University POLITEHNICA of Bucharest
Faculty of Aerospace Engineering
„Nicolae Tipei" Department of Aeronautical Systems Engineering and Aeronautical Management

Flight Path Planning for a UAV Formation with Energy Consumption Minimization

PhD Thesis

Răzvan-Viorel Mihai

Coordinator:
Prof. Dr. Eng. Adrian-Mihail Stoica

Bucharest, April 2021

Contents

Contents **iii**

Mathematical Convention Notations **v**

1 Introduction to the Flight Path Planning Problem **1**
1.1 General considerations on flight path planning methods for a UAV formation 1
1.2 Cooperative and distributive flight path planning 2
1.3 Purpose and main objectives of the doctoral thesis 2

2 Methods for Flight Path Generation and Planning **5**
2.1 'Cell Decomposition' method 5
2.2 'Road Map' method 6
2.2.1 Visibility graph 6
2.2.2 Voronoi diagram 6
2.3 Potential field method 6
2.4 RRT, RRT* and IRRT* methods 7
2.5 Method based on curve interpolation 8
2.5.1 Bézier curves 8
2.5.2 B-Spline curves 8
2.6 Numerical optimization planning 9
2.7 Conclusions 10

3 Quadcopter Model **11**
3.1 Forces and moments acting on the quadcopter 11
3.2 Quadcopter attitude representation 12
3.3 Representation of angular speeds 12
3.4 Quadcopter modeling using the Newton-Euler equations 12
3.5 Conclusions 12

4 Quadcopter Control **13**
4.1 Altitude control 14
4.2 Control of the quadcopter's 2D motion 14
4.3 3D motion control 15
4.4 Conclusions 18

5 Design, Realization and Experimental Testing of the Quadcopter Autopilot **19**
5.1 Main components . . . 19
5.2 Designing, building and programming the automatic control system . . . 20
5.3 Analysis of experimental data . . . 21
5.3.1 Flight results analysis after tracking a 2D trajectory . . . 22
5.3.2 Flight results analysis after tracking a 3D trajectory . . . 25
5.4 Conclusions . . . 26

6 Flight Path Planning with Energy Consumption Minimization in the Absence of Obstacles **27**
6.1 Preparatory elements for calculating flight path using Euler-Lagrange equations . . . 27
6.2 Running a mission with a planned flight trajectory, while minimizing the acceleration of the translation/traction force acting on the structure of the quadcopter . . . 28
6.3 Running a mission with a planned flight path, while minimizing the variation of the translation acceleration/traction force or the moments acting on the structure of the quadcopter . . . 31
6.4 Estimation of energy consumption when carrying out missions with planned flight trajectory, in different scenarios to minimize variables related to quadcopter dynamics . . . 31
6.5 Conclusions . . . 32

7 Flight Path Planning with Energy Consumption Minimization in the Presence of Obstacles **33**
7.1 Theoretical considerations on convex optimization . . . 34
7.2 Planning the trajectory avoiding static obstacles, for an UAV performing solitary flight . . . 34
7.3 Planning the trajectory for a two member flight formation, with inter-collision avoidance . . . 40
7.4 Planning the trajectory for a two member flight formation, with inter-collision and static obstacles collision avoidance . . . 41
7.5 Conclusions . . . 42

8 Results and Conclusions **43**
8.1 General conclusions . . . 43
8.2 Contributions . . . 43
8.3 Prospects for further development . . . 46

Bibliography **47**

Mathematical Convention Notations

Scalar constants: $\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{A}, \mathrm{B}, \mathrm{C}$ etc.

Scalar variables: a, b, c etc.

Vector constants: $\mathbf{a}, \mathbf{b}, \mathbf{c}$, etc.

Vector variables: $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ etc.

Matrices: $\mathbf{A}, \mathbf{B}, \mathbf{C}$ etc.

Coordinate systems or frames: A, B, C, etc.

Rotation matrix from frame B to A, ${}^{A}\mathbf{R}_{B}$

Continuous time: $t \in \mathbb{R}$

Discrete time: $n \in \mathbb{Z}$ sau $k \in \mathbb{Z}$

Continuous time argument: (t)

Discrete time argument: $[n]$ sau $[k]$

Functions: $f, \mathbf{g}$

Chapter 1

Introduction to the Flight Path Planning Problem

The literature presents a multitude of solutions to the flight path planning problem for UAVs, each approach having its own advantages and disadvantages, depending on the particularity of the mission they must perform. Most approaches follow the block scheme from fig. 1.1. The three arrows on the left represent the input data of the planning component,

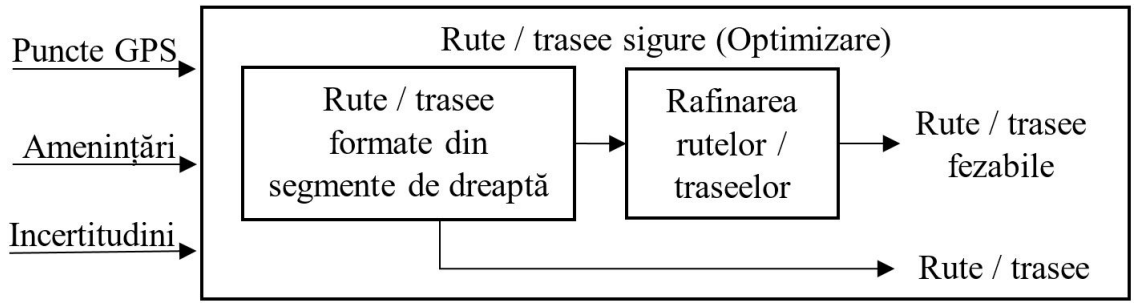


Figura 1.1: Diagram addressing the flight path planning issue

i.e. GPS points (navigation points), the threats that may be posed by collisions with static and dynamic obstacles and uncertainties. The structure of the block "Rute/trasee sigure" suggests that the achievement of the flight path often takes place in two phases:

1) Obtaining simple routes/paths, in the form of straight lines, obtained by direct interconnection of navigation points;
2) Refine simple routes/paths in order to obtain a feasible, curvilinear-shaped trajectory.

1.1 General considerations on flight path planning methods for a UAV formation

Table 1.1 [1] presents a summary of the literature's research work on the most important trajectory planning algorithms.

Table 1.1: Review of trajectory planning algorithms [1]

Methods categories	**Method name**	**Associated algorithms**	**References**
Graph search planning	'Cell Decomposition'	Dijkstra	[2], [3], [4], [5], [6], [7]
	'Road Map'	A^*	[8], [9], [10], [11], [12]
Continuous function planning	Potential Field		[13], [14], [15], [16]
Sampling based planning	Rapidly exploring Random Tree (RRT)		[17], [18]
	Rapidly Exploring Random Tree star (RRT*)		[19], [20], [21]
	Informed Rapidly Exploring Random Tree star (IRRT*)		[22], [23], [24]
Curve interpolation-based planning	Bézier curves		[25], [26]
	B-Spline curves		[27], [28], [29]
Numerical optimization based planning	Numerical optimization		[12], [30], [31], [32]

1.2 Cooperative and distributive flight path planning

Flight path planning for formation flight can be performed by following two separate procedures: i)Cooperative planning; ii)Distributive planning.

1.3 Purpose and main objectives of the doctoral thesis

The purpose of the doctoral thesis is to study the problem of flight path planning from the basic theoretical elements and successively going through all the steps necessary to develop an experimental planning application for a formation of UAVs: a)Theoretical and numerical study of flight path planning methods; b)Optimized flight path planning for a single UAV (mathematical model UAV development; design, numerical simulation and experimental validation of UAV automatic control system; design of trajectory planning algorithms with energy optimization in different flight scenarios, numerical simulation of them and implementation of planning results obtained on the experimental automatic control system of the UAV); c)Optimized cooperative flight path planning for a formation of UAVs (design trajectory planning algorithms with optimizing energy consumption in different flight scenarios and validating them by numerical simulation). For the design and

validation of trajectory planning algorithms, we have designed a flight mission to complete all of the above-mentioned steps. In this flight mission the working scenario that I have followed is presented schematically in fig. 1.2. Among the details of the design and conduct

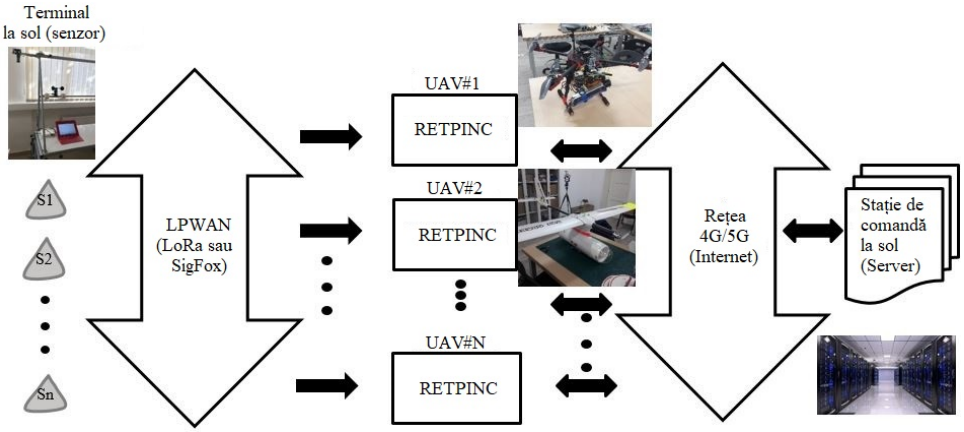


Figure 1.2: Working scenario adopted in the framework of the flight mission [32]

of the flight mission, the following may be stated:

- Using a network of n sensors $S_1, \ldots, S_n$ located on agricultural land, which must provide data to the human user to monitor the condition of the harvest and make predictions on maximum production;
- The range for the wireless transmission of harvest status information is limited for each sensor, depending on the technology used to make the transmission;
- The Band of UAVs flies over the area where the [33] sensors are located to collect the data acquired by them, based on a flight path, generated by the trajectory planning algorithm, which runs on board each member. The REal Time Processing and INternet Communication Unit (RETPINC) block equips each of the members of the [34] flight group;
- Through the UAV formation, sensor data reaches the control station using a 4G/5G high-speed wireless link;
- Through the same 4G/5G communications networks, the Ground Command Station (SCS) receives information about the current state P_t^i (where t represents the time at which the reception is made, and i, represents the uAV index/number) of each band member and the obstacles detected in flight. Based on this information, manoeuvring sets are generated for UAVs involved in a collision conflict.

