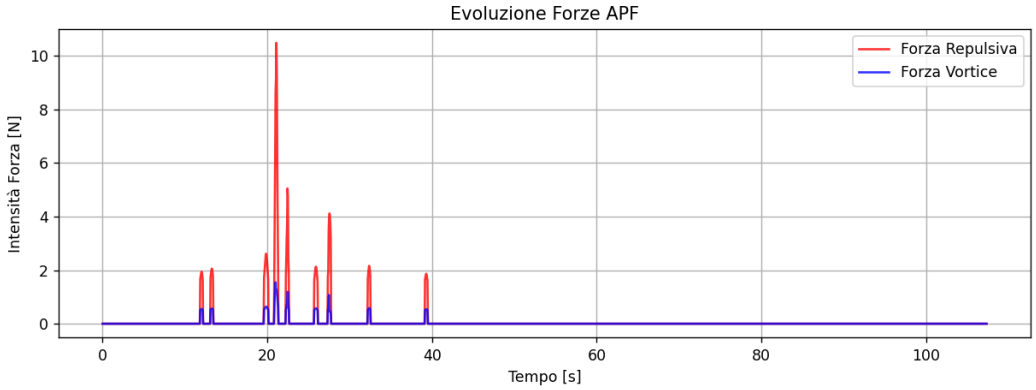
presence of two significantly different drone trajectories highlights a **lack of robustness** in the closed-loop behavior.

In both runs, the target follows the same planned path, while the UAV initially tracks it correctly. However, when the drone enters the obstacle-influenced region, its motion deviates abruptly from the nominal following configuration.
In the first trajectory, the UAV exhibits oscillatory behavior close to the obstacle cluster, resulting in delayed convergence and degraded tracking.
In the second trajectory, the deviation is more severe: the drone performs a large lateral excursion and fails to re-align with the target path within the simulation time.

The fact that **small variations in interaction timing or sensed obstacle geometry lead to qualitatively different trajectories** is a clear indication of instability and sensitivity to perturbations. This behavior is not attributable to the target motion, which remains unchanged, but rather to the interaction between obstacle-induced forces and the tracking controller.



The instability observed at the trajectory level is reflected in the evolution of the error position between bill and the drone.



The **repulsive component** of the artificial potential field exhibits sharp peaks, dominating the total corrective action. The **vortex component**, although present, is significantly smaller and insufficient to smoothly redirect the motion around obstacles. As a result, the avoidance action acts mainly as an impulsive lateral disturbance rather than as a guidance mechanism.

This impulsive behavior propagates directly to the control inputs. Roll and pitch commands show sudden spikes and reach the imposed saturation limits, indicating that the controller is attempting aggressive attitude corrections to counteract the obstacle-induced forces. These abrupt corrections disrupt the tracking objective and lead to a rapid growth of the distance error.

Once the deviation occurs, the controller is unable to fully recover the desired relative configuration, and the distance from the target remains significantly larger than the reference value for the remainder of the simulation.

This scenario demonstrates that, in the absence of additional stabilization mechanisms, a purely reactive APF–vortex formulation may destabilize the person-following task. The closed-loop system becomes highly sensitive to obstacle interaction, producing non-repeatable trajectories and loss of tracking accuracy.

The comparison between the two executions emphasizes that the issue is not a simple transient degradation but a structural limitation of the control strategy in this configuration. This motivates the introduction of smoothing, blending, and hysteresis mechanisms in the following scenarios, aimed at ensuring bounded control actions and consistent behavior across repeated runs.

Control Block	Parameter	Value	Description
Altitude Control (PID-	Proportional gain	10.0	Vertical position regulation

Control Block	Parameter	Value	Description
V)			
	Integral gain	0.2	Steady-state altitude correction
	Velocity feedback gain	−1.5	Vertical damping via velocity
Following Strategy	Desired distance	1.2 m	Longitudinal offset behind target
	Height offset	1.5 m	Constant altitude difference
Obstacle Avoidance (APF)	Repulsive gain	0.4	Intensity of obstacle repulsion
	Vortex gain	0.25	Tangential redirection component
Horizontal Control	Position gain	0.001	Mapping of body-frame error
	Derivative gain	0.5	Damping on setpoint variation
	Attitude derivative gain	2.5	Roll/pitch rate damping
	Maximum tilt	0.6 rad	Saturation limit on roll/pitch
Yaw Control (PD)	Proportional gain	0.1	Heading alignment
	Derivative gain	2.0	Yaw rate damping

