Safe and Minimum Energy Trajectory Planning using Flatness

Summary

Introduction

The use of Unmanned Aerial Vehicles (UAVs) is widely required in many works and scenarios that are too dangerous to be carried out by person or simply unrealizable without devices able to vertical take-off and landing, slow flight and stationarity : some applications are maintenance and inspection of structures, search and rescue operations or aerial transportations.
For these reasons, the UAVs must be able to navigate in environment full of obstacles, generate feasible paths and trajectories to use for hovering in the environment without collision and doing all of this with limited resources (e.g. battery power).

This project is focused on the realization of a safe and minimum energy trajectory planning for quadrotor unmanned aerial vehicles.
Generally, in a trajectory planning problem, the main focus is in the implementation of a system able to provide the minimum time trajectory. In this work, instead, we determine the minimum energy path between a predefined initial and final configuration of the UAV in order to overcome one of the main issues of these devices.

It is known that the major limitation of existing battery-powered quadrotor UAVs is their reduced flight endurance. This problem is widely known, and it was faced by proposing different kinds of solutions in the literature. For example, to alleviate the problem of the energy consumption of the UAVs, in state-of-the-art, a significant effort has been invested in weight reduction, and in the improvement of power-to-height ratio of brushless DC motor.

This project is not focused on mechanical solutions, but differently we propose an algorithm level approach: using several techniques we elaborated a procedure that allows to save energy and extend endurance. It represents a novel approach since in the literature the problem of generating energy-optimal paths for a rotorcraft has received much less attention.

This report will show the different phases of our work: we will present all the steps of the protocol that we realized to best satisfy our goals. We are going to explain the different meanings that we assigned to the concept of safety: we will split this concept into *internal* and *external safety* in order to best satisfy the tasks of our problems using different algorithms. In fact, the external safety concept represents the generation of a free-collision path and it can be achieved by the application of the RRT algorithm; whereas, the internal safety concept can be interpreted as a feature of the optimization problem used to plan the trajectory: here, we add some non-linear constraints in order to accomplish the mission without hitting the actuator constraints.
The theoretical background behind the control approach that we adopted will be described in order to justify and comment our approach. After this introduction we will focus on our implementations to solve the safe and minimum trajectory planning problem satisfying all the assigned conditions.

The final part of the report will describe all the simulation results that we obtained by introducing different scenarios to test the efficiency of our method: we implemented three main environments characterized by different set of obstacles, and we will show the main features of our approach and how it is able to always plan a safe and minimum energy trajectory, in spite of the topography of the environment.

1. Problem Description

The main goal of this work is to find a solution to one of the major limitations that characterizes the Unmanned Aerial Vehicles, namely the reduced flight endurance. In fact, this is typically between 15 and 30 minutes, depending on the model, and different applications were proposed in the literature to alleviate this problem like weight reduction and the improvement of power-to-weight ratio of the brushless DC motors.
In this work, this problem linked to the limited power of the UAVs' batteries is faced proposing a novel algorithm-level solution: a flatness based trajectory planning strategy is proposed for a quadrotor unmanned aerial vehicle. A feasible trajectory is a trajectory that lies inside the admissible state domain and that does not violate the input constraints. If system constraints are not considered, the trajectory may be unfeasible, and the defined mission may not be accomplished.
The purpose of this work is to implement a **safe** and **minimum energy** trajectory since the energy-optimal paths for a rotorcraft has received much less attention in the aerial robotics literature. The safety concept was developed under two perspectives: the first is about the internal mechanisms of the UAV, in fact the input bounds are realistically interpreted as the actuators limits, because when they are hit the actuators cannot deliver the actuation inputs desired by the controller. The second point of view is about the environment where the UAV has to move: if we consider an empty and unbounded environment, the safety is always accomplished, but if we consider boundaries and obstacles in the environment, then a method to generate a path that does not provide collisions is needed.

Indeed, we can define three main objectives for our trajectory planning problem: internal safety, external safety and minimum energy. For these three tasks, we adopted different solutions and then we integrate them in a unique final system able to satisfy all of them. In the next sections, we will show all the steps that lead to the final solution.

1.2 System Model

For this work we employ a commonly quadrotor UAV model with four motors located at the extremities of a cross-shape structure . We start from a simplified deterministic model of the more complex and uncertain real system:

$$\begin{aligned}
&\ddot{x} = u_{1_x} - \frac{k_1}{m}\dot{x}; \qquad \ddot{\theta} = u_2 - \frac{lk_4}{J_1}\dot{\theta} \\
&\ddot{y} = u_{1_y} - \frac{k_2}{m}\dot{y}; \qquad \ddot{\phi} = u_3 - \frac{lk_5}{J_2}\dot{\phi} \\
&\ddot{z} = \ -g + u_1(cos\phi cos\theta) - \frac{k_3}{m}\dot{z} \\
&\ddot{\psi} = u_4 - \frac{lk_6}{J_3}\dot{\psi}
\end{aligned}$$

Where:

$$\begin{aligned}
\boldsymbol{u_{1x}} &= u_1(cos(\phi)\, sin(\theta)\, cos(\psi) + sin(\phi)\, sin(\varphi)); \\
\boldsymbol{u_{1y}} &= u_1(cos(\phi)\, sin(\theta)\, sin(\psi) - sin(\phi)\, cos(\psi)).
\end{aligned}$$

The ***x***, ***y*** and ***z*** are the coordinates of the quadrotor centre of gravity in the Earth-frame. $\boldsymbol{\theta}, \boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are the pitch, roll and yaw angles correspondent to the 3-2-1 rotation. m is the mass and $\boldsymbol{J_i}$ (i =1,2,3) are the moments of inertia along x, y and z directions**.**
$\boldsymbol{k_i}$ (i = 1….6) are the drag coefficients and ***l*** is the distance from the centre of gravity to each rotor.
$\boldsymbol{u_1}$ is the linear acceleration applied to the quadrotor in the z-direction of the body frame.
$\boldsymbol{u_2}, \boldsymbol{u_3}$ and $\boldsymbol{u_4}$ are, respectively, the angular accelerations applied in θ, ϕ and ψ directions.

Even though this is a simplified version of the UAV model, it is still too complicated to deal with. From here, we can start different simplifications that does not affect the robustness of the controller that will be implemented. First by assuming small angles θ and ϕ and a constant yaw angle $\psi = 0$, we can rewrite:

$$\begin{aligned}
&\ddot{x} = u_1\theta - \frac{k_1}{m}\dot{x}; \qquad \ddot{\theta} = u_2 - \frac{lk_4}{J_1}\dot{\theta} \\
&\ddot{y} = u_1\phi - \frac{k_2}{m}\dot{y}; \qquad \ddot{\phi} = u_3 - \frac{lk_5}{J_2}\dot{\phi} \\
&\ddot{z} = \ -g + u_1 - \frac{k_3}{m}\dot{z} \\
&\ddot{\psi} = u_4 - \frac{lk_6}{J_3}\dot{\psi}
\end{aligned}$$

If we consider negligible drag coefficients at low speeds:

$$\begin{aligned}
&\ddot{x} = u_1\theta; \qquad \ddot{\theta} = u_2 \\
&\ddot{y} = u_1\phi; \qquad \ddot{\phi} = u_3 \\
&\ddot{z} = \ -g + u_1 \\
&\ddot{\psi} = u_4
\end{aligned}$$

Finally, decoupling the z and x-y axes, one can assume that in hovering condition, $u_1 \approx g$ in the x and y directions, such that:

$$\begin{aligned} &\ddot{x} = g\theta; \qquad \ddot{\theta} = u_2 \\ &\ddot{y} = -g\phi; \quad \ddot{\phi} = u_3 \\ &\ddot{z} = -g + u_1 \\ &\ddot{\psi} = u_4 \end{aligned}$$

The flatness-based trajectory planning problem is based on determining the maximal thrusts to be applied for a given trajectory. Since it is not easy using the first proposed model due to calculation load issues, these simplifications allowed to define a model that can be used for this control purposes.

Before of proceeding with the presentation of the control strategy, we consider the relation between the accelerations and the rotor thrusts:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \\ \frac{-l}{J_1} & \frac{-l}{J_1} & \frac{l}{J_1} & \frac{l}{J_1} \\ \frac{-l}{J_2} & \frac{l}{J_2} & \frac{l}{J_2} & \frac{-l}{J_2} \\ \frac{C}{J_3} & \frac{-C}{J_3} & \frac{C}{J_3} & \frac{-C}{J_3} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}$$

Where $T_i(i = 1,..,4)$ is the thrusts generated by the i-th rotor and C is the Thrust-to-moment scaling factor.
The vectors in this matrix must be normalized in order to make the problem more understandable and easier to solve. We divide the vector components by their maximum values, such that each component is a dimensionless number that lies between 0 and 1. Assuming $T_{1Max} = T_{2Max} = T_{3Max} = T_{4Max} = T_{max}$, then we get the maximum values of the inputs u_i:

$$u_{1_{max}} = \frac{4}{m} T_{max}; \quad u_{2_{max}} = \frac{2l}{J_1} T_{max}; \quad u_{3_{max}} = \frac{2l}{J_2} T_{max}; \quad u_{4_{max}} = \frac{2C}{J_3} T_{max}.$$

The normalized relation is:

$$\begin{pmatrix} \frac{u_1}{u_{1_{max}}} \\ \frac{u_2}{u_{2_{max}}} \\ \frac{u_3}{u_{3_{max}}} \\ \frac{u_4}{u_{4_{max}}} \end{pmatrix} = \begin{pmatrix} +\frac{1}{4} & +\frac{1}{4} & +\frac{1}{4} & +\frac{1}{4} \\ -\frac{1}{2} & -\frac{1}{2} & +\frac{1}{2} & +\frac{1}{2} \\ -\frac{1}{2} & +\frac{1}{2} & +\frac{1}{2} & -\frac{1}{2} \\ +\frac{1}{2} & -\frac{1}{2} & +\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{T_1}{T_{max}} \\ \frac{T_2}{T_{max}} \\ \frac{T_3}{T_{max}} \\ \frac{T_4}{T_{max}} \end{pmatrix}$$

The normalization process allowed to obtain the advantage to remove all the physical parameters from the relation.

1.3 Flatness-Based Control

The basic idea of flatness based trajectory planning is to parametrize the control inputs to be applied in function of the reference trajectory to be followed. For our control problem, we need to find outputs that can be controlled and chosen as flat outputs of the system. In fact, the flatness property implies that a dynamical system,

$$\dot{x} = f(x,u); \quad y = h(x)$$

with $x \in R^n$ and $u \in R^m$, is called flat if and only if there exist variables $F \in R^m$ called the flat outputs such that $x = \Xi_1(F, \dot{F}, \ldots, F^{(n-1)})$, $y = \Xi_2(F, \dot{F}, \ldots, F^{(n-1)})$ $and\ u = \Xi_3(F, \dot{F}, \ldots, F^{(n)})$, where Ξ_1, Ξ_2, Ξ_3 are three smooth mappings and $F^{(i)}$ is the i-th derivative of F.

This control approach is crucial for the realization of the system: in fact, the parametrization of the control inputs u in function of the flat outputs F plays a key role in the trajectory planning problem: the nominal control inputs to be applied during the mission can be expressed in function of the desired trajectories.

In our case, the flat outputs are chosen as:

$$F_1 = z\,;\ F_2 = x;\ F_3 = y;\ F_4 = \psi\ (= 0\ as\ assumed\ in\ the\ model\ simplification)$$

From these assumptions, we get the parametrization of the control inputs and $\theta\ and\ \phi$ in function of the flat outputs:

$$\theta = \frac{cosF_4\left(\ddot{F}_2 + \frac{k_1}{m}\dot{F}_2\right) + sinF_4\left(\ddot{F}_3 + \frac{k_2}{m}\dot{F}_3\right)}{\ddot{F}_1 + \frac{k_3}{m}\dot{F}_1 + g}$$

$$\phi = \frac{sinF_4\left(\ddot{F}_2 + \frac{k_1}{m}\dot{F}_2\right) - cosF_4\left(\ddot{F}_3 + \frac{k_2}{m}\dot{F}_3\right)}{\ddot{F}_1 + \frac{k_3}{m}\dot{F}_1 + g}$$

To complete the flatness-based control procedure it just remains to define the reference trajectories F_i^*. We chose for our work to use Bézier Polynomials:

where t is the time and a_i are constant coefficients.