In close correlation with the above, during the doctoral thesis I propose to achieve the following objectives:

OB1: Theoretical presentation and numerical simulation of methods of generation and flight path planning;
OB2: Modeling a member of the quadcopter flight formation;

OB3: Description of a flight controller that allows easy software implementation;
OB4: Validation of the flight controller model by numerical simulations;
OB5: Experimental autopilot for multirotor UAV;
OB6: Experimental testing of the developed autopilot;
OB7: Planning, calculating and numerical simulation of a flight mission in different scenarios to minimize variables related to the dynamics of the aircraft;
OB8: Developing a method of planning the flight path with the minimization of energy consumption for a quadcopter in the absence of obstacles;
OB9: Assessment by numerical simulations and demonstration of the performance of the flight path planning algorithm with the minimization of energy consumption in the absence of obstacles;
OB10: Developing a method of planning the flight path with the minimisation of energy consumption for a quadcopter in the presence of obstacles;
OB11: Assessment by numerical simulations and demonstration of the performance of the flight path planning algorithm with energy consumption minimization for a quadcopter in the presence of obstacles;
OB12: Development of a method of planning the flight path with energy consumption minimization for two members of the flight group in the absence of obstacles;
OB13: Assessment by numerical simulations and demonstration of the performance of the flight path planning algorithm with energy consumption minimization for two members of the flight group in the absence of obstacles;
OB14: Developing a method of planning the flight path with the energy consumption minimization for two members of the flight group in the presence of obstacles;
OB15: Assessment by numerical simulations and demonstration of the performance of the flight path planning algorithm with the energy consumption minimization for two members of the flight group in the presence of obstacles.

Chapter 2

Methods for Flight Path Generation and Planning

The methods exhibited, initially developed in applications with terrestrial robots, have been taken up and perfected over time in aerial vehicle applications, with or without a human crew [35]. These fall into five broad categories and are primarily used to generate maps for the flight path planning algorithm. To generate the results from the application of the methods studied in this chapter, we used the hardware and software resources from tab. 2.1.

Table 2.1: Resources used to generate results from the application of the methods listed in the [35]

Nr. crt.	Resource name	Resource type
1	i7-8750H@2.20GHz processor	Hardware
2	16 GB RAM	Hardware
3	x64bit Ubuntu 18.04	Operating system
4	Visual Studio Code	Integrated Development Environment
5	Python3	Programming language

2.1 'Cell Decomposition' method

In the case of this method, the workspace is divided into cells. The next step generates possible routes that pass through adjacent free cells. Free cells are those that are not occupied by obstacles. Obstacles are isolated by identifying connectivity between free cells. This produces a discrete version of the environment. A search algorithm is used to connect adjacent free cells. The shaded (crossed) cells are removed because they are occupied by obstacles (grey).

2.2 'Road Map' method

Applying this method is the first step of the trajectory planning component. The method involves creating a two-dimensional network of straight lines that links the initial and final points without intersecting the obstacles defined in the map. In other words, it is a representation of a 2D space. For a given starting point $\boldsymbol{p_s}$ and the end point $\boldsymbol{p_f}$, all possible connection routes are generated avoiding obstacles. A M^*metric is defined that will join the generated routes. This metric may consider for example the shortest route between the two points, the minimum number of interconnected routes, etc.

2.2.1 Visibility graph

As the name suggests, the visibility graph produces a direct line of visibility through an environment. This is one of the oldest methods used for trajectory planning. For this method, only polygonal obstacles are considered. For a $\boldsymbol{G = (V, E)}$, where $\boldsymbol{V}$ represents the vertexes of obstacles, while $\boldsymbol{E}$ represents the edges of obstacles (nodes connected with straight lines). Therefore, only nodes that are visible (in the sense that each peak can be seen another) are included in the connectivity network.

2.2.2 Voronoi diagram

A Voronoi chart is a graph of connectivity generated by the formation of polygons around obstacles. Each edge of the polygons is defined by first building a set of lines connecting the obstacle centers. The set of polygons is then built by drawing a set of lines perpendicular to the lines joining the obstacles. They are then adjusted to meet at a minimum set of nodes.

2.3 Potential field method

This method requires a specific description of the map or workspace so that known obstacles are described in the form of polygons. This time, the workspace will no longer be described discreetly, as a cell map, but using continuous form functions:

$$U(q) = U_{atr}(q) + U_{res}(q) \tag{2.1}$$

where $U_{atr}(q)$ characterizes the function describing 'attraction' between 'potential' of the start and end point, while $U_{res}(q)$ characterizes the function of 'rejection' against obstacles along the [13] trajectory.

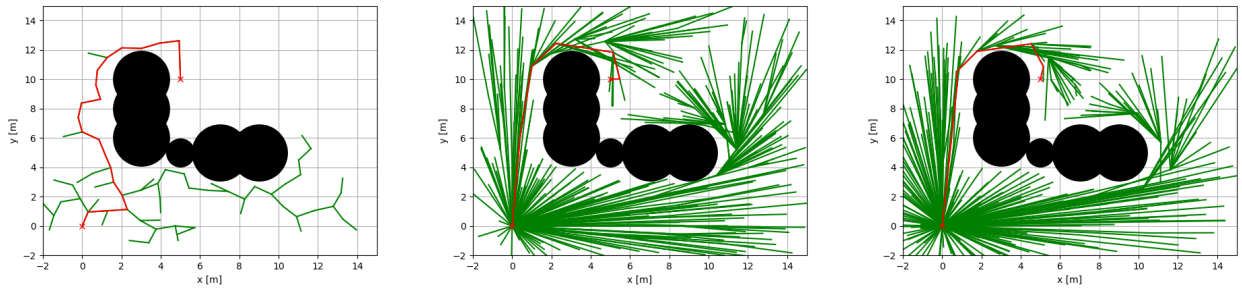
2.4 RRT, RRT* and IRRT* methods

This method belongs to the category of trajectory planning methods, using randomly sampled points. According to [36], both the RRT method and the derived forms of this algorithm follow the procedure in the algorithm (1). The above algorithm starts with the

Algorithm 1 RRT algorithm [37]

function ALGORITMRRT($\boldsymbol{p_{initial}}$)
  $\boldsymbol{T} \leftarrow InitializareRetea()$;
  $\boldsymbol{T} \leftarrow InsereazaPunct(\emptyset, \boldsymbol{p_{initial}}, \boldsymbol{T})$;
  $i \leftarrow 0$;
  $N \leftarrow FixatDeUtilizator()$;
loop:
  if $i \leq N$ **then**
    $\boldsymbol{p_{aleatoriu}} \leftarrow Esantioneaza(i)$;
    $\boldsymbol{p_{apropiat}} \leftarrow DeterminaPunctApropiat(\boldsymbol{T}, \boldsymbol{p_{aleatoriu}})$;
    $\boldsymbol{p_{nou}} \leftarrow AdaugaPunct(\boldsymbol{p_{apropiat}}, \boldsymbol{p_{aleatoriu}})$;
    if $VerificaColiziune(\boldsymbol{p_{nou}})$ **then**
      $\boldsymbol{T} \leftarrow InsereazaPunct(\boldsymbol{p_{min}}, \boldsymbol{p_{nou}}, \boldsymbol{T})$;

InitializareRetea function, with a role in initializing the graph or network that will contain $\boldsymbol{p_{initial}}$ after running the *InsereazaPunct* function. The variable i is used to account for the current number of iterates, a number that will be compared to a reference value fixed by the user via the N variable. RRT is an algorithm that is based on the random selection of points in the workspace. Then, determine the nearest point on the existing $\boldsymbol{T}$ network, compared to $\boldsymbol{p_{random}}$. The $\boldsymbol{p_{new}}$ point that is added to the existing network results from the selection between $\boldsymbol{p_{close}}$ and $\boldsymbol{p_{random}}$, running the *AdaugaPunct* function, which evaluates which of the two points is closer to the destination point of the trajectory.



(a) RRT (b) RRT* (c) IRRT*

Figure 2.1: RRT, RRT* and IRRT* methods

Finally, verify that the newly obtained point is outside the space occupied by obstacles,

and if so, it is added to the $\boldsymbol{T}$ network. Starting from the algorithm (1), the RRT* [21], IRRT* [23] algorithms, and the numerical results we obtained are illustrated in fig. 2.1.

2.5 Method based on curve interpolation

2.5.1 Bézier curves

According to [38], obtaining a curvilinear-shaped flight path can be generated using the Bézier curves. The general form of the expression for a *nth* order Bézier curve is:

$$P(t) = \sum_{i=0}^{n} B_i^n(t)\boldsymbol{P_i}, \qquad t \in [0,1], \tag{2.2}$$

where $\boldsymbol{P_i}$ represents the control points so that $\boldsymbol{P}(0) = \boldsymbol{P_0}$ and $\boldsymbol{P}(1) = \boldsymbol{P_n}$, and $B_i^n(t)$ is a polynomial of the form:

$$B_i^n(t) = C_n^i(1-t)^{n-i}t^i, \qquad i \in 0, 1, \ldots, n. \tag{2.3}$$

The use of Bézier curves to calculate the trajectory of a robot has the following advantages:

- The trajectory will include the points $\boldsymbol{P_0}$ to $\boldsymbol{P_n}$;
- Curves joining points $\boldsymbol{P_0} \to \boldsymbol{P_1} \to \ldots \to \boldsymbol{P_{n-1}} \to \boldsymbol{P_n}$ are tangent to the corresponding segments in $\boldsymbol{P_0}$ and respectively $\boldsymbol{P_n}$;

2.5.2 B-Spline curves

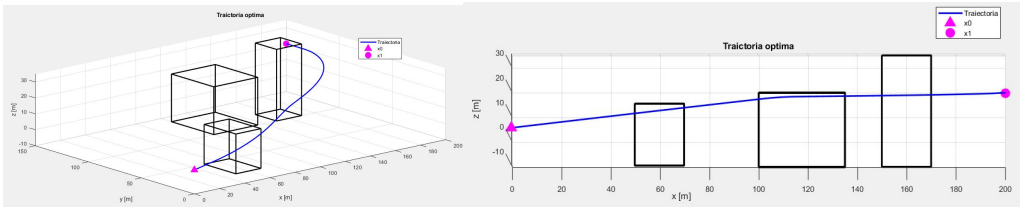
The disadvantage of the trajectory determined by the Bézier curve interpolation algorithm is that the control points on the basis of which the Bézier curve is generated will not be visited by the UAV. Even if the trajectory planning using curves 'B-spline' does not solve the problem of visiting the control points, the advantage that this method offers over the Bézier curve refers to the efficiency of the algorithm's execution time, which is not affected by the increase in the number of control points.