5.3 Scenario 3 – Obstacles with Heuristic Stabilization (Stable behavior)

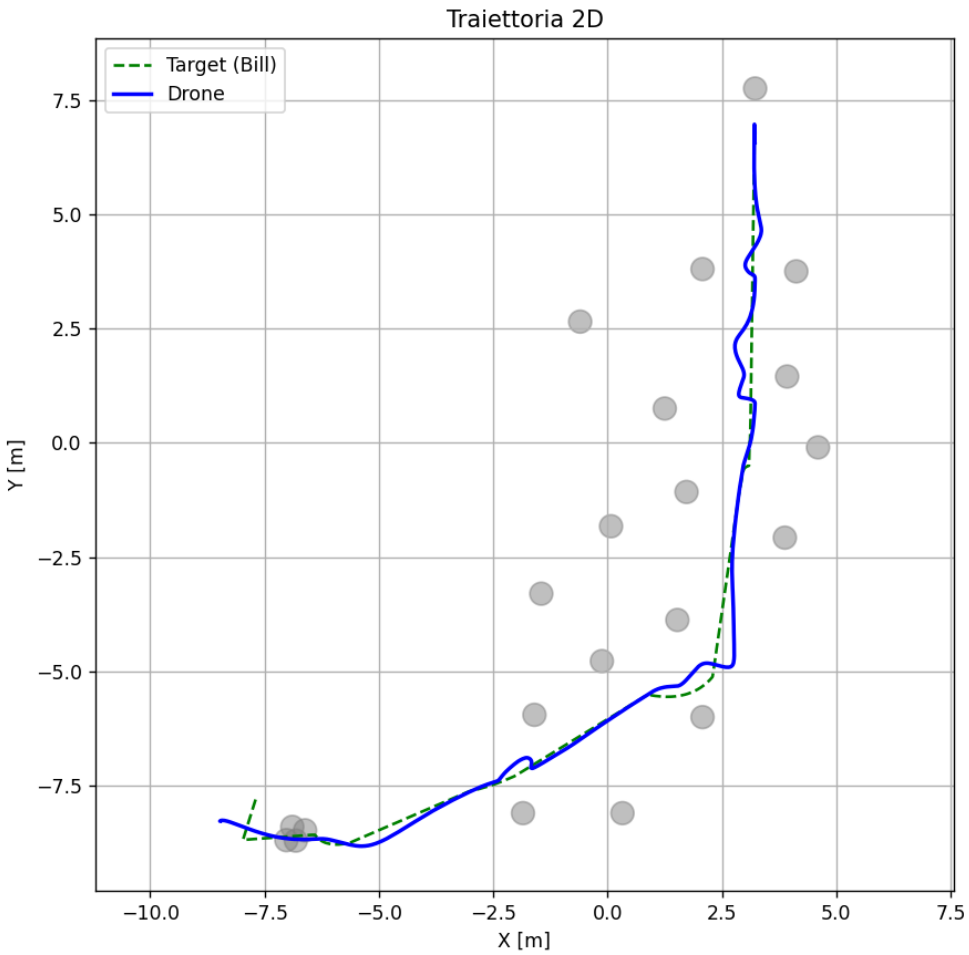
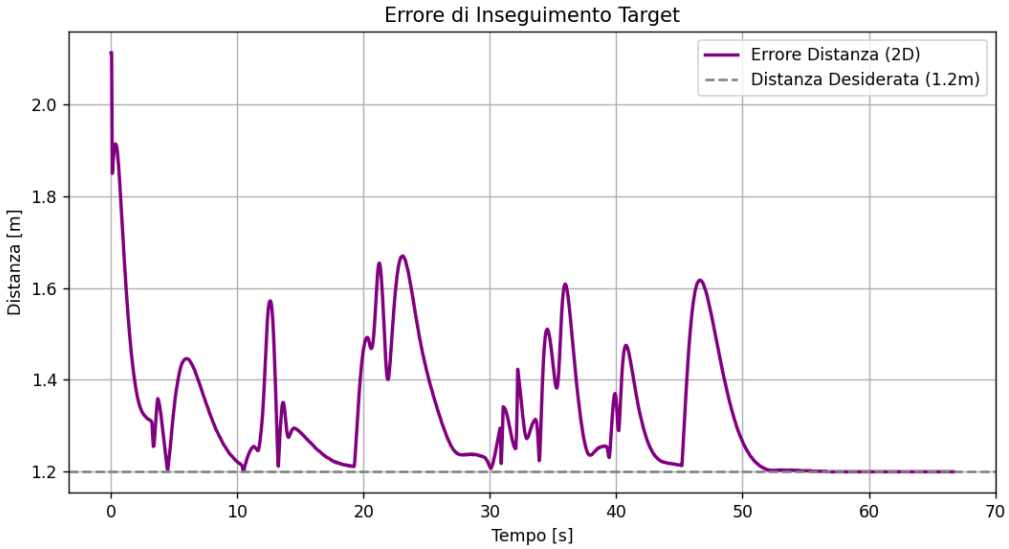
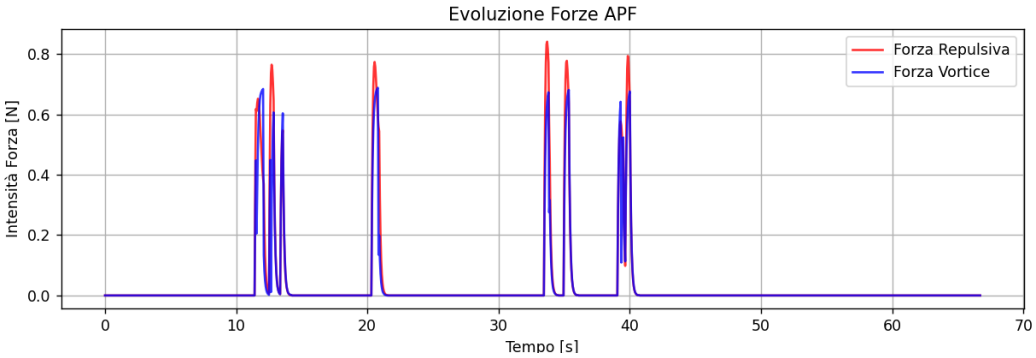