In order to obtain smooth control inputs, we decided to employ Bézier polynomial function of degree 5 for F_1 and F_4 and degree 9 for F_2 and F_3, such that the reference trajectories are:

$$F_i^* = a_5^i t^5 + a_4^i t^4 + a_3^i t^3 + a_2^i t^2 + a_1^i t + a_0 \quad (i = 1{,}4)$$

$$F_i^* = a_9^i t^9 + a_8^i t^8 + a_7^i t^7 + a_6^i t^6 + a_5^i t^5 + a_4^i t^4 + a_3^i t^3 + a_2^i t^2 + a_1^i t + a_0 \quad (i = 2{,}3)$$

The coefficients of the polynomials can be easily calculated in function of the initial and final conditions. In particular, assuming t_0 *and* t_f as respectively initial and final instants of the mission, we have that a_j^i:

- For $i = 1,4 \ and \ j = 0, \dots, 5$:
  - Initial conditions: $F_i(t_0), \dot{F}_i(t_0), \ddot{F}(t_0)$
  - Final conditions: $F_i(t_f), \dot{F}_i(t_f), \ddot{F}(t_f)$

- For $i = 2,3 \ and \ j = 0, \dots, 9$:
  - Initial conditions: $F_i(t_0), \dot{F}_i(t_0), \ddot{F}(t_0), F_i^{(3)}(t_0), F_i^{(4)}(t_0)$
  - Final conditions: $F_i(t_f), \dot{F}_i(t_f), \ddot{F}(t_f), F_i^{(3)}(t_f), F_i^{(4)}(t_f)$

The choice of this control approach is used specially to have the chance to tune the parameters of the desired trajectory so that the actuator constraints are not violated. Moreover, we chose to use the Bézier Polynomials because they allow to calculate in a straightforward manner the extrema of the control inputs to be applied along the trajectory. These trajectory planning features that we are mentioning in this paragraph will be clearer in the next sections, in particular when we have to deal with the implementation of what we called “internal safety”.

2. Minimal-Energy Trajectory Planning

As mentioned in the previous sections, the three main constraints of this trajectory planning problem are the so called internal safety, external safety and minimum-energy. In the following paragraphs, we will see that the accomplishment of the third element is intrinsic in the general control approach. In fact, the starting point of our implementation is to define a system able to plan a trajectory using the minimum energy effort and it is achieved by defining an optimization problem.

Before of discussing about the definition of this control problem, we have to denote the nominal thrusts $\underline{T_i^*} = T_i/T_{max}$ to be applied along the reference trajectories F_i^*. they can be computed as follows:

$$\underline{T_1^*} = \frac{u_1^*}{u_{1max}} - \frac{1}{2}\frac{u_2^*}{u_{2max}} - \frac{1}{2}\frac{u_3^*}{u_{3max}} + \frac{1}{2}\frac{u_4^*}{u_{4max}}$$

$$\underline{T_2^*} = \frac{u_1^*}{u_{1max}} - \frac{1}{2}\frac{u_2^*}{u_{2max}} + \frac{1}{2}\frac{u_3^*}{u_{3max}} - \frac{1}{2}\frac{u_4^*}{u_{4max}}$$

$$\underline{T_3^*} = \frac{u_1^*}{u_{1max}} + \frac{1}{2}\frac{u_2^*}{u_{2max}} + \frac{1}{2}\frac{u_3^*}{u_{3max}} + \frac{1}{2}\frac{u_4^*}{u_{4max}}$$

$$\underline{T_4^*} = \frac{u_1^*}{u_{1max}} + \frac{1}{2}\frac{u_2^*}{u_{2max}} - \frac{1}{2}\frac{u_3^*}{u_{3max}} - \frac{1}{2}\frac{u_4^*}{u_{4max}}$$

where $u_{i_{max}}$ are constants and $u_i^*$are the nominal control inputs expressed as:

$$u_1^* = \ddot{F}_1^* + g \ ; \ u_2^* = \frac{1}{g}F_2^{(4)*} \ ; \ u_3^* = -\frac{1}{g}F_3^{(4)*} \ ; \ u_4^* = \ddot{F}_4^*.$$

Using this approach, we can express the nominal thrusts in function of the reference trajectories F_i^*. The profile of F_i^* depends on the coefficients a_j^i of the reference trajectories which are defined as polynomial functions of fixed degrees. These coefficients are defined by imposing the initial and final conditions and then they depend on t_0 and t_f. Assuming that the initial time is always known, then the only unknown parameter is t_f, and it implies that the trajectory can be obtained by determining t_f.The trajectory planning problem consists in computing the final time that allows to get a minimal energy effort to drive the system from an initial position to a final one. This can be expressed as an optimization problem where the objective function to minimize is expressed by the energy consumed during the execution of the task and the decision variable is t_f.
The energy consumed in the running time interval can be computed as follows:

$$J = E(t)|_{to}^{tf} = E(t_f) - E(t_0)$$

where $E(t)$ is the primitive:

$$E(t) = \int_{t_0}^{t_f} (4u_1^{*2} + u_2^{*2} + u_3^{*2} + u_4^{*2})\, dt$$

Then the optimization problem consists in:

$$Minimize \qquad E(t_f) - E(t_0)$$

The total energy spent during the mission can be divided in two parts: the first part serves in counteracting the gravity and keeping the system in the *xy*-plane; whereas, the second part is needed to track the desired trajectory and driving the system to the final desired position.

2.1 Minimal-Energy Trajectory Planning: Internal Safety

The definition of the optimization problem is still incomplete: in fact, the profile of F_i^* can be tuned to drive the system from an initial position to a final one without hitting the actuator constraints. This is the concept expressed by what we called "Internal Safety".

Indeed, the optimization problem can be reformulated by introducing the actuator constraints:

$$\begin{cases} Minimize & E(t_f) - E(t_0) \\ Subject\ to & 0 \le T_i^* \le 1 \quad (i = 1, \dots, 4) \end{cases}$$

The nominal thrusts are in function of time t, thus solving the problem requires using calculus of variations which may impose heavy calculations. This problem can be overcome by considering only the extrema of the nominal thrusts, such that the control problem becomes:

$$\begin{cases} Minimize & E(t_f) - E(t_0) \\ Subject\ to & 0 \le T_{i_{Ext}}^* \le 1 \quad (i = 1, \dots, 4) \\ & 0 \le T_i^*(t_0) \le 1 \\ & 0 \le T_i^*(t_f) \le 1 \end{cases}$$

where it is evident that the last two rows of the constraints indicate the initial and final conditions to be applied to the thrusts values, whereas, $T_{i_{Ext}}^*$ indicates the extrema of T_i^*. The extrema are in function of initial and final time (the only unknown is the final time. They can be written as $T_{i_{Ext}}^* = T_i^*(z_i^*)$, where z_i^* are the critical points. These latter are the solutions of the derivative of T_i^* with respect to time equalized to zero.
From this approach we can define $T_{i_{Min}}^*$ and $T_{i_{Max}}^*$ as the global minima and global maxima such that:

$$T_{i_{Min}}^* \le T_i^* \le T_{i_{Max}}^* \quad \forall t \in [t_0, t_f]$$

It demonstrates that this approach leads to a solution of the control problem as in the first formulation but providing much less computations. It is necessary to highlight that the derivative of T_i^* with respect to time equal to zero has a degree not less than five, and according to theorem of Abel and Ruffini it cannot return an algebraic solution in term of finite number additions, subtractions, multiplications, divisions and root extractions. This problem was already solved by introducing the final simplified model of the quadrotor that we have presented in section 1.1. This way, T_i^* is a polynomial equation of degree five, and then its derivative has degree four and it can be used to solve the optimization problem.
It is important to highlight the fact that this solution allows to reduce the weight of calculations such that the planning can be also realized on real time on board implementation. In fact, it is sufficient to plug the initial and final conditions to get the solution t_f.

2.2 Minimal-Energy Trajectory Planning: External Safety

The optimization control problem as described up to now is able to work very well but only when the quadrotor moves in a free space. In fact, this solution does not still present any algorithm for the obstacle avoidance in the case where the configuration space is occupied by obstacles too.
In order to avoid misses and collisions during the mission, in this work we introduced an implementation of a *Rapidly-exploring Random Tree* (RRT) algorithm. We propose the traditional version of the RRT but we do not want to take into consideration only one final path that it provides. In fact, since we are aiming to find the path that requires less energy consumption then we test different paths in order to determine what is the minimum energy path. This test phase cannot be done before of the execution of the RRT algorithm: this algorithm provides the 3D points that are used to build the final trajectory, then it is evident that we can determine the energy spent during the mission just after that we know the points to build the path. Anyway, it does not imply any problem because the system can be used to test all the proposed trajectories and then choose the one that consumes less energy.

The RRT algorithm is one of the most used algorithms for robot obstacle avoidance. This is a probabilistic method able to analyse the configuration space where the robot has to move and then it provides the best final path from the given starting position to the final position. Unlike the multiple-query planners as PRM, this technique does not rely on the generation of a roadmap that represents exhaustively the connectivity of the free configuration space; in fact, it tends to explore only a subset of the free configuration space that is relevant for solving the problem.
It is based on a very simple randomized procedure which leads to build a tree T from the staring position to the final position that are given as input: firstly, a random configuration q_{rand} is generated according to a uniform probability distribution ***C***. Successively, the configuration q_{near} in T that is closer to q_{rand} is found, and a new candidate configuration q_{new} is produced on the segment joining q_{near} to q_{rand} at a predefined distance δ from q_{near}. A collision check is then run to verify both q_{new} and the segment going from q_{near} to q_{new} belong to the free configuration space. If this is the case, T is expanded by incorporating q_{new} and the segment joining it to q_{near}.

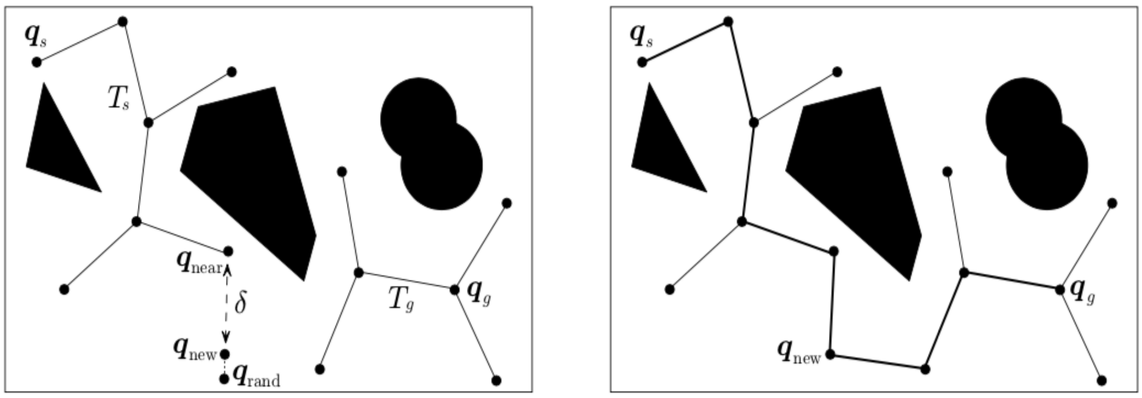


Figure 1. RRT algorithm

In our case, we propose a trajectory planning based using a *bidirectional-Rapidly-exploring Random Tree* (fig. 1). The bidirectional RRT allows to speed up the search procedure by building two different trees, one expands from the starting point and the other from the final point, when they meet in a point you found a solution.
This approach allows to generate different possible trajectories from the starting point of the UAV to the final point assigned to the mission. Moreover, a spline method was implemented and inserted into the algorithm to generate an additional smoother path. It was developed with the intention to find two different solution to the trajectory planning problem, in order to compare the shortest and smoothest solution to another one. This way, as explained previously, it is possible to compare the energy consumed travelling along both the paths and then choose the best one.

3. Implementation Design

In this section we want analyse the implementation setup and we want to show how we proceeded to develop the main idea of the project. As a reminder, the goal of this work is to build a system able to navigate in an non-free environment which can be occupied by different obstacles and simultaneously planning a safe and minimal energy trajectory in order to move the quadrotor from an initial to a final position, according to the definitions that we provided in the previous sections of the report.

As suggested more times in the report, this main goal can be divided in three sub-goals, and each of these requires a particular implementation:

1. **RRT Implementation**:
   Firstly, the RRT algorithm allows to obtain a rough safe path among the obstacles simply consisting of points connected by segments. We use the connecting points among two segments as waypoints for our problem: in fact, this phase is only required to generate a free-collision path, but it does not provide any trajectory.