2.6 Numerical optimization planning

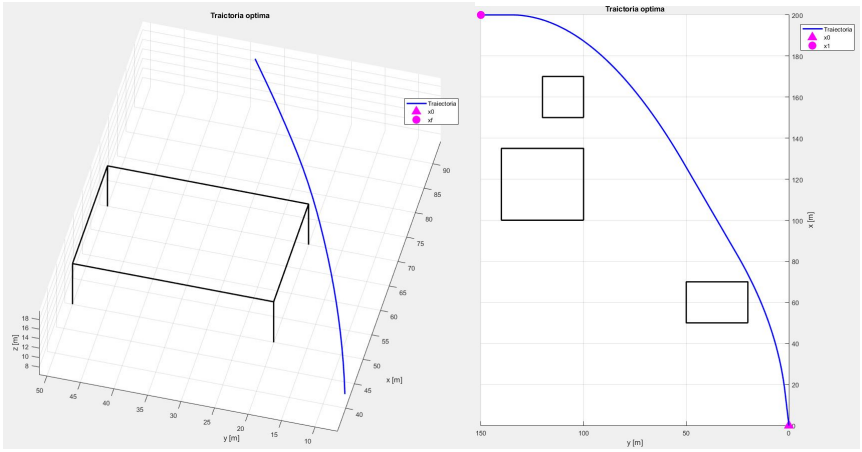
Current techniques based on numerical optimization to solve the path planning problem are represented by search methods in the workspace such as acrfullmilp, genetic algorithms [39], etc. MILP is a numeric optimization method called linear programming with integer or binary restrictions. These restrictions are used for logical decisions, such as 'left turn' or 'right turn'. This method plays a role in generating a safe path for a vehicle or vehicle formation, depending on the application [40], [41].

In the figure fig. 2.2 is illustrated a flight scenario for a UAV, the calculated and simulated trajectory of which is highlighted by the blue colored curve. The starting and

final points of the flight mission are represented using triangular and circular symbols, respectively, colored with magenta. The three static obstacles with black sides have been added additionally to assess the performance of the flight path planning method under collision avoidance conditions. The condition for avoiding static obstacles is satisfied during the entire simulation, and this can be seen in the figure 2.2c.



(a) 3D view (b) 2D view- ***xz***



(c)3D view-up (d)2D view- ***xy***

Figure 2.2: Feasible flight path for a UAV, calculated with the MILP method [32]

2.7 Conclusions

The numerical results, obtained from algorithms resulting from the trajectory planning methods, the specifics of the flight mission, and the model of the UAV as a member of the flight formation, led to the IRRT* algorithm being considered for adding numerical optimization elements.

Table 2.2: Advantages and disadvantages of trajectory planning algorithms.

Method or algorithm	**Advantage**	**Disadvantage**
Dijkstra	Determines the shortest trajectory based on the points or nodes of a map.	Searching for routes is not heuristic. The algorithm runs slower and slower as the number of destinations increases. The trajectory obtained is not continuous.
A^*	It's based on the Dijkstra algorithm. It uses a heustic function, which reduces execution time.	The trajectory obtained is not continuous. The flight path planning problem is not solvable every time.
RRT, RRT*, IRRT*	It generates solutions even for multidimensional systems. Consistently converges to the solution and identifies the trajectory if it exists and if the allotted time is sufficient	The trajectory obtained is not continuous. It depends significantly on the time allotted for execution.
Polynomial curves	Efficient as execution time. Concatenation of several curves is possible.	Determining optimal coefficients specific to $\geq$4 curves is difficult.
Bézier curves	Efficient as execution time. Curve can be intuitively manipulated through control points.	As the order of the curve increases, so does the execution time. Used mainly in two-dimensional space.
B-Spline curves	Efficient as execution time. Continuous and controllable trajectory through control points.	Obtaining a continuous curvature may violate the restrictions. Avoid planning trajectories in obstacle environments.
Numerical optimization	Consider all restrictions and minimize the cost function formulated.	Requires a significant calculation resource because the algorithm runs at millisecond time intervals.

Chapter 3

Quadcopter Model

In the image fig. 3.1 are highlighted the rotational directions of the propellers, whose rotational speeds are denoted with ω_i, the forces generated by the motor-propeller assemblies, denoted with $\boldsymbol{F_i}$, the resistant torque of the propeller denoted with $\boldsymbol{M_i}$. The axis verses of $\boldsymbol{xyz}$ are marked with $\mathbf{a_x a_y a_z}$, and the angles of attitude, characterizing the roll, pitch and yaw rotations, are denoted with ϕ, θ, ψ.

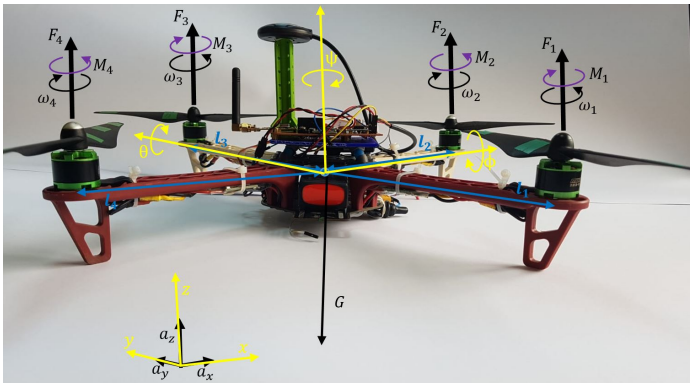


Figure 3.1: Representation of motor rotation and direction of the resulting thrust forces

3.1 Forces and moments acting on the quadcopter

Thrust force can be written as $\boldsymbol{F} = \sum_{i=1}^{4} \boldsymbol{F_i} - \boldsymbol{G} = \sum_{i=1}^{4} \boldsymbol{F_i} - \mathrm{m}\boldsymbol{g}$, in which m represents quadcopter mass, $\boldsymbol{g}$ gravitational acceleration. Moments around the center of gravity of the quadcopter, considering each arm of the structure, can be written as $\boldsymbol{M} = \sum_{i=1}^{4}(r_i \boldsymbol{F_i} + \mathrm{k_M}\boldsymbol{\omega_i}^2)$. At equilibrium (the situation of the flight at a fixed point and in the absence of rotational motion), the resultant forces and the resulting moment acting on the structure of the quadcopter are zero.

3.2 Quadcopter attitude representation

To describe the three-dimensional motion, characterized by both the position and orientation of the UAV, we used the transformation matrix

$$\begin{bmatrix} \boldsymbol{p_1}' \\ \boldsymbol{p_2}' \\ \boldsymbol{p_3}' \end{bmatrix} = \begin{bmatrix} c\psi c\theta - s\phi s\phi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{p_1} \\ \boldsymbol{p_2} \\ \boldsymbol{p_3} \end{bmatrix}. \tag{3.1}$$

3.3 Representation of angular speeds

In terms of the speed of rotation of a coordinate system we have

$$\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{q} \\ \boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\phi}} \\ \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{\psi}} \end{bmatrix}, \tag{3.2}$$

where p, q, r are angular speeds in the coordinate system linked to the device.

3.4 Quadcopter modeling using the Newton-Euler equations

The non-linear quadcopter model is

$$\begin{cases} \ddot{\boldsymbol{x}} = & \frac{1}{\mathrm{m}} \sum_{i=1}^{4} \boldsymbol{F}_i (\cos\psi \sin\theta + \cos\theta \sin\phi \sin\psi) \\ \ddot{\boldsymbol{y}} = & \frac{1}{\mathrm{m}} \sum_{i=1}^{4} \boldsymbol{F}_i (\sin\psi \sin\theta - \cos\psi \cos\theta \sin\phi) \\ \ddot{\boldsymbol{z}} = & \frac{1}{\mathrm{m}} \sum_{i=1}^{4} \boldsymbol{F}_i (\cos\phi \cos\theta) - \boldsymbol{g} \\ \dot{\boldsymbol{p}} = & \frac{\ell \mathrm{k_F} (\boldsymbol{\omega_2}^2 - \boldsymbol{\omega_4}^2)}{\mathrm{I_{xx}}} + \frac{\mathrm{I_{yy}} - \mathrm{I_{zz}}}{\mathrm{I_{xx}}} (\dot{\boldsymbol{\psi}} + \theta \sin\phi)(\dot{\boldsymbol{\phi}} \sin\psi + \theta \cos\phi \cos\psi) \\ \dot{\boldsymbol{q}} = & \frac{\ell \mathrm{k_F} (\boldsymbol{\omega_3}^2 - \boldsymbol{\omega_1}^2)}{\mathrm{I_{yy}}} + \frac{\mathrm{I_{zz}} - \mathrm{I_{xx}}}{\mathrm{I_{yy}}} (\dot{\boldsymbol{\psi}} + \theta \sin\phi)(\dot{\boldsymbol{\phi}} \cos\phi - \theta \cos\phi \sin\psi) \\ \dot{\boldsymbol{r}} = & \frac{\mathrm{k_M} (\boldsymbol{\omega_1}^2 - \boldsymbol{\omega_2}^2 + \boldsymbol{\omega_3}^2 - \boldsymbol{\omega_4}^2)}{\mathrm{I_{zz}}} + \frac{\mathrm{I_{xx}} - \mathrm{I_{yy}}}{\mathrm{I_{zz}}} (\dot{\boldsymbol{\phi}} \sin\psi + \theta \cos\phi \cos\psi)(\dot{\boldsymbol{\phi}} \cos\psi - \theta \cos\phi \sin\psi) \end{cases} \tag{3.3}$$

3.5 Conclusions

The chapter aimed to achieve the **OB2** objective of modeling the quadcopter, as a member of the flight formation. The role of the model obtained is to help design the automatic control and control system that will equip the aircraft, allowing a description of its evolution in three-dimensional space, both in terms of translation movement and in terms of rotational motion.