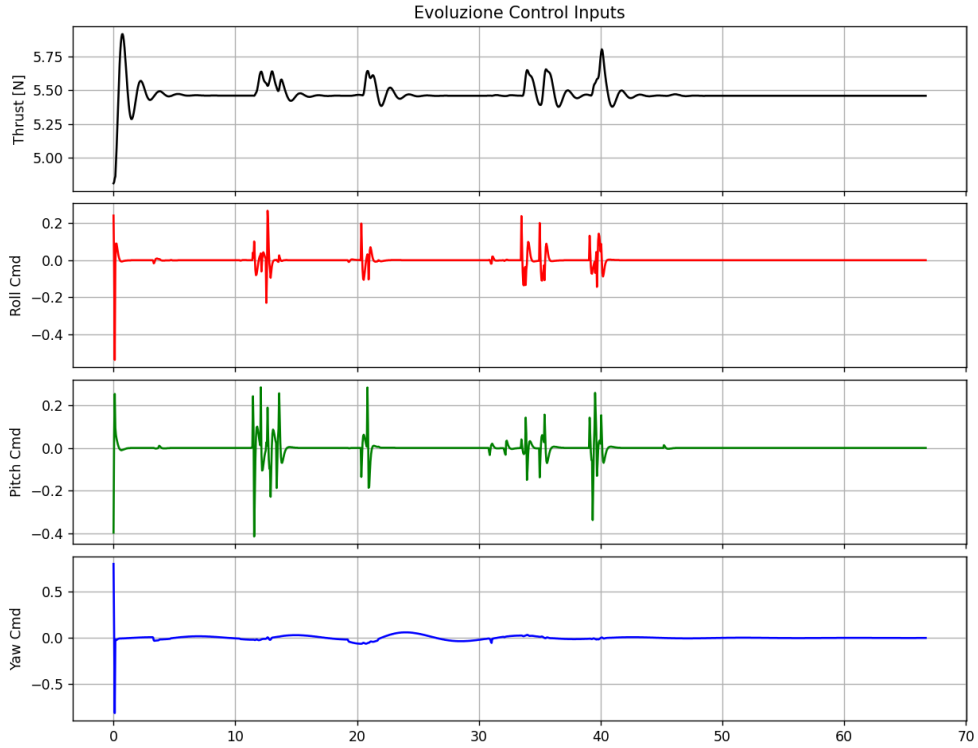
Figure **(2D Trajectory)** shows that the UAV is able to follow the target path while passing through an environment populated with obstacles. Differently from the unstable case, the drone trajectory remains **bounded and repeatable**, and the deviation from the target path is limited to a narrow corridor around the obstacle cluster. The UAV briefly "slides" laterally to bypass obstacles and then naturally re-aligns behind the target, indicating that obstacle interaction does not break the tracking task.