2. **Energy Optimization**:
   After the generation of the path, for each subsequent couple of points, we adopt the optimization problem proposed in section 2 in order to compute the best time that allows to minimize the energy consumption during the mission.

3. **Building Final Trajectory**:
   Finally, we combine all the previous solutions by using the generated array of execution times given for all the couple of points in order to compute the final minimal energy trajectory.

In the next section we are going to analyse in detail the features of the implementation of these three steps.

3.1 RRT Implementation

In the directory of the project, the Usage.m file is where we set up the RRT algorithm. Once algorithm properties were properly defined, such as the search parameters like number of tree expansion, then it is possible to set the problems parameters, namely: initial position, final position and obstacle position. The latter is built as an imported .txt file where we define position and dimension of the obstacles defined as set of coordinates of the three dimensional points that correspond to the vertices.

Since this algorithm returns a free-collision path for a single moving point in the unsafe environment, we needed to adapt it for a trajectory planning problem for a quadrotor: for this purpose, we provide as input a modified version of the obstacles with enhanced dimension in order to produce a safer path with also a modifiable and suitable distance margin from the obstacle. This is what is generally called the "obstacle region".
After that everything was properly set up (the initial and final position cannot be in the obstacle region), we can run this algorithm and it returns a safe path defined as a set of consequents points which we use as waypoints for the next part of the project.
As mentioned in section 2, this algorithm is a faster version of the RRT algorithm since it is a bidirectional Rapidly-exploring Random Tree, and it is highlighted in the following figure.

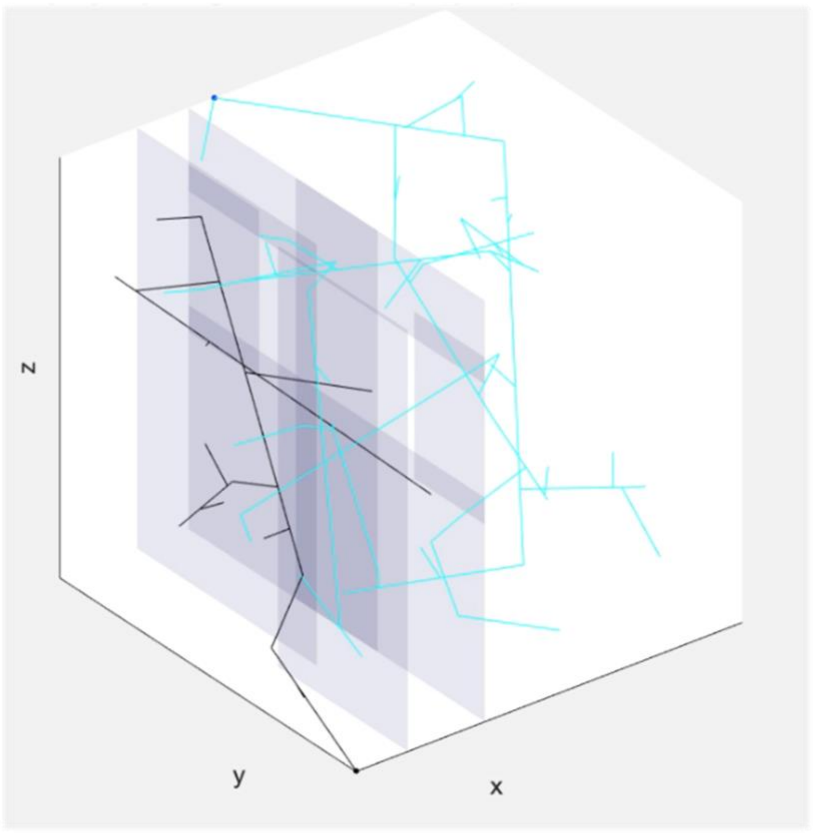


Figure 2. Implementation of the bidirectional Rapidly-exploring Random Tree

3.2 Energy Optimization

This part is the heart of the project and the work explained in this section is inspired mainly from the reference paper (Chamseddine, Join, Zhang, Theilliol, & Rabbath, 2012). As we suggested before, the energy minimization for the trajectory is obtained by considering the optimization problem for each succeeding pair of points.
This is a non-linear optimization problem and its development and execution was realized in MATLAB where we had the chance to use the *fmincon* function, namely a nonlinear programming solver that belongs to the MATLAB Optimization Toolbox.

Changing *fmincon*'s parameters cause different consequences in the whole problem execution and in the results section. A detailed analysis on these features' mechanism is provided in the results section of the report.
The most important parameters to be set are the *ConstraintTolerance*, that is the tolerance on the constraint violation, and *StepTolerance*, which is the termination tolerance on the final solution time. The analysis is provided because we noted that much of the execution time is spent to improve final time or total energy of a very small increment going to weigh to all system performance.

Once we defined these features, we kept on building the two main functions: Objective_function.m and Non_linear_constraint.m.

- In the Objective_function.m, given initial position, final position and initial time as inputs we compute the final energy and final time. The conditions on the initial and final velocity, acceleration, jerk and snap are settled independently inside these functions and varying from initial point (quadrotor start from a rest position), middle points (quadrotor moving through the path) and final position (quadrotor has to stop).
  Here we also defined the dynamic parameters of the quadrotor according to the ones expressed in the reference paper *"Sliding Mode Control of a Quadrotor Helicopter" by* Xu, Rong & Ozguner, Umit.
  After the setup of these parameters, we are able to compute the Bezier coefficients that defines the trajectory of the quadrotor, computing the flat outputs that define the system model, computing the energy but the unknown parameter t_f is still left to be solved.

- In Non_linear_constraint.m function, we define the constraints of the problem: it has the same initial structure of Objective_function.m to obtain parameters and flat outputs in order to define constraints in function of the thrusts.

These two functions, setting other problem parameters (upper bound, lower bound, initial point) are passed to *fmincon* solver which returns the time that corresponds to the minimal energy consumption.
After performing this procedure for the path (composed of *n* points), we have at the end a list of minimal times of length *n-1*.
Now we can pass this list to the last section of the project which has to build the final trajectory.

3.3 Building Final Trajectory

Now we are able to solve the optimization problem by solving the unknown parameter t_f that allows to minimize the final consumed energy.

In the file Build_trajectory.m we use the previous results to build the final minimal energy trajectory. As specified in the theory chapter, given initial and final conditions and final times for each points pair we're able to build the desired trajectory using flatness property. In fact, once computed Bezier's coefficients, we utilize the flat outputs corresponding to cartesian coordinate to generate the final trajectory. In this file we implement also the simulation part of the project and the plot arrangement.

4. Simulation Results

In the previous section, we presented both the background theory and the implementation details for solving the proposed trajectory planning problem. We described the safety concept and the minimum energy goal and now, we are going to evaluate the efficiency of the proposed solution.
In this section, we will show the simulation results of the project: we are going to introduce several plots where we wanted to highlight the properties of the system, advantages and disadvantages.

In order to show the efficiency of our method, we are going to present the results in different phases:

- The first proposed results represent the *fmincon* solver function used in MATLAB for solving the optimization problem.

- The second section of results will test the efficiency of the method in different scenarios. We will evaluate if the system is able to find the safe and minimum energy trajectory in environments with different sets of obstacles. This second section of results is linked to another section that could be considered as a third part: in fact, each experiment produced in the particular scenario will be deeply evaluated by analysing the UAV's behaviour during the mission.

All the results are collected by setting the parameters of the systems as follows:

$$I_1 = I_2 = 1.25\frac{Ns^2}{rad}\ ; \quad I_3 = 2.5\frac{Ns^2}{rad}; \quad m = 2\ kg\ ; \quad l = 0.2\ m\ ; \quad g = 9.8\ m/s^2$$

3.1 *The fmincon* Solver: Evaluation of the System Behaviour

As mentioned in the section 3.2, the *fmincon* solver allows in MATLAB to define and implement the optimization problem. Thank to this function, we are able to solve the problem by computing the time for the minimal energy trajectory.

The system behaviour is correlated to the setting of this function and the computational behaviour of this latter is conditioned two main parameters: the tolerance on the constraint violation called "*ConstraintTolerance*"; the termination tolerance on the final solution time called "*StepTolerance*".

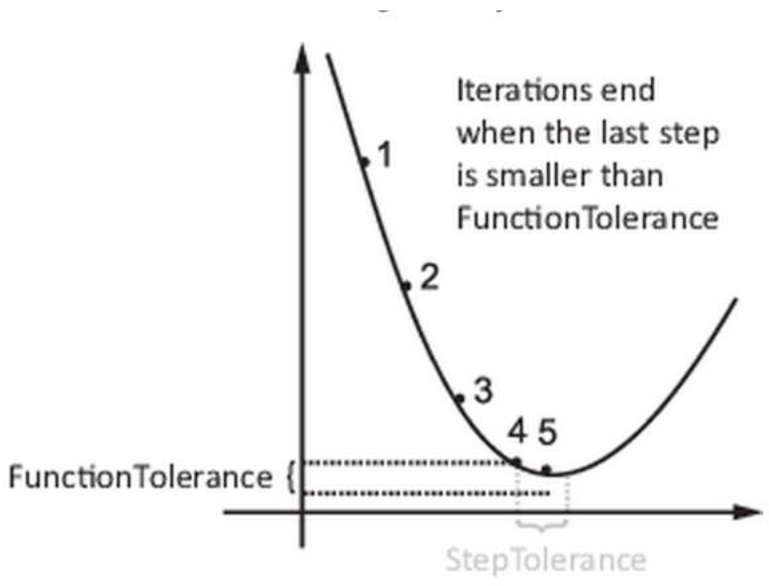


Figure 3. fmincon mechanism

We evaluated the behaviour of the system by setting different combinations of these two parameters in order to better understand what the best settings are to solve properly our control problem.

In our system we do not consider the safety margin parameter ρ, but since the idea is to introduce this concept as an improvement of the system, we decided to run these tests with different values of ρ.

The following table of results takes into consideration a fixed scenario to evaluate the behaviour of the system. The environment of this scenario is characterized by the presence of three obstacles. These experiments are really focused on the efficiency of the optimization problem and not in finding the free-collision path, so it is normal to assume always the same scenario.

Exp	ρ	StepTol	ConsTol	Optimization time(s)	Execution time(s)	Final Energy
1	0	1e-06	1e-06	220.0473	18.9340	20.6975
2	0	1e-06	1e-02	218.7798	18.9340	20.6975
3	0	1e-02	1e-06	181.8210	18.9356	20.6944
4	0	1e-02	1e-02	182.1381	18.9356	20.6944
5	0.1	1e-06	1e-06	204.1705	19.0993	20.7162
6	0.1	1e-06	1e-02	193.9819	19.0993	20.7162
7	0.1	1e-02	1e-06	177.0472	19.1198	20.7165
8	0.1	1e-02	1e-02	143.5410	19.0954	20.7132
9	0.1	1e-09	1e-06	203.2202	19.0993	20.7162
10	0.2	1e-02	1e-02	134.8000	20.4518	21.1434
11	0.2	1e-01	1e-01	105.1248	19.9965	20.9374
12	0.2	1	1e-02	147.7760	20.6652	21.2541
13	0.2	1e-02	1	168.7099	20.4518	21.1434
14	0.2	1	1	53.8318	15.7395	24.8048
15	0.3	1e-01	1e-01	139.1152	22.1353	22.1683

These results show how the parameters that define the *fmincon* solver affects the final time of the optimization problem, the execution time and the final consumed energy. Even though is not considered as a main result of the control problem, the execution time is really important too when we have to deal with on-board trajectory planning.
The table shows the existing trade-off between computational cost and effectiveness of the solution: the more the optimization time increases the more the quality of the solution increases too, even if the rate among them is highly unbalanced.

The first four experiments represent the system as it was implemented and then run for all the experiments presented in this report. The 5-15 experiments represent optimization control problem implemented as follows:

$$\begin{cases} \text{Minimize} & E(t_f) - E(t_0) \\ \text{Subject to} & \rho \le T^*_{i_{Ext}} \le 1-\rho \quad (i = 1, \dots, 4) \\ & \rho \le T^*_i(t_0) \le 1-\rho \\ & \rho \le T^*_i(t_f) \le 1-\rho \end{cases}$$

As we can see, the value of the safety margin parameter affects the constraints applied to the thrusts in order to avoid that the actuators hit their limits.
The safety margin parameter represents the model uncertainties, and this is why its value is not easy to compute. We provide an overview of the system affected by the application of this parameter with different values.