Chapter 4

Quadcopter Control

The control methodology used in this research work is highlighted in the block schema in fig. 4.1, in which the blue dotted inner loop plays a role in attitude control, and the green dotted outer loop plays a role in position control. Command $\boldsymbol{u_F}$ is

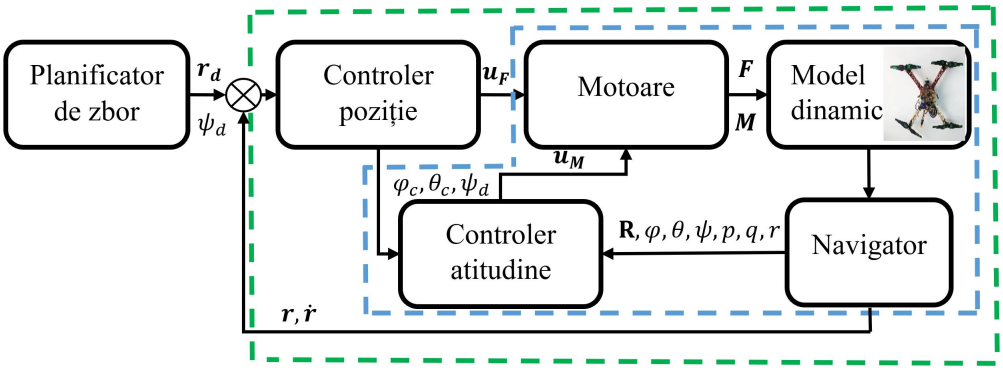


Figure 4.1: Block scheme of attitude and position control

$$\boldsymbol{u_F} = \mathrm{m}\boldsymbol{g} + \mathrm{m}\big(\ddot{\boldsymbol{z}}_{\boldsymbol{d}} + \mathrm{K_{dz}}(\dot{\boldsymbol{z}}_{\boldsymbol{d}} - \dot{\boldsymbol{z}}) + \mathrm{K_{pz}}(\boldsymbol{z_d} - \boldsymbol{z})\big), \tag{4.1}$$

and $\boldsymbol{u_M}$

$$\boldsymbol{u_M} = \begin{bmatrix} \boldsymbol{u_{Mx}} \\ \boldsymbol{u_{My}} \\ \boldsymbol{u_{Mz}} \end{bmatrix} = \begin{bmatrix} \mathrm{I_{xx}}\Big(\ddot{\boldsymbol{\phi}}_{\boldsymbol{c}} + \mathrm{K_{d\phi}}(\dot{\boldsymbol{\phi}}_{\boldsymbol{c}} - \dot{\boldsymbol{\phi}}) + \mathrm{K_{p\phi}}(\phi_c - \phi)\Big) \\ \mathrm{I_{yy}}\Big(\ddot{\boldsymbol{\theta}}_{\boldsymbol{c}} + \mathrm{K_{d\theta}}(\dot{\boldsymbol{\theta}}_{\boldsymbol{c}} - \dot{\boldsymbol{\theta}}) + \mathrm{K_{p\theta}}(\theta_c - \theta)\Big) \\ \mathrm{I_{zz}}\Big(\ddot{\boldsymbol{\psi}}_{\boldsymbol{c}} + \mathrm{K_{d\psi}}(\dot{\boldsymbol{\psi}}_{\boldsymbol{c}} - \dot{\boldsymbol{\psi}}) + \mathrm{K_{p\psi}}(\psi_c - \psi)\Big) \end{bmatrix}. \tag{4.2}$$

We will divide the problem of quadcopter control in the sub-problem of one-way movement (1D), then add the roll/tanging motion to the plane $\boldsymbol{xz}$ or $\boldsymbol{yz}$ (2D) in order to determine the set of coefficients $\mathrm{K_{px}}$, $\mathrm{K_{dx}}$ and $\mathrm{K_{p\theta}}$, $\mathrm{K_{d\theta}}$, or $\mathrm{K_{py}}$, $\mathrm{K_{dy}}$ and $\mathrm{K_{p\phi}}$, $\mathrm{K_{d\phi}}$. The 1D+2D composite movement will allow us to make the final adjustment of all the coefficients in the (4.1) and (4.2).

4.1 Altitude control

We aim to execute trajectory

$$\begin{bmatrix} \boldsymbol{r}(0) \\ \psi(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \boldsymbol{r_d}(T) \\ \psi_d(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2.2m \\ 1rad \end{bmatrix}, \tag{4.3}$$

which signifies take-off at the time $t = 0$, and then moving and rotating along and around the axis $\boldsymbol{z}$ until $t = T$, when $\boldsymbol{z_d} = 2.2m$ and $\psi_d = 1rad$. In this case, the control laws are

$$\boldsymbol{u_{1D}} = \begin{bmatrix} \boldsymbol{u_F} \\ \boldsymbol{u_{M,1D}} \end{bmatrix} = \begin{bmatrix} \mathrm{m}\boldsymbol{g} + \mathrm{m}\big(\ddot{\boldsymbol{z}}_{\boldsymbol{d}} + \mathrm{K_{dz,1D}}(\dot{\boldsymbol{z}}_{\boldsymbol{d}} - \dot{\boldsymbol{z}}) + \mathrm{K_{pz,1D}}(\boldsymbol{z} - \boldsymbol{z_d})\big) \\ 0 \\ 0 \\ \mathrm{I_{zz}}\big(\ddot{\boldsymbol{\psi}}_{\boldsymbol{d}} + \mathrm{K_{d\psi,1D}}(\dot{\boldsymbol{\psi}}_{\boldsymbol{d}} - \dot{\boldsymbol{\psi}}) + \mathrm{K_{p\psi,1D}}(\psi_d - \psi)\big) \end{bmatrix}, \tag{4.4}$$

in which $\mathrm{K_{pz}}, \mathrm{K_{p\psi}} \in \mathbb{R}$, $\mathrm{K_{pz}}, \mathrm{K_{p\psi}} > 0$ are proportional coefficients, $\mathrm{K_{dz}}$, $\mathrm{K_{d\psi}} \in \mathbb{R}$, $\mathrm{K_{dz}}$, $\mathrm{K_{d\psi}} > 0$ are derivative coefficients.

The technical specifications of the quadcopter that were used in the simulations to assess the performance of control laws in each of the 1D, 2D and 3D scenarios are highlighted in tab. 4.1. The results remained unchanged even though the value of the $\mathbf{I}_C$ inertia tensor was modified to use the three sets of values according to the research papers mentioned in tab. 4.1.

Table 4.1: Quadcopter's technical specifications

Parameter	Description	Value
$\ell_1, \ell_2, \ell_3, \ell_4$	Arm length	0.225m
m	Takeoff mass	1540g
$\mathbf{I}_C$	Inertia matrix	[42], [43], [44]
ω_{max}	Propeller maximum RPM	7200 RPM (Rotations per minute)
$\boldsymbol{F}_{max}$	Total thrust force	30.8N

4.2 Control of the quadcopter's 2D motion

In the case of plane movement, we choose to determine the values of the control law coefficients for the $\mathrm{K_{pphi}}, \mathrm{K_{dphi}}$. The effect of this movement will lead to a translational movement along the $\boldsymbol{y}$ axis. Add movement to the $\boldsymbol{xy}$ plane to the trajectory described

by (4.3), implies following the trajectory

$$\begin{bmatrix} \boldsymbol{r(0)} \\ \psi(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \boldsymbol{r_d(T)} \\ \psi_d(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 1m \\ 2.2m \\ 0 \end{bmatrix} \tag{4.5}$$

to which the 1m length was added along the $\boldsymbol{y}$ axis. It was necessary to remove the yaw motion in order not to influence the translation movement being followed, and in this respect we modified the `linie(t)` function to fix $\psi_d = \dot{\psi}_d = \ddot{\psi}_d$ =0. Control laws for this scenario are

$$\boldsymbol{u_{2D}} = \begin{bmatrix} \boldsymbol{u_F} \\ \boldsymbol{u_{M,2D}} \end{bmatrix} = \begin{bmatrix} \mathrm{m}\boldsymbol{g} + \mathrm{m}\big(\ddot{\boldsymbol{z}}_{\boldsymbol{d}} + \mathrm{K_{dz,1D}}(\dot{\boldsymbol{z}}_{\boldsymbol{d}} - \dot{\boldsymbol{z}}) + \mathrm{K_{pz,1D}}(\boldsymbol{z} - \boldsymbol{z_d})\big) \\ \mathrm{I_{xx}}\big(\mathrm{K_{d\phi}}(\dot{\phi}_c - \dot{\phi}) + \mathrm{K_{p\phi}}(\phi_c - \phi)\big) \\ 0 \\ \mathrm{I_{zz}}\big(\ddot{\psi}_d + \mathrm{K_{d\psi,1D}}(\dot{\psi}_d - \dot{\psi}) + \mathrm{K_{p\psi,1D}}(\psi_d - \psi)\big) \end{bmatrix}. \tag{4.6}$$

4.3 3D motion control

The tracked flight path has been completed so that

$$\begin{bmatrix} \boldsymbol{r(0)} \\ \psi(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \boldsymbol{r_d(T)} \\ \psi_d(T) \end{bmatrix} = \begin{bmatrix} 1m \\ 1m \\ 2.2m \\ 0 \end{bmatrix}, \tag{4.7}$$

and as can be seen, movement along the $\boldsymbol{x}$ axis has also been added.