The **distance error** plot confirms this behavior: the UAV quickly converges from the initial condition mismatch and then keeps the target distance close to the desired 1.2 m. During obstacle interactions, temporary peaks appear (up to roughly 1.6 m), but the system always recovers and returns to the reference distance. Importantly, the error does not diverge and does not enter the large excursions seen in the unstable configuration.



From the **APF force plot**, repulsive and vortex components remain **bounded and comparable in magnitude**, producing short corrective actions rather than impulsive disturbances. This results in smoother avoidance maneuvers and avoids “runaway” lateral pushes.



Consistently, the **control inputs** remain within reasonable ranges: roll and pitch show short bursts during avoidance, but without sustained saturation. Thrust stays close to its nominal value, with only small oscillations due to the coupling with attitude corrections. Overall, the control commands indicate a stable closed-loop response where avoidance acts as a bounded perturbation rather than dominating the tracking objective.

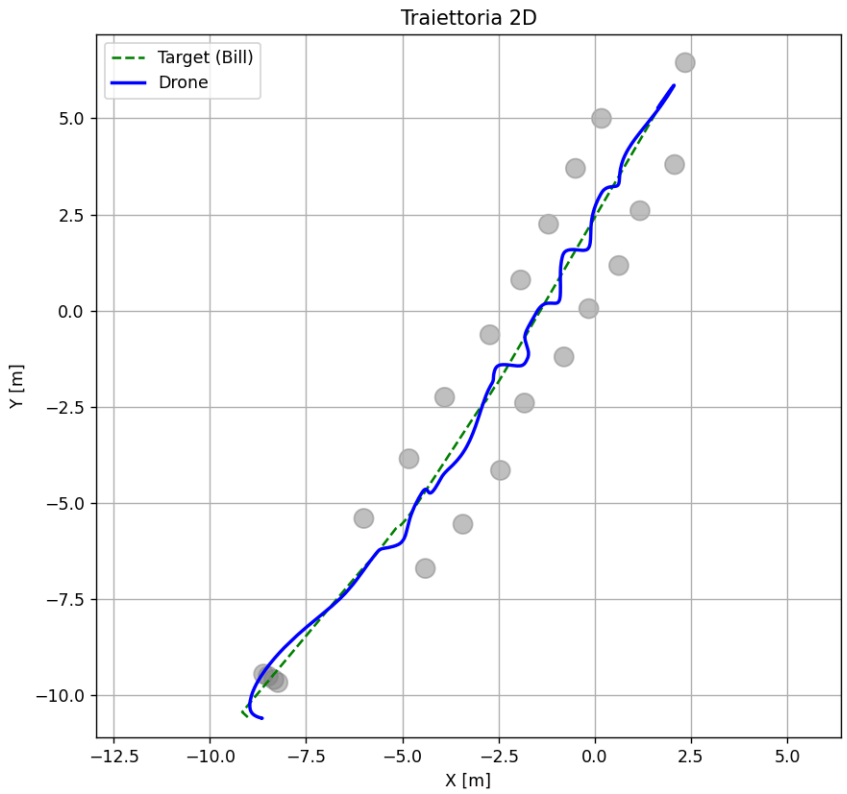
This scenario demonstrates that the heuristic additions (force shaping, smoothing, and blending between tracking and avoidance) are effective in stabilizing the person-following controller in cluttered environments. The resulting behavior is robust: obstacle interactions cause only local deviations and short transients, while the system consistently returns to the desired following configuration.