It is evident that the best solutions are obtained when the safety margin parameter is equal to 0 or to 0.1. This represents how our system was implemented and tested because this approach allows to satisfy all the problem constraints at the same time.
If we set $\rho = 0.2$ we would obtain worse results but safer: in fact, generally the results would increase of a certain amount, but anyway we would obtain a satisfiable and safer solution.

Since we assume that 0.2 is a realistic value to be applied to the safety margin parameter, we highlighted the experiment 11 as the one that returns the best trade-off between the two *fmincon* parameters. The next results will be produced by running several simulations where we will compare the system with and without the application of the safety margin parameter. As explained before, we assume that the realistic best value is 0.2, and we are going to use this value for testing the system.

3.2 Safe and Minimum-Energy Trajectory

In the following experiments we will present the trajectory planned by the system to be tracked by the UAV. The system considers all the constraints that we have presented in the section 2 of this report, in order to return the safe, free-collision, minimum-energy trajectory.

These experiments are conducted in three different scenarios characterized by a different composition of the environment: in fact, we are going to evaluate the behaviour of the system when it has to plan a trajectory in a free environment and in an environment with one or three obstacles. In order to evaluate, the behaviour of the system, we will analyse different features of the quadrotor: the velocity and the acceleration produced during the mission and, especially, the thrust produced by the UAV rotors. All these components are analysed in order to understand what is the UAV's behaviour that produces the minimum energy consumption.

3.2.1 No-obstacles scenario

The first proposed scenario is the easiest scenario which can be encountered by the quadrotor. In fact, here, we consider a free environment where the quadrotor starts from an initial position and it only has to reach an assigned final position.
This is a particular scenario because in this case a part of the algorithm can be completely removed: in fact, the external safety part of the solution that produces a free-collision path adopting the RRT algorithm can be completely ignored because there is no chance of collision. The computational weight of the method benefits from discarding this part the algorithm and it can be understood in terms of execution time.

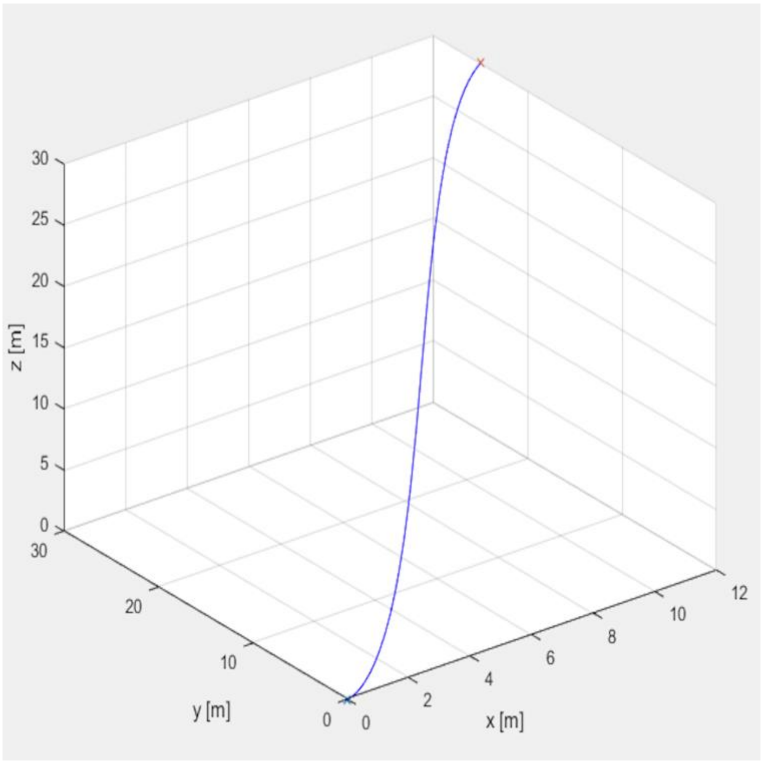
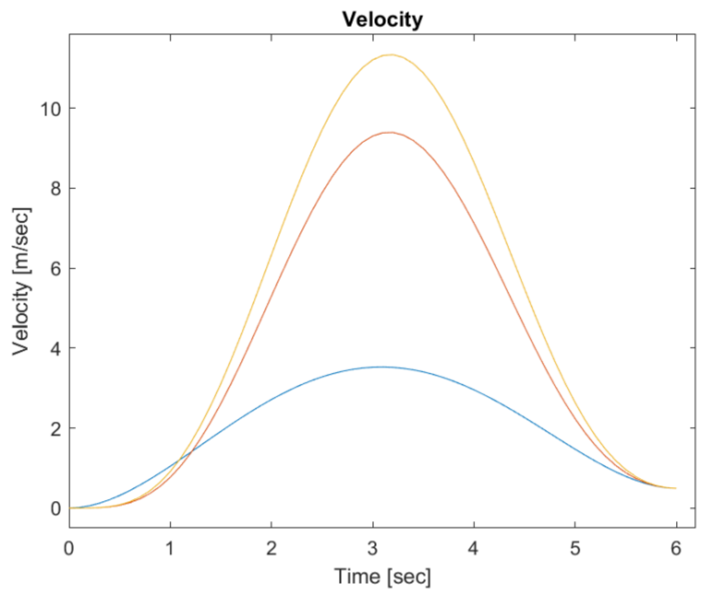
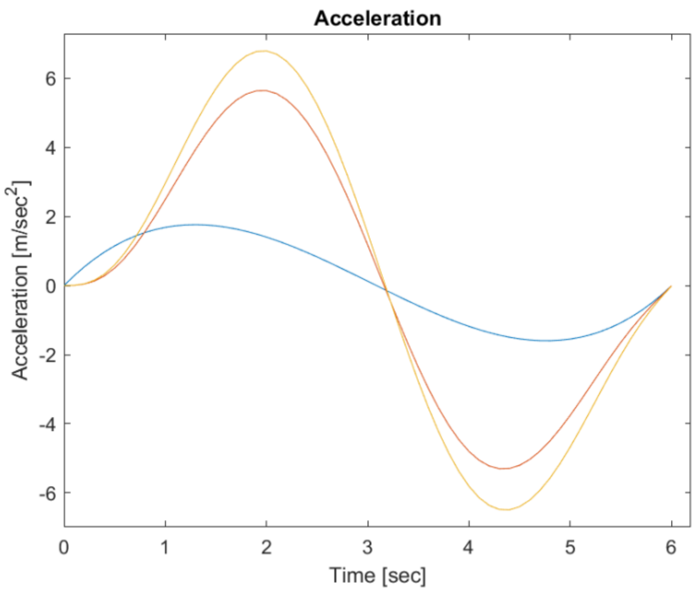


Figure 4. No-obstacle scenario: Planned Trajectory

The figure 4 shows the planned trajectory that the UAV should track in order to consume the minimum amount of energy without hitting the constraints. Even though the environment is free, it is not always correct to assume that the minimum-energy trajectory coincides with the minimal time trajectory. In fact, it depends on the dynamic behaviour of the quadrotor.

In the following figures, it is possible to analyse the velocity and the accelerations produced by the UAV's rotors to achieve the task. These results are used to understand the behaviour of the system to consume the minimum amount of energy.





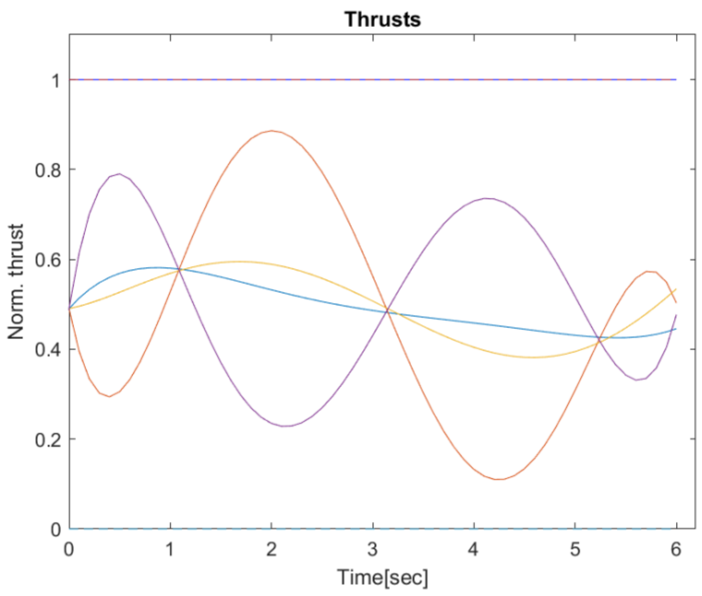


Figure 5. No-obstacles scenario (from top to the bottom):

Velocity results; Acceleration results; Normalized Thrusts results

The velocity and acceleration results represent the classical results obtained from a rest-to-rest planning problem. The most significant result is offered by the graph of the thrust: it is possible to see that they do not hit the actuator constraints and they neither get close to the maximum value. Thanks to these dynamics characteristics the system is able to plan a trajectory that can be tracked in 6,07 s and the energy is minimally consumed spending only 6,82 J.

As mentioned in the previous section, we are going to test the system even in presence of the safety margin parameter. The application of this additional is expected to produce worse results but it allows to generate safer trajectories to be tracked by the UAV.
The following graph shows how the system behaves when the new constraints are introduced.

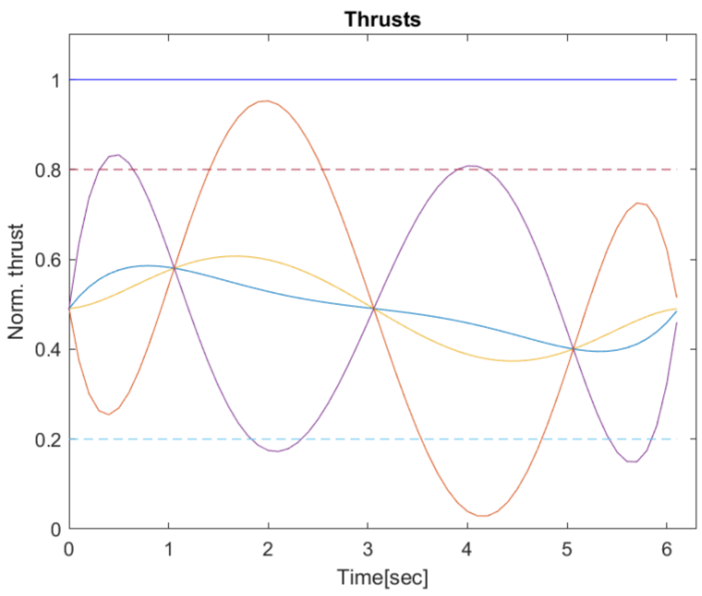


Figure 6. No obstacles scenario with ρ=0.2: Normalized Thrusts

The obtained results confirm almost all the expectations: in fact, we got a longer final time equal to 6,11 s but a larger amount of energy consumed equal to 6,79 J. Unfortunately, the results shown in the previous graph are not valid because the system does not respect the safety margin constraint. We investigated to figure out this problem, but we were not able to solve it: as we will mention later, this problem can be addressed to the choice of the fmincon parameters and this is why our purpose is to optimize this problem in the future.

However, we propose another solution where the results are really worse than the previous ones, but all the constraints are satisfied.

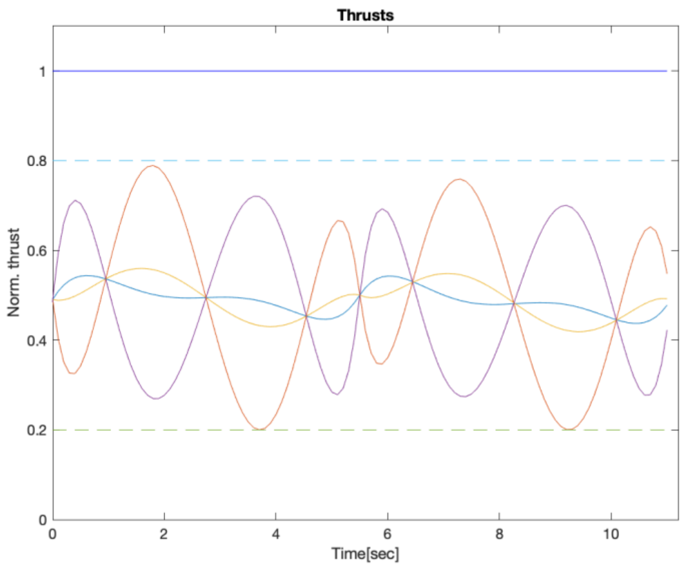


Figure 7. No obstacles scenario with ρ=0.2: Normalized Thrusts (Three points)

This approach implies the use of three points, instead of two, to define the path from the starting point to the final position. The solution respects all the imposed conditions and it returns a final time equal to 11.05 s and the amount of consumed energy is equal to 11,34 J.