$$\boldsymbol{u_{3D}} = \begin{bmatrix} \boldsymbol{u_F} \\ \boldsymbol{u_{M,3D}} \end{bmatrix} = \begin{bmatrix} \mathrm{m}\boldsymbol{g} + \mathrm{m}\big(\ddot{\boldsymbol{z}}_{\boldsymbol{d}} + \mathrm{K_{dz,1D}}(\dot{\boldsymbol{z}}_{\boldsymbol{d}} - \dot{\boldsymbol{z}}) + \mathrm{K_{pz,1D}}(\boldsymbol{z} - \boldsymbol{z_d})\big) \\ \mathrm{I_{xx}}\big(\ddot{\phi}_c + \mathrm{K_{d\phi}}(\dot{\phi}_c - \dot{\phi}) + \mathrm{K_{p\phi}}(\phi_c - \phi)\big) \\ \mathrm{I_{yy}}\big(\ddot{\theta}_c + \mathrm{K_{d\theta}}(\dot{\theta}_c - \dot{\theta}) + \mathrm{K_{p\theta}}(\theta_c - \theta)\big) \\ \mathrm{I_{zz}}\big(\ddot{\psi}_d + \mathrm{K_{d\psi,1D}}(\dot{\psi}_d - \dot{\psi}) + \mathrm{K_{p\psi,1D}}(\psi_d - \psi)\big) \end{bmatrix} \tag{4.8}$$

Results highlighted using graphs in fig. 4.3, fig. 4.4 as well as trajectory tracking (illustrated in fig. 4.2) with an accuracy of 0.37% confirm how the issue of quadcopter control was addressed and resolved. Finally, the expression according to the yaw rotation (4.3 was added to the model described by (4.7), which resulted in a decrease in the accuracy with which the flight path was tracked by approximately 0.001%. Therefore, the performance of the control law in (4.8) is maintained.

Algorithm 2 PD-type controller for altitude and roll and pitch control

function CONTROLER3D(*stare_curentă, stare_dorită, parametri_quad*)

$(\mathrm{K_{pz}}, \mathrm{K_{dz}}) \leftarrow (1.4, 0.75)$;
$(\mathrm{K_{p\psi}}, \mathrm{K_{d\psi}}) \leftarrow (4, 0)$;
$(\mathrm{K_{p\phi}}, \mathrm{K_{d\phi}}) \leftarrow (200, 35)$;
$(\mathrm{K_{py}}, \mathrm{K_{dy}}) \leftarrow (17, 8)$;
$(\mathrm{K_{p\theta}}, \mathrm{K_{d\theta}}) \leftarrow (200, 35)$;
$(\mathrm{K_{px}}, \mathrm{K_{dx}}) \leftarrow (17, 8)$;

$$u_F \leftarrow parametri\_quad.m \cdot parametri\_quad.g + parametri\_quad.m\big(\ddot{stare\_dorită}.z + \mathrm{K_{dz}}(\dot{stare\_dorită}.z - \dot{stare\_curentă}.z) + \mathrm{K_{pz}}(stare\_dorită.z - stare\_curentă.z)\big);$$

$$\phi_c \leftarrow 1/parametri\_quad.g\Big(\big((\ddot{stare\_dorită}.x + \mathrm{K_{dx}}(\dot{stare\_dorită}.x - \dot{stare\_curentă}.x) + \mathrm{K_{px}}(stare\_dorită.x - stare\_curentă.x)\big)\sin(stare\_dorită.\psi)\Big) - \Big((\ddot{stare\_dorită}.y + \mathrm{K_{dy}}(\dot{stare\_dorită}.y - \dot{stare\_curentă}.y) + \mathrm{K_{py}}(stare\_dorită.y - stare\_curentă.y)\big)\cos(stare\_dorită.\psi)\Big)\Big)$$

$$\theta_c \leftarrow 1/parametri\_quad.g\Big(\big((\ddot{stare\_dorită}.x + \mathrm{K_{dx}}(\dot{stare\_dorită}.x - \dot{stare\_curentă}.x) + \mathrm{K_{px}}(stare\_dorită.x - stare\_curentă.x)\big)\cos(stare\_dorită.\psi)\Big) + \Big((\ddot{stare\_dorită}.y + \mathrm{K_{dy}}(\dot{stare\_dorită}.y - \dot{stare\_curentă}.y) + \mathrm{K_{py}}(stare\_dorită.y - stare\_curentă.y)\big)\sin(stare\_dorită.\psi)\Big)\Big)$$

$(\dot{\phi}_c, \ddot{\phi}_c) \leftarrow (0, 0)$;
$(\dot{\theta}_c, \ddot{\theta}_c) \leftarrow (0, 0)$;

$$\boldsymbol{u_{Mx,3D}} \leftarrow \mathrm{K_{d\phi}}(\dot{\phi}_c - \dot{stare\_curentă}.\phi) + \mathrm{K_{p\phi}}(\phi_c - stare\_curentă.\phi)$$
$$\boldsymbol{u_{My,3D}} \leftarrow \mathrm{K_{d\theta}}(\dot{\theta}_c - \dot{stare\_curentă}.\theta) + \mathrm{K_{p\theta}}(\theta_c - stare\_curentă.\theta)$$
$$\boldsymbol{u_{Mz,3D}} \leftarrow parametri\_quad.\mathrm{I_{zz}}\big(\ddot{stare\_dorită}.\psi + \mathrm{K_{d\psi,1D}}(\dot{stare\_dorită}.\psi - \dot{stare\_curentă}.\psi) + \mathrm{K_{p\psi,1D}}(stare\_dorită.\psi - stare\_curentă.\psi)\big);$$

$\boldsymbol{u_{M,3D}} \leftarrow [\boldsymbol{u_{Mx,3D}}, \boldsymbol{u_{My,3D}}, \boldsymbol{u_{Mz,3D}}]^T$;

returnează $\boldsymbol{u_F}, \boldsymbol{u_{M,3D}}$;

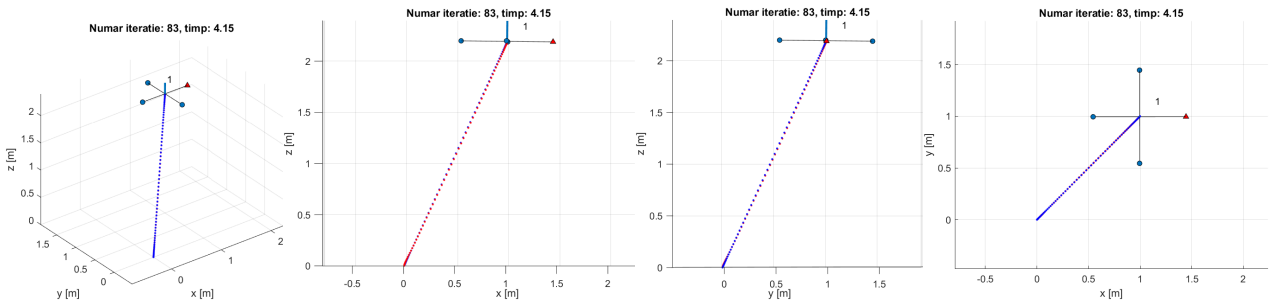


Figure 4.2: Tracking a path in the shape of a right segment represented in 3D

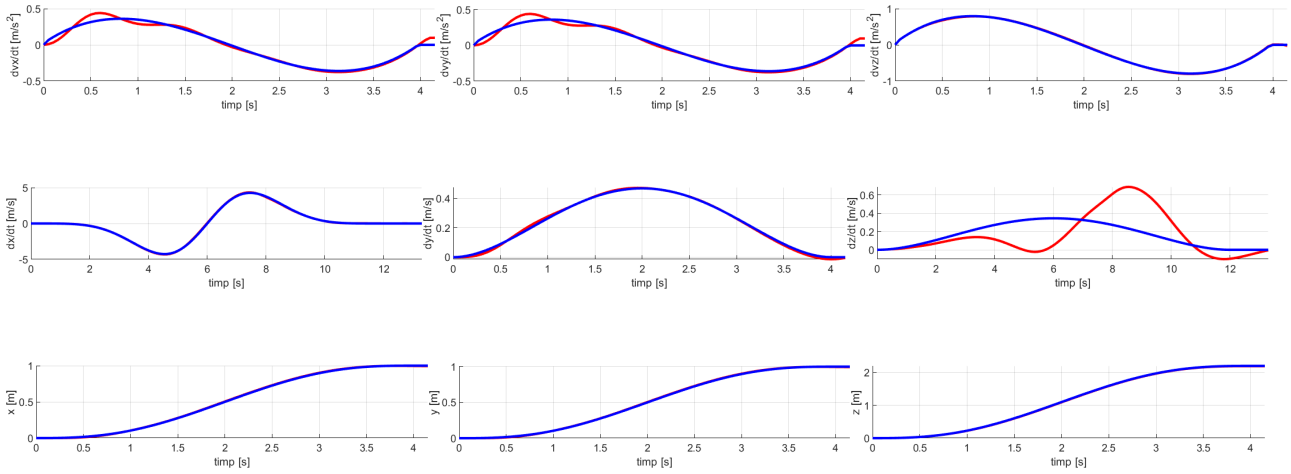


Figure 4.3: Evolution over time of the position, speed and acceleration of translation in the case of 3D flight

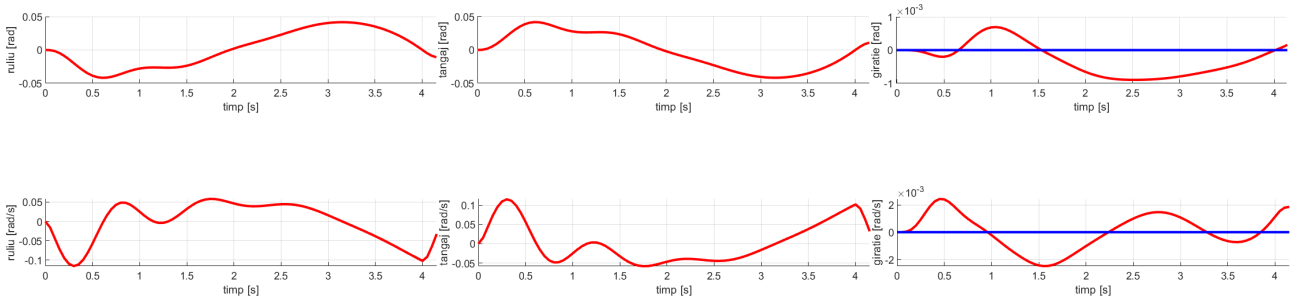


Figure 4.4: Evolution over time of angular position and speed of gyration in the case of 3D flight

The set with the final values of proportional and derivative coefficients is highlighted in tab. 4.2.