Control Parameters Used (Scenario 3 – Obstacles + Heuristic)

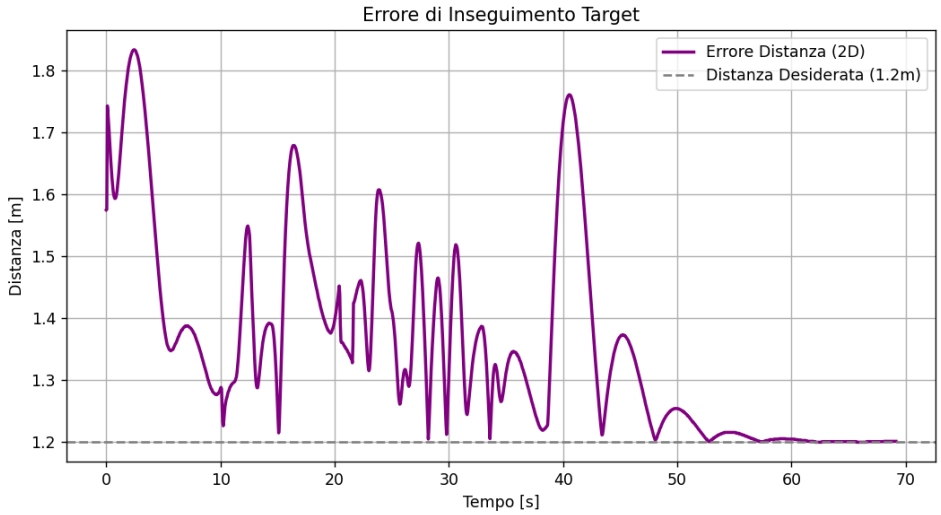
Block	Parameter	Value	Meaning / Role
Altitude Control (PID-V)	Proportional gain	10.0	Altitude error feedback
	Integral gain	0.2	Removes steady-state altitude error
	Velocity feedback	−1.5	Vertical damping via vertical

Block	Parameter	Value	Meaning / Role
	gain		velocity
	Derivative gain	0.4	Defined but not applied (commented in thrust law)
Following Strategy	Desired distance	1.2 m	Offset behind the target
	Height offset	1.5 m	Constant altitude difference
Obstacle Avoidance Field	Repulsive gain	0.7	Repulsive component intensity (shaped)
	Vortex gain	0.7	Tangential redirection intensity (shaped)
	Max avoidance force	1.0	Global clamp on avoidance force components (Fx, Fy)
	Smoothing factor	0.4	Low-pass filtering on repulsive/vortex components
	Damping gain	0.7	Velocity-projection damping along obstacle direction
Horizontal Control	Attitude derivative gain	1.5	Roll/pitch damping from attitude-related error terms
	Position gain	0.01	Mapping of body-frame position error into tilt
	Position derivative gain	0.5	Damping on setpoint variation (sp - psp)
	Max tilt	0.6 rad	Saturation limit on roll/pitch
Yaw Control (PD)	Proportional gain	0.1	Heading alignment
	Derivative gain	1.0	Yaw rate damping

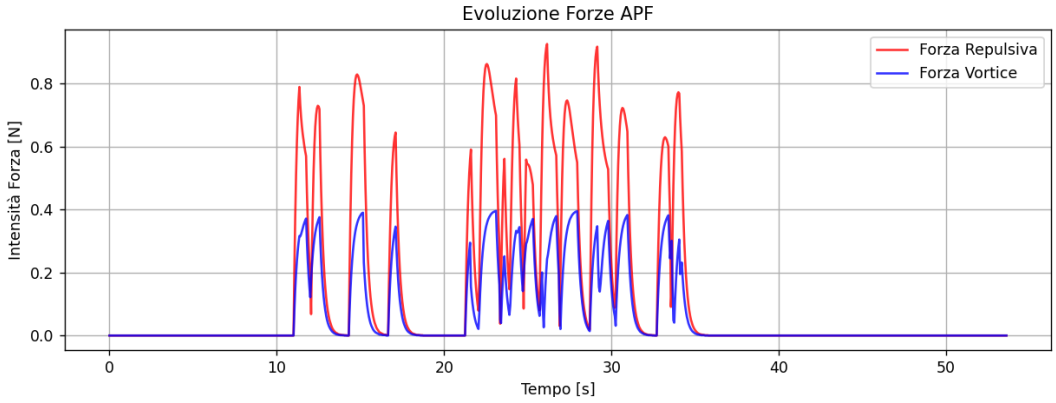
5.4 Scenario 4 – Corridor Scenario (Obstacle-Dense Passage)