Anyway, we think that the free environment is not a good scenario to test significantly the introduction of these parameters, but in the next results it will be possible to better appreciate this variation of the system.

3.2.2 One-obstacle scenario

Now we propose the results obtained running the simulations in the second scenario characterized by the presence of one obstacle. In the following figure, the safe and minimum energy trajectory that was planned by the system is represented. The starting and final points are the same that we set for the previous experiments, but the introduction of the obstacle causes the change in the planned trajectory with respect to the previous case and it is expected that this change will produce an increment of the final time and the consumed energy.

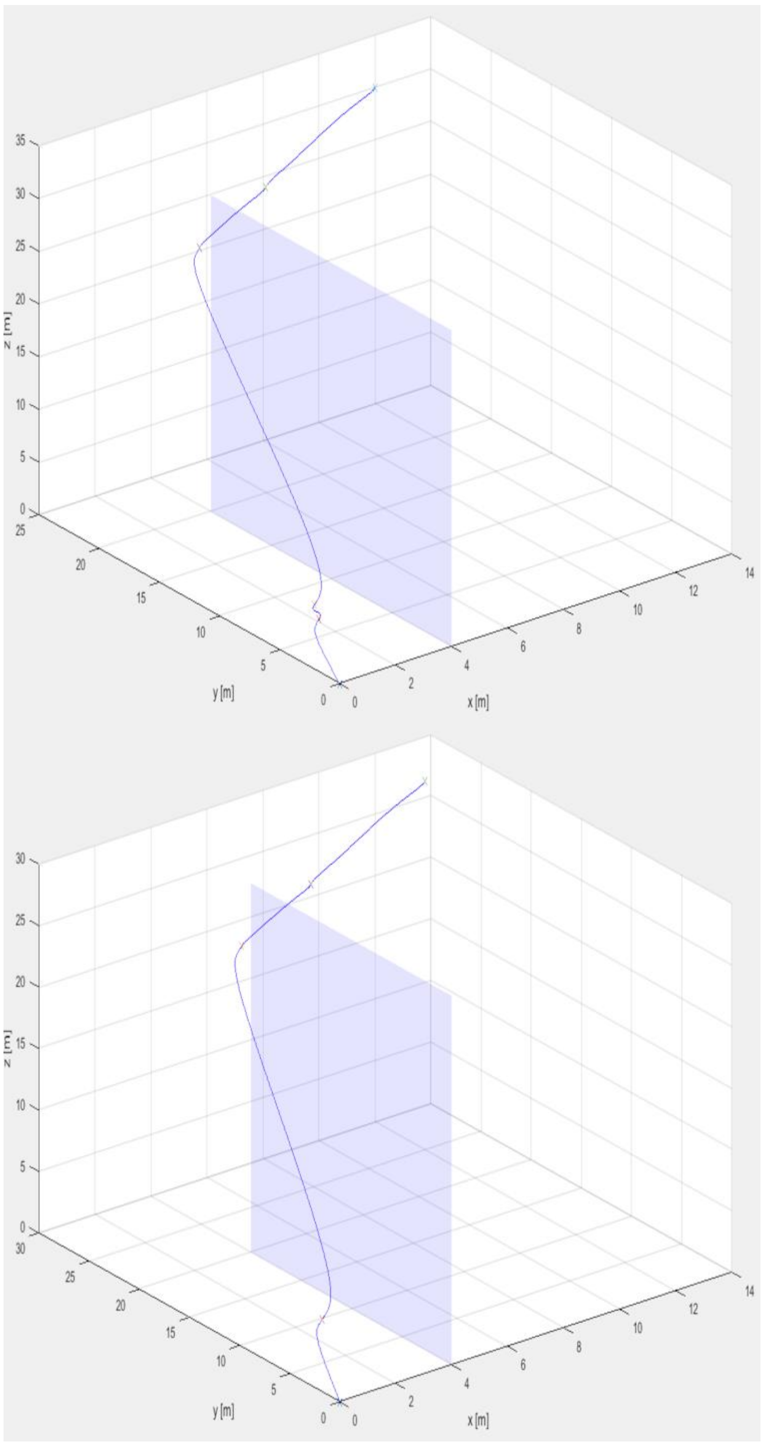


Figure 8. One-obstacle scenario (from top to the bottom): Original Trajectory; Smooth Trajectory

In the case with no obstacles, we planned one single trajectory because the emptiness of the environment made easy to understand that the considered trajectory is surely the both the minimum time and minimum energy trajectory. In this case, instead, we consider two trajectories: the RRT algorithm in fact is able to compute an additional smoother trajectory. We decided to test both these trajectories in order to compare the results obtained by the application of the original trajectory and the smoother one.

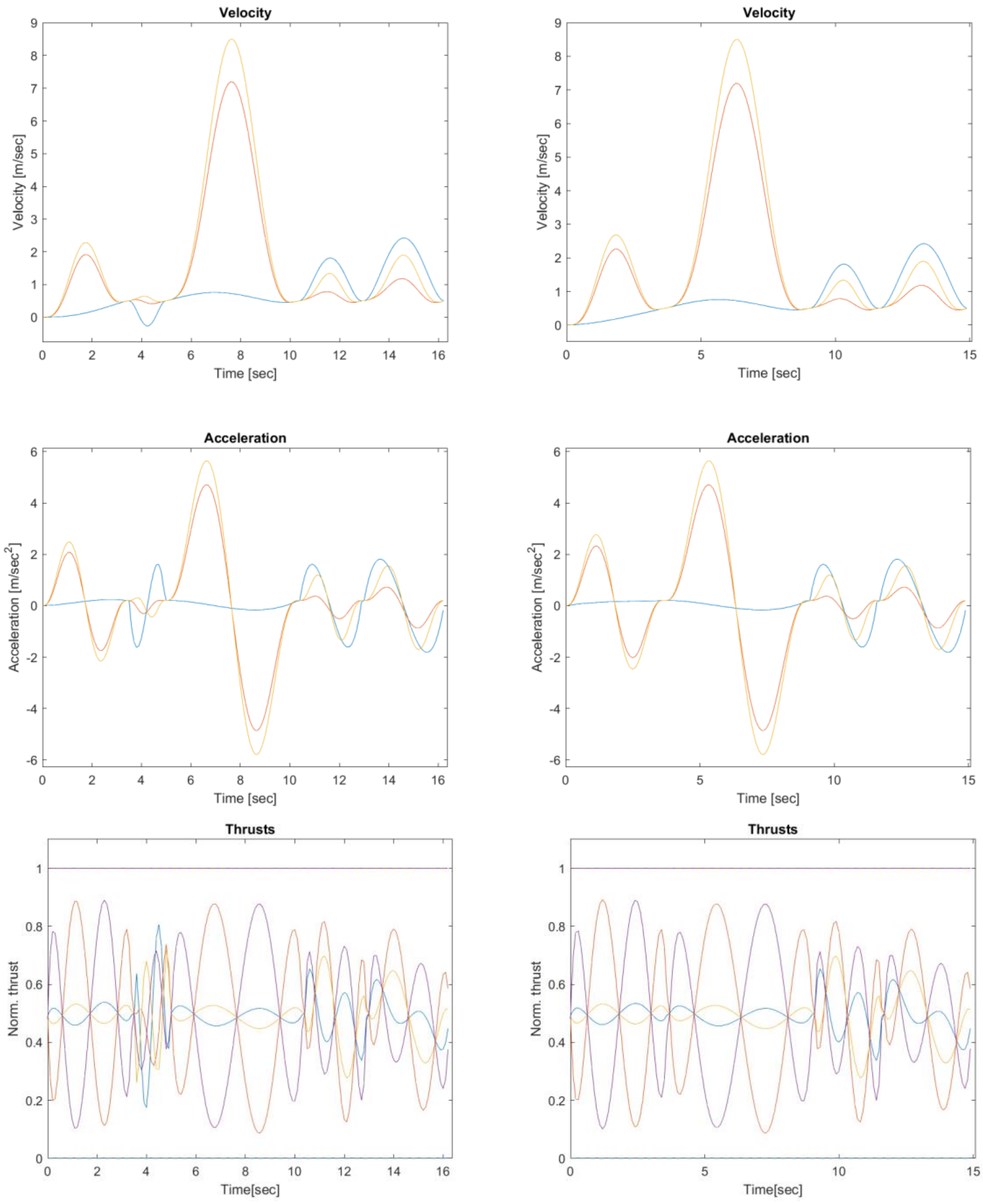


Figure 9. One-obstacle scenario (from top to the bottom; original trajectory results on the left and smooth trajectory results on the right): Velocity results; Acceleration results; Normalized Thrusts results

In the previous graphs, we can compare the performances of the UAV during the tracking of the two proposed trajectories. The final time and the energy consumed during the two missions are respectively 15,98 s and 17,26 J for the original trajectory and 14,73 s and 15,75 J for the smooth trajectory. Indeed, it is evident that the minimum energy trajectory is represented by the smooth trajectory in the scenario characterized by only one obstacle. These results are confirmed in particular by the accelerations and normalized thrusts graphs: in fact, these images clearly represent the additional maneuvers adopted by the UAV during tracking the original trajectory. This can be translated in terms of energy consuming, and then it explains why the smooth trajectory is the minimum energy trajectory.

Even in this case, we wanted to test the system when we apply the safety margin parameters. In order to test this case, we take into consideration the smooth trajectory because is the best result that we got.
The final time and the energy consumed by the UAV are respectively 15,94 s and 16,38 J. These results are obtained by applying the thrust represented in the next graph.

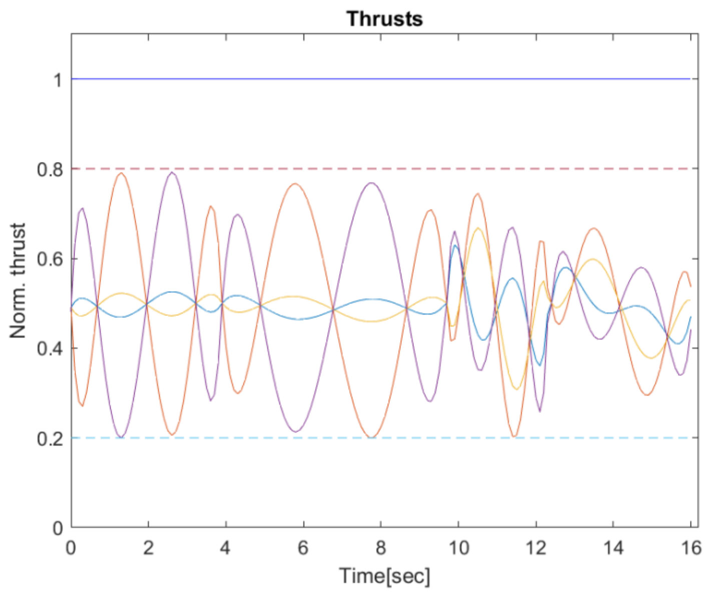


Figure 10. One-obstacle scenario with ρ =0.2: Normalized Thrusts results

Even in this case, the system works properly and in fact all the thrusts respects the limits imposed by the safety margin parameter. These limits are the cause of the increase of final time and energy consumed with respect to the smooth trajectory planning without the safety margin constraint.

3.2.3 Three-obstacles scenario

The final scenario that we are going to represent, as mentioned before, is characterized by the presence of three obstacles. This scenario is the main test bench to prove the efficacy of our method. In fact, navigating in an environment full of obstacles can be quite tricky but our system is able to produce several free-collision paths that generate the desired trajectories, as shown in figure 11.