Table 4.2: PD control law coefficients, numerically determined

	K_{px}	K_{dx}	K_{py}	K_{dy}	K_{pz}	K_{dz}	$K_{p\phi}$	$K_{d\psi}$	$K_{p\theta}$	$K_{d\theta}$	$K_{p\psi}$	$K_{d\psi}$
Values "linie(t)"	17	8	17	8	1.4	0.75	200	35	200	35	4	0.1
Values "spirală(t)"	17	8	17	8	**90**	**11**	200	35	200	35	310	15

4.4 Conclusions

The chapter aimed to develop an automatic control system to equip each of the formation members involved in flight path planning studies. The elements presented during the chapter led to the achievement of the objectives **OB3** and **OB4** proposed for the achievement of the doctoral thesis. The numerical results obtained encouraged the practical realization of the flight controller for experimental testing of the required trajectories.

Chapter 5

Design, Realization and Experimental Testing of the Quadcopter Autopilot

Information related to: the components used; the design and manufacturing mechanisms of the electronic module (development stages related to versions A, B, C and D); software implementation of main flight modes (attitude control, altitude control, fixed-point flight) and an optional/safety mode ('Return to Home'); experimental testing, processing and analysis of data purchased to assess the performance of the flight modes implemented.

5.1 Main components

The components used to carry out the practical part of this doctoral thesis are as follows:

1. Quadcopter frame
    - Frame: Dji F450
    - Brushless motor 1000kV: A2212
    - Electronic speed controller, 30A
    - Propeller, 10 inch Dji
    - Three cell Li-Po battery, 3000mAh, 15C: Turnigy Graphene
2. Automatic control system
    - Microcontroller: STM32F103C8T6
    - Digital inertial unit - accelerometru și giroscop: MPU6050
    - Digital magnetometer: HMC5883L
    - Gobal positioning module: NEO-M8
    - Digital barometer: MS5611
    - Radio receiver: FS-iA6B
    - Telemetry radio module: APC220

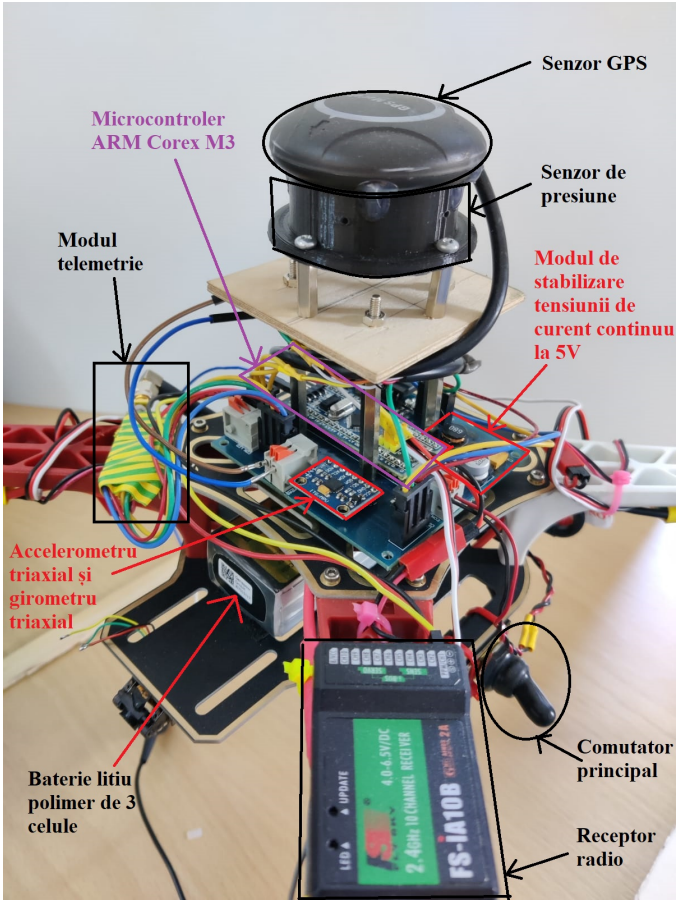


Figure 5.1: Close-up view of on-board electronic components

5.2 Designing, building and programming the automatic control system

The final version of the autopilot is illustrated in fig. 5.2.

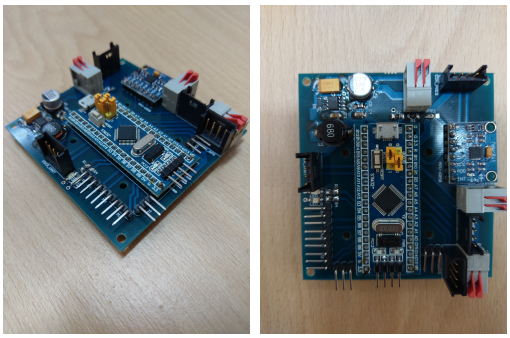


Figure 5.2: D version of autopilot for quadcopter

Software deployment of the following flight modes has been achieved:

1. Attitude control;
2. Altitude control;
3. Fixed-point control;
4. Return to the first GPS point recorded at startup or 'Return To Home'

The software implementation of control algorithms was carried out on a hardware architecture represented by a 32-bit microcontroller, of the Advanced Reduced Instruction Set Computer (RISC) Machine (ARM)-Cortex M3. I used as a programming environment µVision® IDE, and the software was written in C/C++. The source files corresponding to the control algorithms were obtained using the Matlab package, Embedded Coder®.

5.3 Analysis of experimental data

The achievement of the trajectories obtained for each simulated case in Matlab required the development of the flight mission programming software fig. 5.3. Using C#, we developed a graphical interface in VisualStudio. This can be used for flight mission planning by manually selecting GPS points on the map. The analysis of the acquired data was carried

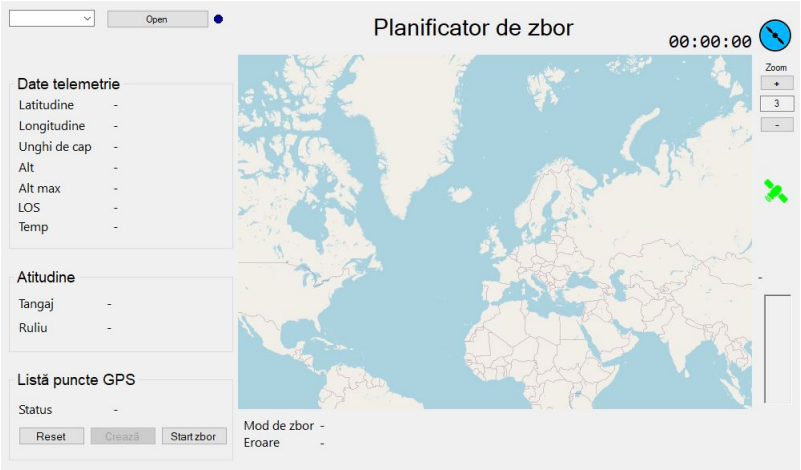


Figure 5.3: Screenshot of the graphical interface

out using Matlab, a program in which we generated fig. 5.4. The recorded parameters were transmitted to the ground via the radio module APC220 transceiver, during the course of the flight.

In the next step related to in-flight testing, we programmed a fixed-point flight to test whether the latitude and longitude deviations specified in the global positioning sensor catalog sheet used are compliant. Thus, in fig. 5.5 is highlighted that the fixed-point flight lasted about 33 seconds from the time of activation of the flight mode, at which point the values in the tab. 5.1. The purpose of this flight mode is to keep the quadcopter in space at

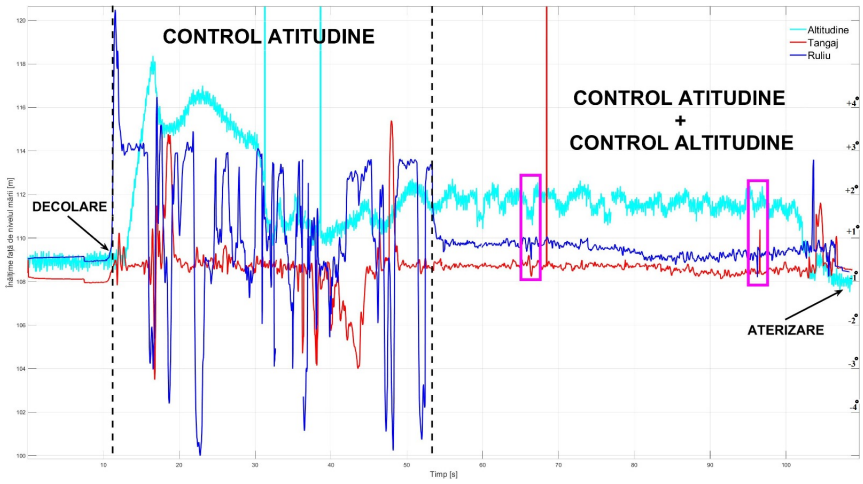


Figure 5.4: Evolution over time of the roll and pitch angles as well as the flight altitude

Figure 5.5: The position of the quadcopter, represented using the GoogleEarth app, in the case of the fixed-point flight scenario

Table 5.1: List of GPS coordinates and absolute flight altitude used in GPSfix flight mode

Flight type (abbreviation)	Figure	Latitude [°]	Longitude [°]	Flight altitude [m]
GPSfix	fig. 5.5	44.418163	26.086555	3

the 3D position specified by latitude, longitude and flight altitude relative to sea level. It is important to note that the maximum absolute deviation in both latitude and longitude does not exceed the maximum allowable value recorded in the catalogue sheet of the GPS sensor used.