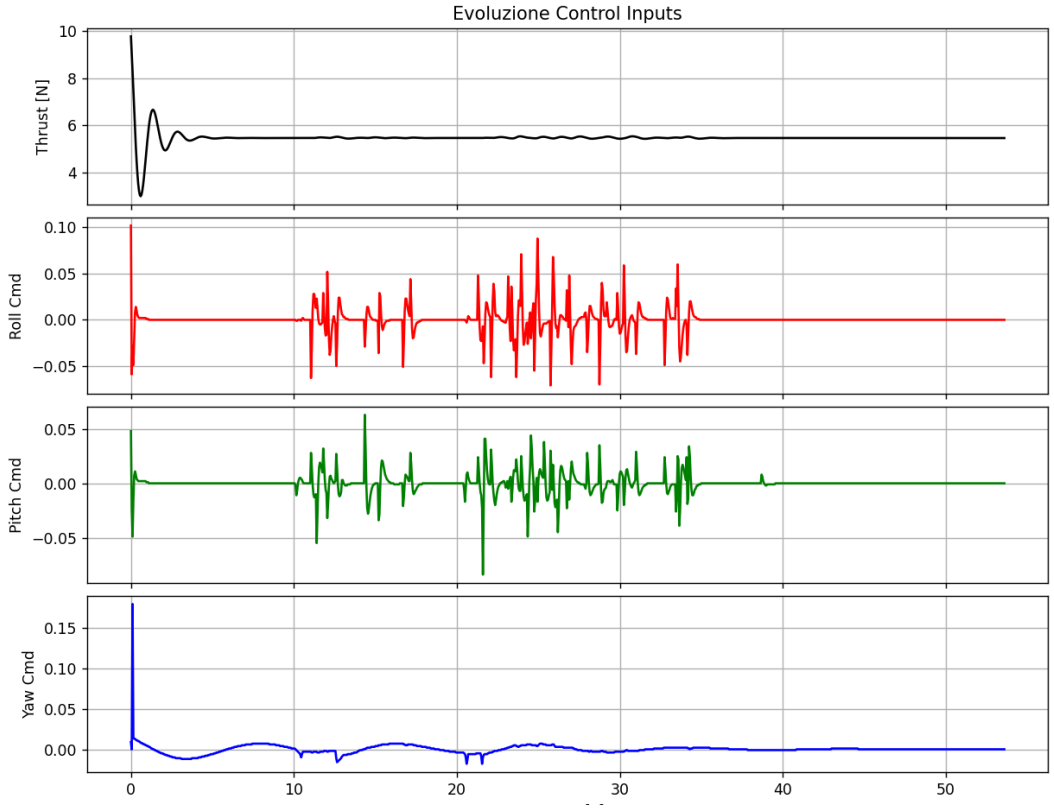
From the **2D trajectory** plot, the UAV follows the target along the corridor-shaped obstacle configuration while keeping a path that stays **inside the free space** and avoids collisions. Compared to Scenario 3, the motion appears **more constrained**: the drone cannot "go wide" around obstacles, therefore it performs small lateral corrections while remaining close to the target's route. The trajectory remains smooth and does not show divergence or large loops, indicating a stable closed-loop behavior even in a narrow passage.



The **distance-to-target error** confirms that the corridor generates a stronger tracking/avoidance interaction. The UAV starts with an initial transient and then oscillates around the reference distance during the corridor traversal, producing several peaks (up to ~1.8 m). However, the error remains bounded and progressively settles: once the corridor section is cleared, the distance converges back close to **1.2 m**, showing recovery and no long-term drift.



The **APF force evolution** shows frequent activations in the corridor segment (dense obstacle interactions). The repulsive component dominates, while the vortex contribution remains lower but persistent, supporting lateral redirection instead of pure "push-back". Importantly, forces remain bounded (no runaway growth).