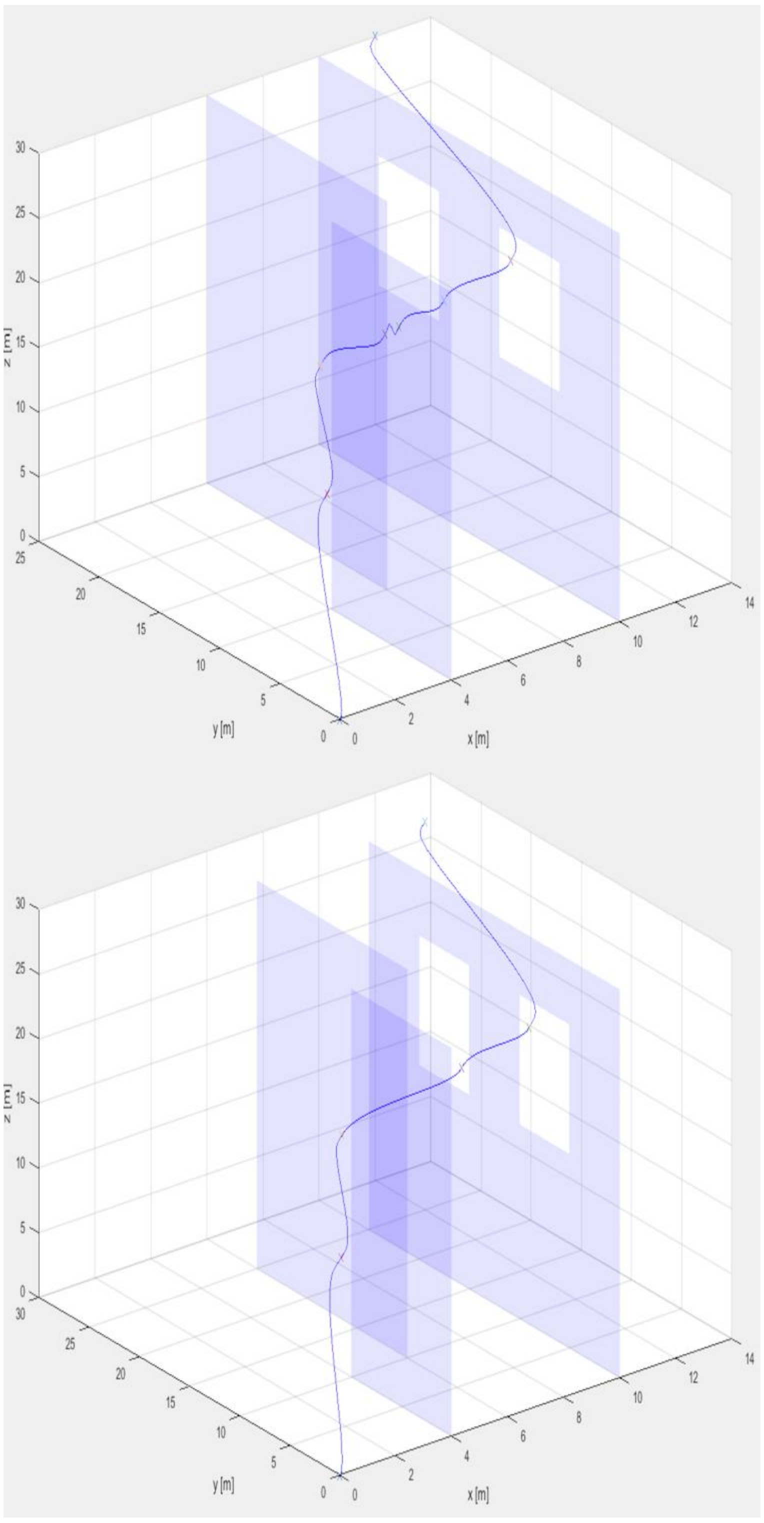


Figure 11. Three-obstacles scenario (from top to the bottom): Original Trajectory; Smooth Trajectory

As we did for the previous case, we are going to compare the performances of the UAV for two different safe planned trajectories: a normal trajectory and a smooth trajectory.

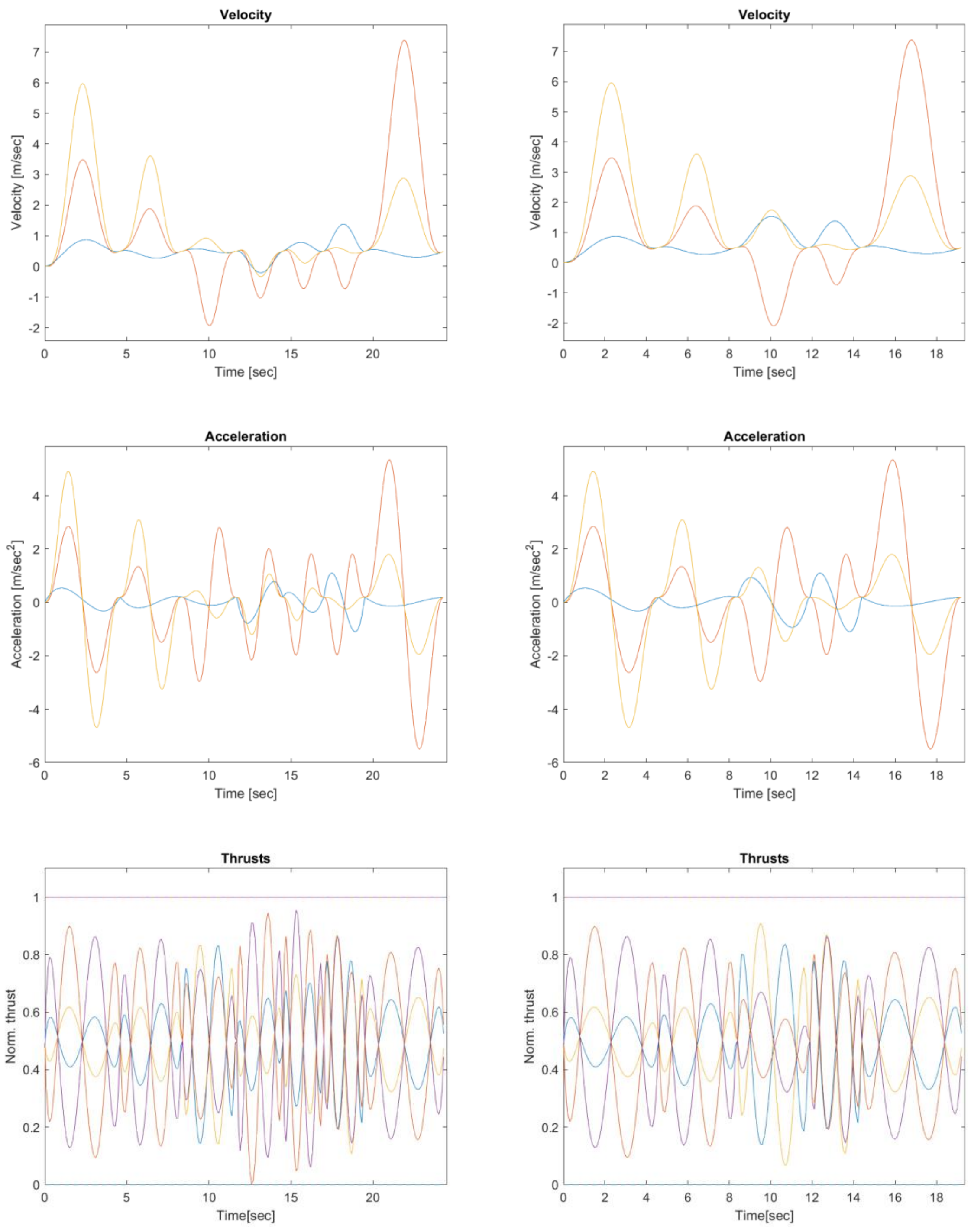


Figure 12. Three-obstacles scenario (from top to the bottom; original trajectory results on the left and smooth trajectory results on the right): Velocity results; Acceleration results; Normalized Thrusts results

In this scenario, it is immediately evident from the normalized graph that the best results in terms of energy are produced by the smooth trajectory. In fact, it is

highlighted even in the representation of the planned trajectory that the number of maneuvers to do are much more than in the smooth case, and it causes the increment of the energy spent during the mission.
As supposed, in fact, the results produce by both these trajectories are 24,06 s and 26,89 J for the original trajectory and 19,03 s and 21,03 J for the smooth trajectory.

The big gap between the two performances is highlighted by testing the smooth trajectory with the application of the safety margin parameter.

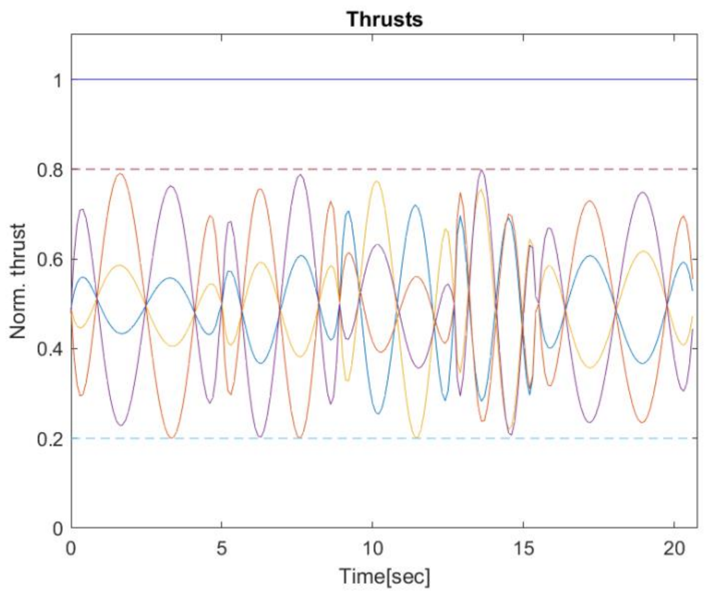


Figure 13. Three-obstacles scenario with ρ=0.2: Normalized Thrusts results

Even here the safety margin constraints are properly respected. Contrarily to the scenario with one obstacle where the safety margin test produced intermediate results between the two proposed trajectories, here the returned results are much closer to the smooth trajectory results. In fact, we obtain a final time equal to 20,48 s and the energy consumed during the travel is equal to 21,40 J. It highlights the differences between the performances related to the original trajectory and the one related to the smooth trajectory, because even though we introduce new constraints the system returns much better results than in the original trajectory case.

This final table wraps up all the previous results in order to give an overview on the run simulations to test our system:

Exp	Scenario	Time (s)	Energy (J)
1	No obstacles	6.0708	6.8226
2	No obstacles (ρ=0.2)	6.1185	6.7911
3	1 obstacle: Original Trajectory	15.9814	17.2603
4	1 obstacle: Smooth Trajectory	14.7346	15.9529
5	1 obstacle: Smooth Trajectory (ρ = 0.2)	15.9414	16.3889
6	3 obstacles: Original Trajectory	24.0648	26.8944
7	3 obstacles: Smooth Trajectory	19.0392	21.0348
8	3 obstacles: Smooth Trajectory (ρ = 0.2)	20.4849	21.4088

5 Conclusion

The use of Unmanned Aerial Vehicles is limited by the lifetime of the included batteries, and this lifetime depends on the rate at which the energy is consumed. In fact, one of the most important limitation in the UAVs field is the reduced flight endurance that is typically between 15 and 30 minutes.
In the literature many methods have been proposed to alleviate this problem, but in this project we wanted to show a novel approach that allows to face this problem from an algorithm point of view.
In this work we focused on trajectory planning problem in order to save energy and extend endurance of the device: we presented a safe and minimum energy trajectory planning method that allows to compute a free-collision path from a given starting position to a given final position minimizing at the same time the energy consumed during the flight.

In this report we showed he different steps that were implemented to find a solution to our problem: the heart of the solution corresponds to the realization of a optimization problem used to minimize the energy during the trajectory; we introduced the concepts of *internal* and *external safety* and they were developed respectively using additional constraints in the control problem in order to avoid to hit the actuators limits and implementing a RRT algorithm used to avoid collisions during the path. This latter is able to provide different paths such that we had the chance to test if the minimum energy path always corresponds to the shortest one.

After that we have introduced all the theorical background behind this project and once that we have defined the main features that characterize our implementation, we started to run several simulations in different scenarios in order to evaluate the effectiveness of our algorithm.

The optimization problem was developed thanks to the use of the *fmincon* Matlab function and the first part of the results section was presented as a sort of "calibration test" where we wanted to check what are the best parameters combinations to get the best results.
After this preliminary phase we started to collect several results in order to check the performances obtained during the simulations.
These results showed that the algorithm is always able to find the minimum energy path between two assigned configurations even in presence of different obstacles. Moreover, we showed the results associated to the UAVs performances where we have collected the thrusts, the velocities and the accelerations. These results confirmed that the devices are able to find the minimum energy path in a safe way: in fact, when we set the most difficult scenario, it is characterized by the presence of three obstacles and the path generation leads to several energy consuming maneuvers, but the system is always able to plan a free-collision path minimizing the energy without hitting the actuator constraints, and it means that the mission is always accomplished according to all the assigned conditions.