5.3.1 Flight results analysis after tracking a 2D trajectory

The following are the results of the autopilot performance assessment, conducting a comparative analysis between the results obtained experimentally and those obtained numerically on the following flight scenarios:

- Flight by trajectory in fig. 5.6, imposed by four GPS points
- Flight by trajectory in fig. 5.8, imposed by five GPS points
- Flight by trajectory in fig. 5.10, imposed by six GPS points

The three scenarios gradually increased in complexity, with the vehicle being forced to perform repeated acceleration and deceleration, sometimes until a fixed-point flight is reached, in order to change the direction of flight. These developments imposed by choosing the coordinates of the GPS points that the aircraft was obliged to visit.

Scenariul 1: Flight plan imposed by four GPS points

(a) View from afar, from satellite (b) Close-up view from satellite

Figure 5.6: Four-point GPS trajectory

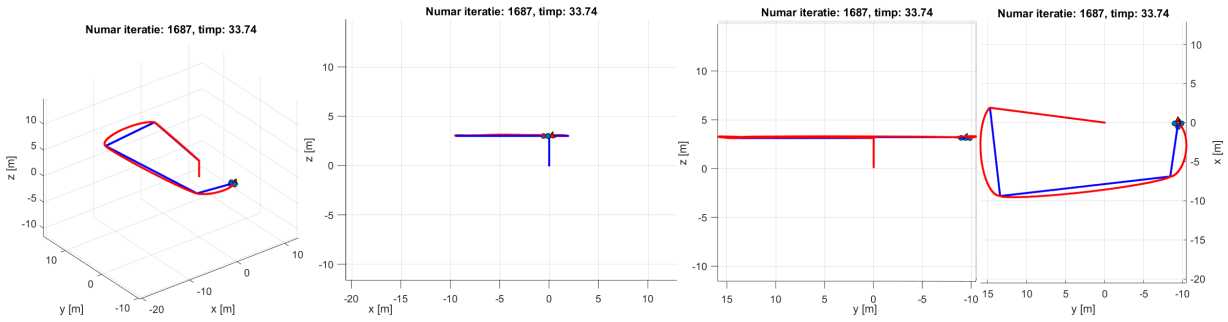


Figure 5.7: Tracking the 2D trajectory (imposed and coloured blue) in 5.6b

Scenariul 2: Flight plan imposed by five GPS points

As in the previous scenario, the trajectory in the 5.8a was made using the app Google Earth. In essence, the trajectory of the five GPS points is similar to that of the fig. 5.6, with the indication that another point has been added and the absolute flight altitude has been set at 4m.

Scenario 3: Flight plan imposed by six GPS points

The purpose of this flight scenario was to evaluate the drive behavior on a different shape than the first two flight missions to test the effectiveness with which the quadcopter

(a) View from afar, from satellite (b) Close-up view from satellite

Figure 5.8: The trajectory of the quadcopter consisting of five GPS points

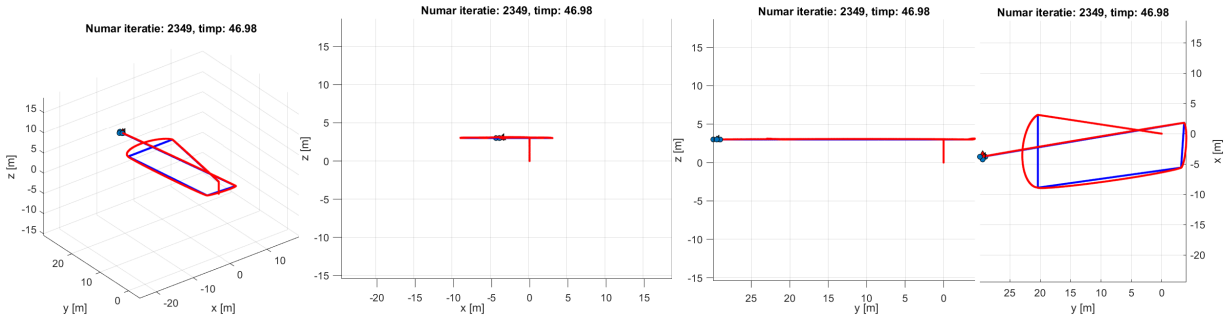
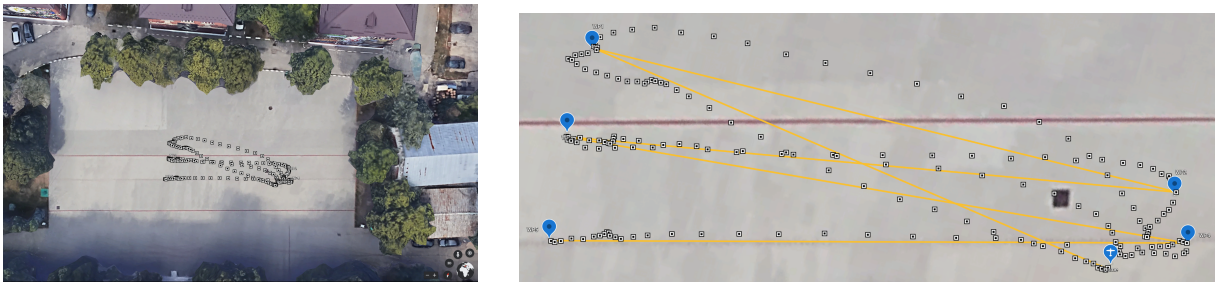


Figure 5.9: Tracking a 2D trajectory (imposed and colored blue), consisting of five coordinates



(a) View from afar, from satellite (b) Close-up view from satellite

Figure 5.10: The trajectory of the quadcopter consisting of six GPS points

manages to accelerate and decelerate, with the flight direction changing to nearly 180° at each GPS point of interest.

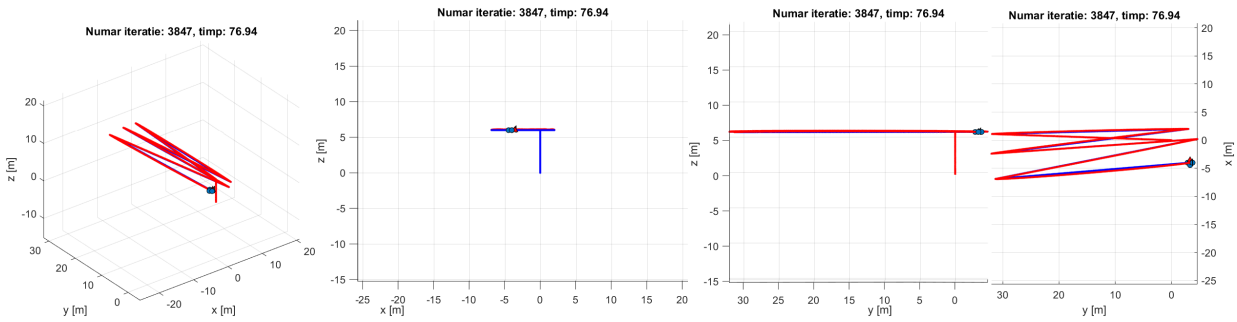


Figure 5.11: Tracking a 2D trajectory (imposed and colored blue), consisting of six coordinates

5.3.2 Flight results analysis after tracking a 3D trajectory

In addition to the previously tested situation, for this test we implemented the concept of optional/safe flight mode 'Return To Home', which leads to the execution of a 3D trajectory as follows:

1. Compare the flight altitude with the value set at +20m from the altitude (from sea level) recorded at take-off and the first step is to make the movement along the axis $\boldsymbol{z}$ to reach the desired altitude;
2. Compare the current GPS position with that recorded at take-off and determine the difference in latitude and longitude, after which the movement is controlled in the plane $\boldsymbol{xy}$ to turn the quadcopter;
3. Once the quadcopter has reached the GPS point recorded at the start, the landing is ordered, thus gradually decreasing the position corresponding to the axis $\boldsymbol{z}$.

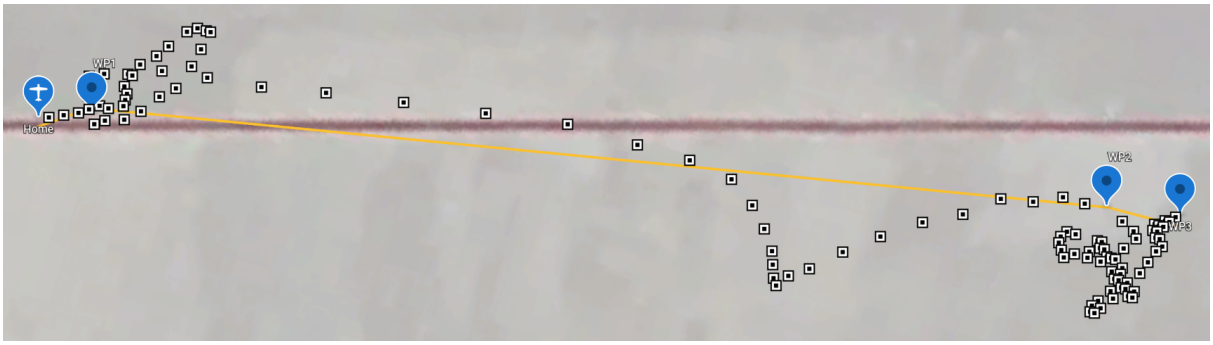


Figure 5.12: The trajectory of the quadcopter consisting of three GPS points

This flight mode (optional) is intended to return the quadcopter to the GPS point recorded at take-off. This measure was taken for safety reasons, in particular in cases where the radio link between the flight path programming application (graphic interface)

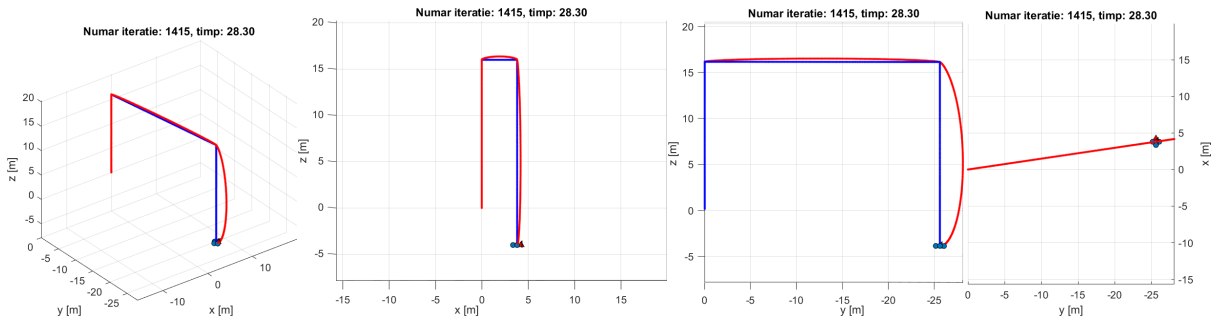


Figure 5.13: Tracking the 3D trajectory (imposed and coloured blue) in fig. 5.12

running on the ground and the quadcopter is lost, and in cases where control of the aircraft is lost by the pilot. Flight mode 'Return To Home' or 'Return To Land' was activated after performing flight scenarios 5.6b and 5.8b, which can be easily observed by analysing images.