Consistently, the **control inputs** remain small and well-behaved: roll and pitch commands stay within a tight range (order of ±0.05–0.1 in the plot), meaning the drone reacts with **micro-corrections** rather than aggressive tilts. Thrust stabilizes quickly

around its nominal value after the initial transient, indicating that the corridor interaction mainly affects lateral control rather than altitude regulation.

This scenario stresses the controller in a constrained geometry: obstacle avoidance must be continuously active while still allowing tracking. The results show a stable compromise: tracking quality temporarily degrades (bounded peaks in distance error), but the UAV maintains safe motion through the corridor and restores the desired following distance once free space increases.

Block	Parameter	Value	Meaning / Role
Altitude Control (PID-V)	Proportional gain	10.0	Altitude error feedback
	Integral gain	0.2	Removes steady-state altitude error
	Velocity feedback gain	−1.5	Vertical damping via vertical velocity
	Derivative gain	0.4	Defined but not applied (commented in thrust law)
Following Strategy	Desired distance	1.2 m	Offset behind the target
	Height offset	1.5 m	Constant altitude difference
	Lateral offset	0.0 m	No side-offset
Obstacle Avoidance Field	Repulsive gain	0.8	Repulsive component intensity (shaped)
	Vortex gain	0.4	Tangential redirection intensity (shaped)
	Max avoidance force	1.0	Global clamp on avoidance force components (Fx, Fy)
	Smoothing factor	0.2	Low-pass filtering on repulsive/vortex components
	Damping gain	0.7	Velocity-projection damping along obstacle direction
Horizontal Control	Attitude derivative gain	2.0	Roll/pitch damping from attitude-related error terms
	Position gain	0.01	Mapping of body-frame position error into tilt

Block	Parameter	Value	Meaning / Role
	Position derivative gain	0.3	Damping on setpoint variation (sp - psp)
	Max tilt	0.6 rad	Saturation limit on roll/pitch
Yaw Control (PD)	Proportional gain	0.1	Heading alignment
	Derivative gain	1.0	Yaw rate damping

6 Conclusions and Future Works

6.1 Conclusions

The primary objective of this project was to design and validate a control architecture for a UAV capable of performing person-following tasks while autonomously negotiating cluttered environments. The simulation results confirm that the proposed solution, based on a reactive Vortex Potential Field, successfully meets these requirements, overcoming the inherent limitations of traditional repulsive methods.

The core contribution of this work lies in the robustness of the avoidance strategy. Unlike standard potential fields that often lead to local minima with consequent stop or even instability of the robot in front of obstacles, the implemented vortex logic generates tangential forces that allow the UAV to slide smoothly around impediments. This behavior ensures continuous forward motion and allows the drone to maintain the target in sight even during avoidance maneuvers.

Furthermore, the integration of signal conditioning techniques proved essential for flight stability. The continuous blending mechanism w effectively resolved the conflict between the tracking and avoidance objectives, eliminating the jerky control responses typically observed in switching-based architectures. Similarly, the use of hysteresis thresholds and force saturation ensured that the system remained robust against sensor noise and boundary fluctuations, preventing chattering effects near the safety limits. In conclusion, the project proves that a hierarchical controller augmented with vortex-based logic offers a computationally efficient and reliable solution for real-time UAV assistance.

6.2 Future Works

To further enhance the applicability of the system in real-world scenarios, several improvements can be adopted for future development:

- **Navigation in Narrow Corridors:** A critical limitation of potential field methods is the traversal of narrow spaces, as corridors or doorways. In such scenarios, repulsive forces from opposite walls may cancel each other out or constrain the drone in an oscillatory equilibrium. Future iterations could implement an adaptive influence radius d_0 or a specific "corridor mode" that aligns the repulsive field with the axis of the passage to facilitate safe traversal

- **State Estimation and Sensor Noise:** The current simulation assumes ideal, noise-free sensing. Future work should integrate a state estimator, such as an EKF (Extended Kalman Filter), to handle measurement uncertainty and sensor delays, which are unavoidable in physical hardware

- **Dynamic Obstacle Avoidance:** While the current system effectively handles static environments, extending the vortex logic to account for the velocity vector of moving obstacles would significantly improve safety in dynamic crowds

- **Vision-Based Target Tracking:** Replacing the ground-truth position data provided by the simulator with an onboard computer vision module (e.g., using YOLO or similar algorithms) would render the system fully autonomous and deployable without external motion capture systems