Thanks to this solution we achieved all the assigned tasks but there is still something to improve: we noted that the computation time given by the system sometimes it appears to be quite long. We think that the solution to this problem could be either finding an alternative to *fmincon* Matlab's function or to find the optimal configuration that allows to improve the speed of the algorithm.

Since the objective of this study is to derive safe and minimum energy trajectory planning both for offline and online applications, for this latter it would be really important to improve these features.
In this work, our main goal was to implement and test a control system able to face many realistic problems: for this reason, as mentioned before, we want to concentrate on the development of a more accessible and faster algorithm.

References

Chamseddine, Abbas & Zhang, Youmin & Rabbath, Camille & Join, Cédric & Theilliol, Didier. (2012). *Flatness-Based Trajectory Planning/Replanning for a Quadrotor Unmanned Aerial Vehicle*. IEEE Transactions on Aerospace Electronic Systems. 48. 2832-2848. 10.1109/TAES.2012.6324664.

Sudhakara, Priyanka & Ganapathy, V. & Sundaran, Karthika. (2017). *Optimal trajectory planning based on bidirectional spline-RRT∗ for wheeled mobile robot*. 10.1109/SSPS.2017.8071566.

Omerdic, Edin & Roberts, Geoff. (2004). *Thruster fault diagnosis and accommodation for open-frame underwater vehicles. Control Engineering Practice*. 12. 1575-1598. 10.1016/j.conengprac.2003.12.014.

Fabio Morbidi, Roel Cano, David Lara. *Minimum-Energy Path Generation for a Quadrotor UAV.* IEEE International Conference on Robotics and Automation, May 2016, Stockholm, Sweden.

Abdilla, Analiza & Richards, Arthur & Burrow, S.G.. (2015). *Power and endurance modelling of battery-powered rotorcraft*. 675-680. 10.1109/IROS.2015.7353445.

Jongerden, Marijn & Haverkort, Boudewijn. (2009). Which battery model to use?. IET Software. 3. 445-457.

DuToit Ryan, Holt Matt, Lyle Megan, Biaz Saad. *UAV Collision Avoidance Using RRT* and LOS Maximization Technical Report #CSSE12 – 03.*

Xu, Rong & Ozguner, Umit. (2007). *Sliding Mode Control of a Quadrotor Helicopter. Proceedings of the IEEE Conference on Decision and Control*. 4957 - 4962. 10.1109/CDC.2006.377588.

Roberts, James & Zufferey, Jean-Christophe & Floreano, Dario. (2008). *Energy Management for Indoor Hovering Robots.* 1242 - 1247. 10.1109/IROS.2008.4650856.

Kirk Donald E. (1970), *Optimal Control Theory: An Introduction.* Prentice-Hall, Inc.

Lee, Dasol & Shim, David. (2014). *RRT-based path planning for fixed-wing UAVs with arrival time and approach direction constraints.* 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings. 317-328. 10.1109/ICUAS.2014.6842270.

Fliess, Michel & Lévine, Jean & Martin, Philippe & Rouchon, Pierre. (1995). *Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples*. International Journal of Control. 61. 13-27. 10.1080/00207179508921959.

Bortoff, Scott. (2000). *Path planning for UAVs*. American Control Conference, 2000. Proceedings of the 2000. 1. 364 - 368 vol.1. 10.1109/ACC.2000.878915.

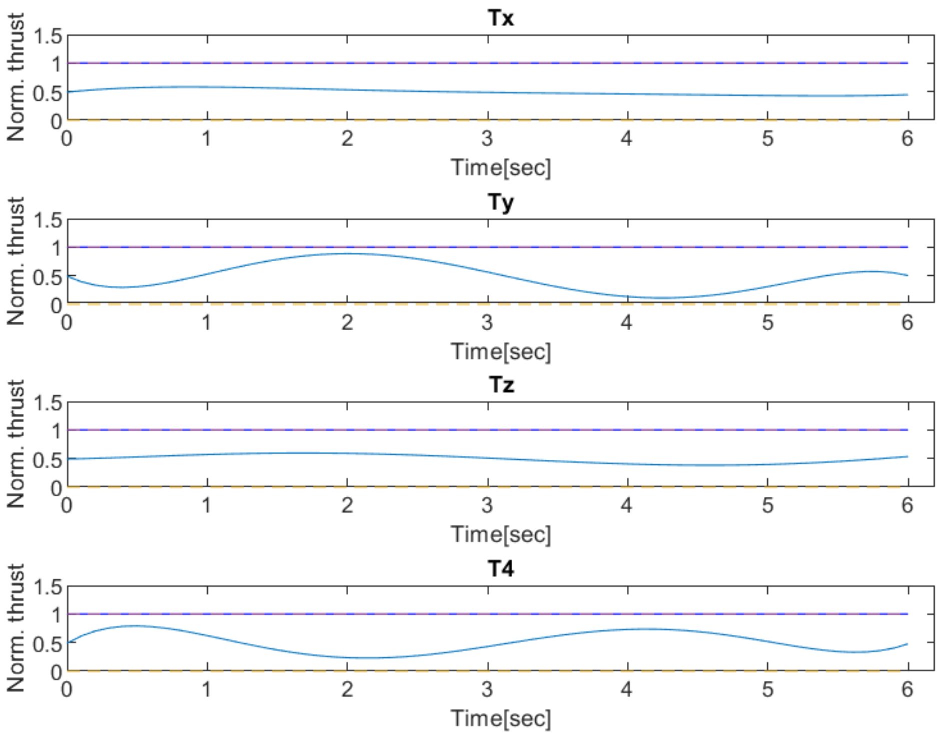
Lai, L-C., Yang, C-C., and Wu, C-J.
Time-optimal control of a hovering quad-rotor helicopter. Journal of Intelligent and Robotic Systems, 45 (2006), 115—135.

Hagenmeyer, V. and Delaleau, E.
Continuous-time non-linear flatness-based predictive control: An exact feedforward linearisation setting with an induction drive example.
International Journal of Control, 81, 10 (2008), 1645—1663.

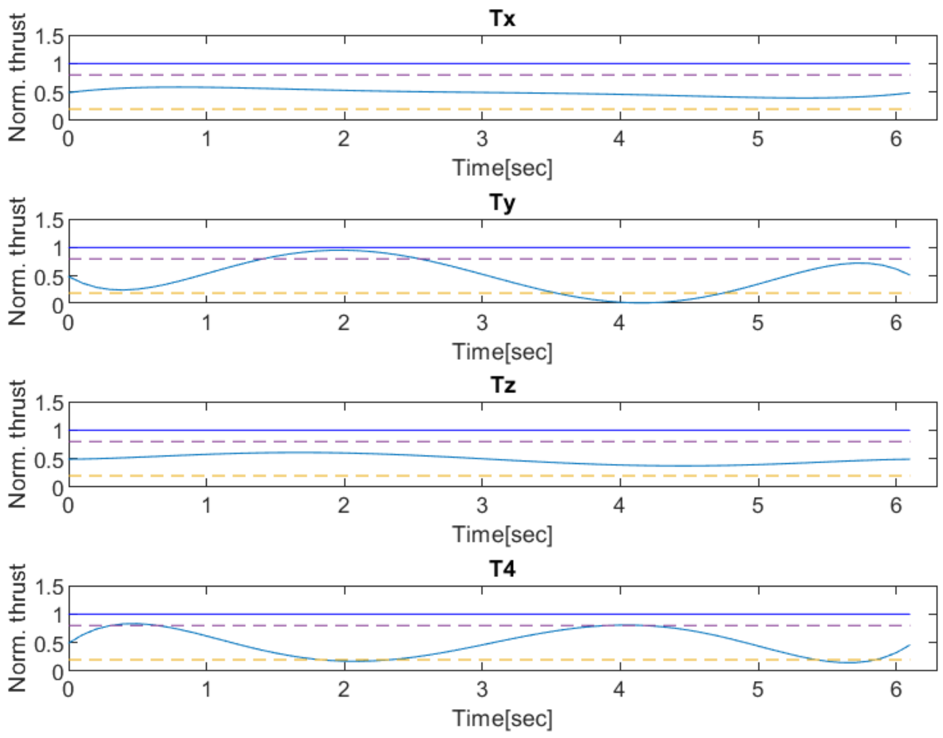
Zoladek, H.
The topological proof of Abel-Ruffini theorem. Journal of the Juliusz Schauder Center, 16 (2000), 253—265.

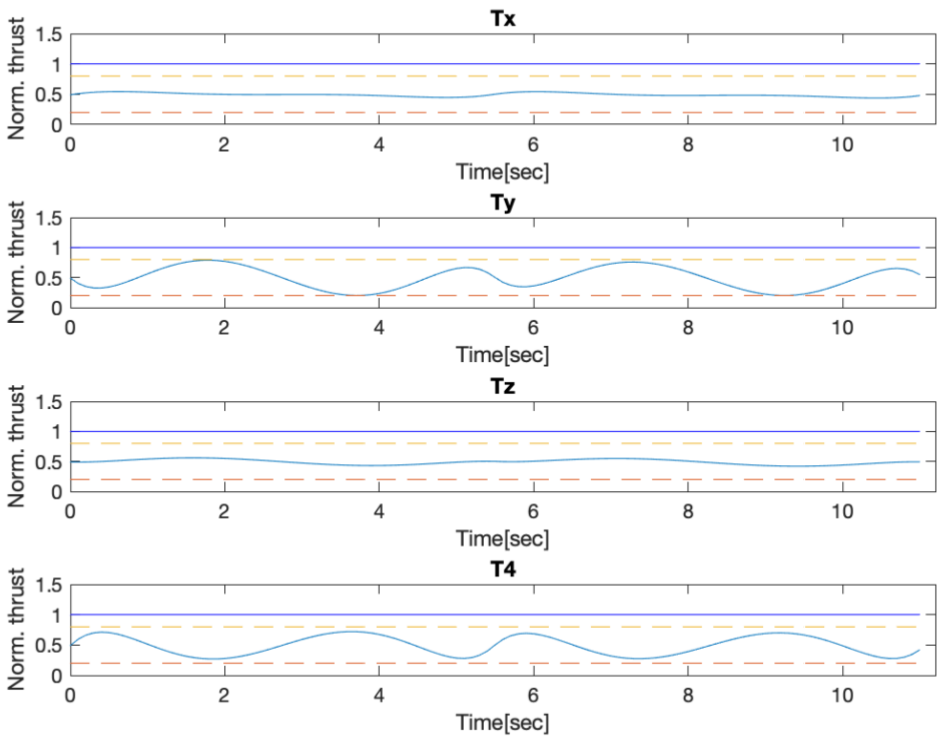
Appendix

A. No-obstacles scenario: Normalized thrusts components

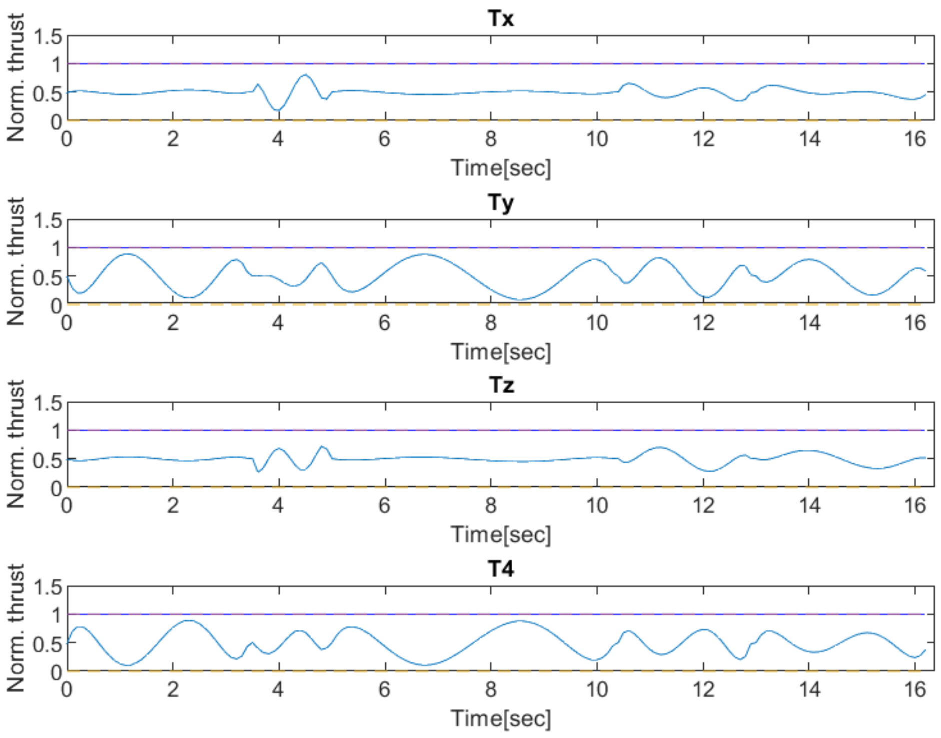


B. No-obstacles scenario with ρ = 0.2: Normalized thrusts components

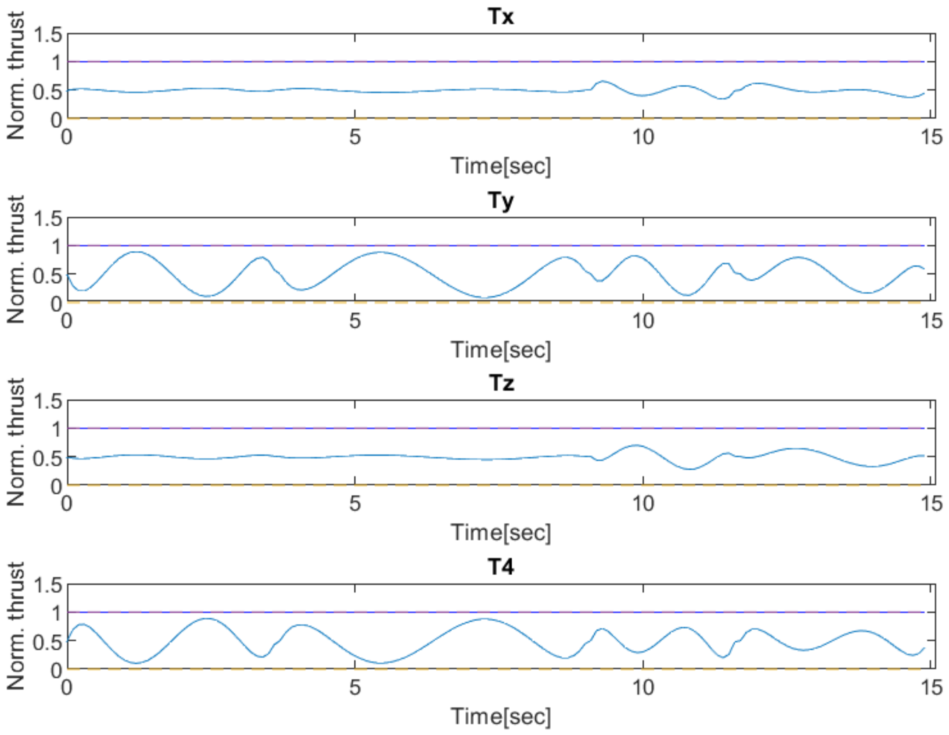




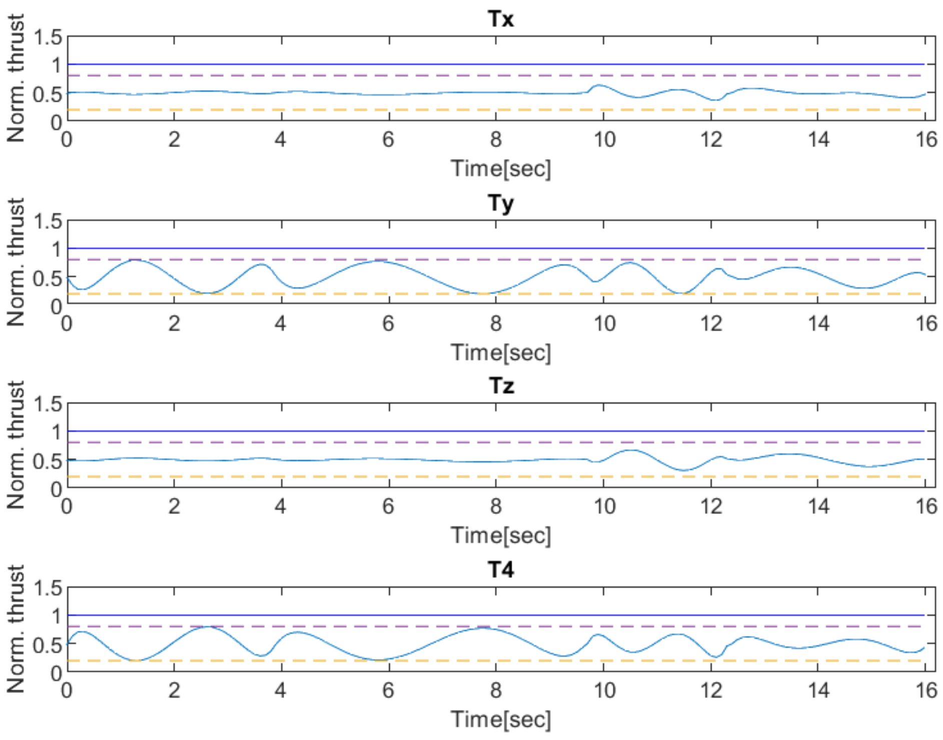
C. One-obstacle scenario (Original Trajectory): Normalized thrusts components



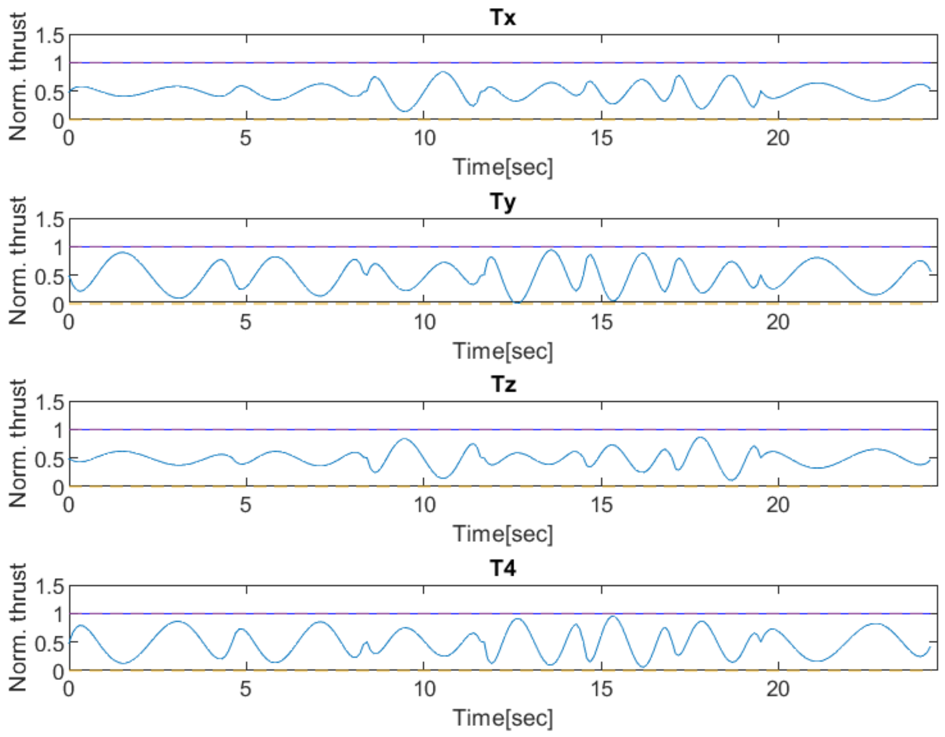
D. One-obstacle scenario (Smooth Trajectory): Normalized thrusts components



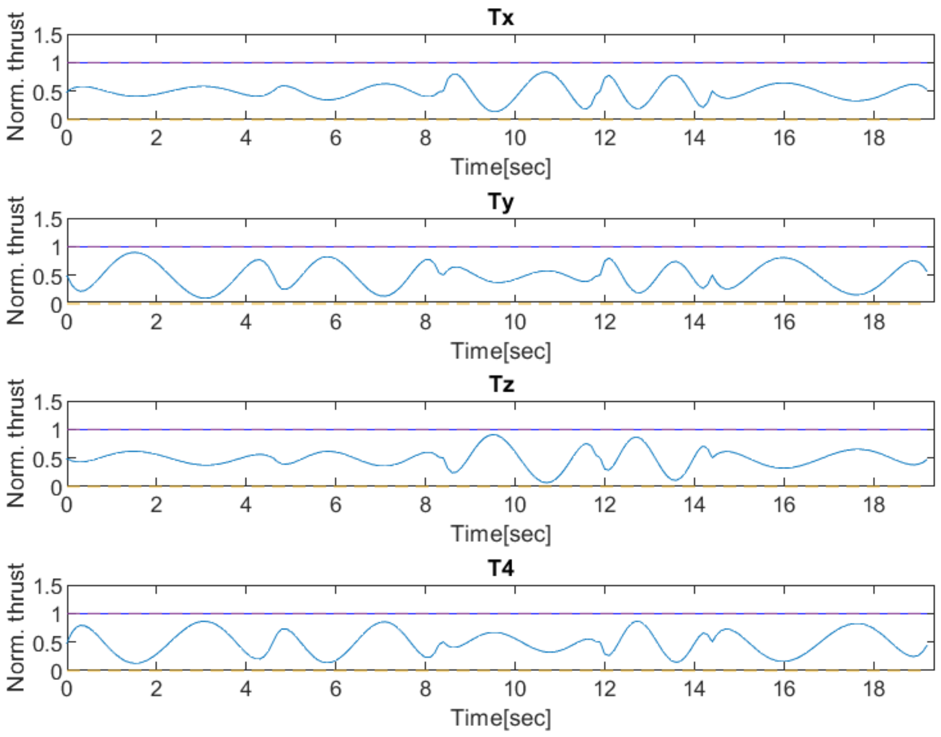
E. One-obstacle scenario (Smooth Trajectory) with ρ = 0.2: Normalized thrusts components



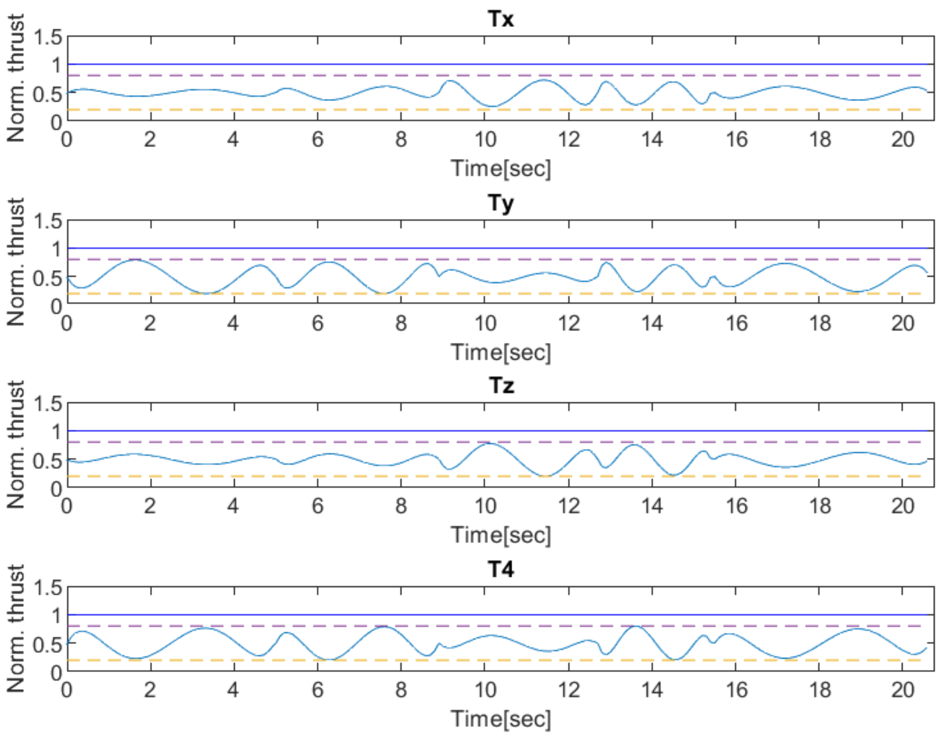
F. Three-obstacle scenario (Original Trajectory): Normalized thrusts components



G. Three-obstacle scenario (Smooth Trajectory): Normalized thrusts components



H. Three-obstacle scenario (Smooth Trajectory) with ρ = 0.2: Normalized thrusts components


Tx
Ty
Tz
T4
Norm. thrust
Time[sec]