5.4 Conclusions

The results presented in this chapter have led to the achievement of the **OB5** and **OB6** of the doctoral thesis, namely: experimental autopilot achievement for the multirotor UAV and experimental testing of the autopilot developed. The results set out in this chapter confirmed that the quadcopter is ready for use to test the flight path planning algorithms described in the cap. 6 and cap. 7. The performance achieved is encouraging and provides both the hardware and software support provided in the structure of the RETPINC system (from fig. 1.2), which assumes the existence of the control component that will run together with the flight path planning component.

Chapter 6

Flight Path Planning with Energy Consumption Minimization in the Absence of Obstacles

6.1 Preparatory elements for calculating flight path using Euler-Lagrange equations

Summarize those envisaged to solve the flight path planning problem as follows:

(i) Fixing the starting, stop, intermediate spatial coordinates (GPS points) via the $\mathbf{r_d}$;
(ii) Setting guidelines for GPS points established through ψ_d;
(iii) Minimize variation of order entries $\boldsymbol{u_F}, \boldsymbol{u_M}$;
(iv) Establishing the system order *(n)* and the imposition of *(n-1)* restrictions for formulating and resolving the cost function of the flight path planning problem that minimizes total electricity consumption.

Solving this problem involves the use of tools from the variational calculation, to which essential contributions were made by Joseph Louis Lagrange. The purpose of using the variance calculation is to find the optimal function $x^*(t)$ for the following objectives:

- Minimizing speed, $n = 1$;
- Minimizing accelerations, $n = 2$;
- Minimizing acceleration variation, $n = 3$;
- Minimizing the variation of acceleration variation, $n = 4$; etc.

6.2 Running a mission with a planned flight trajectory, while minimizing the acceleration of the translation/traction force acting on the structure of the quadcopter

One of the main objectives of this doctoral thesis is the planning of the flight path, aiming to reduce energy consumption. Intuitively, to achieve this is to avoid sudden maneuvers. This can be translated into the minimization of the translation accelerations of the quadcopter, which essentially involves minimizing the forces acting on the structure of the UAV. To minimize the total acceleration along the flight path, consider the example of the flight scenario in the figure 4.2a. Solving the problem of planning the flight path in three-dimensional space, assumes, in addition to the three coordinates $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$ to identify the optimal for the angle of the head, ψ. Therefore

$$\boldsymbol{x^*(t)}, \boldsymbol{y^*(t)}, \boldsymbol{z^*(t)}, \psi^*(t) = \arg\min \boldsymbol{x(t)} \int_0^T \mathcal{L}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\boldsymbol{z}}, \dot{\psi}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \psi, t)\, dt$$

$$\int_0^T \mathcal{L}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\boldsymbol{z}}, \dot{\psi}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \psi, t)\, dt = \int_0^T \left(\ddot{\boldsymbol{x}}^2 + \ddot{\boldsymbol{y}}^2 + \ddot{\boldsymbol{z}}^2 + \ddot{\psi}^2 \right) dt \tag{6.1}$$

$$\int_0^T \mathcal{L}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\boldsymbol{z}}, \dot{\psi}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \psi, t)\, dt = \int_0^T \ddot{\boldsymbol{x}}^2\, dt + \int_0^T \ddot{\boldsymbol{y}}^2\, dt + \int_0^T \ddot{\boldsymbol{z}}^2\, dt + \int_0^T \ddot{\psi}^2\, dt.$$

Solving the trajectory planning problem was done in Matlab , based on the algorithm (3). This algorithm is essentially a function or method, called by the algorithm (4), which, based on the current flight mission time t and the GPS points describing the trajectory, will provide the position, speed and acceleration of the translation (in the inertia coordinate system) that the quadcopter must have at the next iteration of the control algorithm.

Algorithm 3 3D trajectory generation with translation acceleration optimization

function GENERARETRAIECTORIE(*timp, puncteGPS*)

if *status* = *mod_initializare* **then**
$\left[\dot{\mathbf{r}}_{\mathbf{d,3D}}, \ddot{\mathbf{r}}_{\mathbf{d,3D}}, \psi_{d,3D}, \dot{\psi}_{d,3D}\right] \leftarrow [0, 0, 0, 0]$
$c_x \leftarrow$ CalculeazaCoef(*puncteGPS_x*)
$c_y \leftarrow$ CalculeazaCoef(*puncteGPS_y*)
$c_z \leftarrow$ CalculeazaCoef(*puncteGPS_z*)
$\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}} \leftarrow$ valoare_stabilita
$d \leftarrow$ euclid (puncteGPS, $\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}}$)
timp_traiectorie $\leftarrow$ suma(d)
status $\leftarrow$ mod_continuu

else

if timp > *timp_traiectorie*[final] **then**
timp = *timp_traiectorie*[final]
index_timp $\leftarrow$ gaseste_index_timp(timp, timp_traiectorie)

if timp = 0 **then**
index_timp $\leftarrow$ 1
factor $\leftarrow$ (*timp_traiectorie*[index_timp])/ d[index_timp]
$t_0 \leftarrow$ CalculeazaValoriTimp(ordin_polinom_0, #derivata_0, factor)
$t_1 \leftarrow$ CalculeazaValoriTimp(ordin_polinom_1, #derivata_1, factor)
$t_2 \leftarrow$ CalculeazaValoriTimp(ordin_polinom_2, #derivata_2, factor)
$idx \leftarrow [4(\text{index\_timp} - 1) + 1 : 4\text{index\_timp}]$
$\mathbf{r}_{\mathbf{d,3D}} \leftarrow \left[c_x[idx]t_0;\ c_y[idx]t_0;\ c_z[idx]t_0\right]$
$\dot{\mathbf{r}}_{\mathbf{d,3D}} \leftarrow \left[c_x[idx]t_1/d[index\_timp];\ c_y[idx]t_1/d[index\_timp];\ c_z[idx]t_1/d[index\_timp]\right]$
$\ddot{\mathbf{r}}_{\mathbf{d,3D}} \leftarrow \left[c_x[idx]t_2/d[index\_timp]^2;\ c_y[idx]t_2/d[index\_timp]^2;\ c_z[idx]t_2/d[index\_timp]^2\right]$
$\psi_{d,3D} \leftarrow 0$
$\dot{\psi}_{d,3D} \leftarrow 0$
return stare_dorită:$\left[\mathbf{r}_{\mathbf{d,3D}},\ \dot{\mathbf{r}}_{\mathbf{d,3D}},\ \ddot{\mathbf{r}}_{\mathbf{d,3D}},\ \psi_{d,3D},\ \dot{\psi}_{d,3D}\right]$.

Algorithm 4 3D trajectory simulation with translation acceleration optimization

function SIMULARE3D(*GenerareTraiectorie, Controler3D*)
inizializare_grafice();
timp_alocat_maxim ← 30s
inițial:
control_increment_timp ← 4ms
traiectorie_increment_timp *gets* 10control_increment_timp
numar_iterații ← timp_alocat_maxim / control_increment_timp
timp_curent ← 0
$[\mathbf{r_{d,3D_{START}}}, \dot{\mathbf{r}}_{\mathbf{d,3D_{START}}}, \ddot{\mathbf{r}}_{\mathbf{d,3D_{START}}}]$ ← GenerareTraiectorie(timp_curent)
$[\mathbf{r_{d,3D_{STOP}}}, \dot{\mathbf{r}}_{\mathbf{d,3D_{STOP}}}, \ddot{\mathbf{r}}_{\mathbf{d,3D_{STOP}}}]$ ← GenerareTraiectorie(∞)
abatere_r ← 0.05
abatere_$\dot{r}$ ← 0.01
loop:
if iterație ≤ numar_iterații **then**

if iterație =1 **then**
ReprezentareGrafică(*quad*,traiectorie_increment_timp)
else
[*timp,stări*] ← ModelQuad(
EcuațiiMișcare(Controler3D,GenerareTraiectorie,ParamQuad),
iterație,$\boldsymbol{r}, \dot{\boldsymbol{r}}, \ddot{\boldsymbol{r}}$)
ReprezentareGrafică(*quad*,traiectorie_increment_timp)
iterație ← iterație+1

if VerificareStop(
$\mathbf{r_{d,3D_{STOP}}}, \dot{\boldsymbol{r}}, \ddot{\boldsymbol{r}}$, timp_curent,
abatere_r, abatere_$\dot{\boldsymbol{r}}$, numar_iterații)=TRUE **then**
close
goto *loop*
close
ReprezentareGrafice($\boldsymbol{r}, \dot{\boldsymbol{r}}, \ddot{\boldsymbol{r}}$, timp).

6.3 Running a mission with a planned flight path, while minimizing the variation of the translation acceleration/traction force or the moments acting on the structure of the quadcopter

Related to this sub-chapter, we will solve the problem of flight path planning of the form

$$\begin{aligned}\boldsymbol{x}^*(\boldsymbol{t}), \boldsymbol{y}^*(\boldsymbol{t}), \boldsymbol{z}^*(\boldsymbol{t}) = &\underset{\boldsymbol{x}(\boldsymbol{t})}{\arg\min} \left(\int_{t_0}^{t_1} \dddot{\boldsymbol{x}}^2 \, dt + \cdots + \int_{t_3}^{t_4} \dddot{\boldsymbol{x}}^2 \, dt \right) + \\ &\underset{\boldsymbol{y}(\boldsymbol{t})}{\arg\min} \left(\int_{t_0}^{t_1} \dddot{\boldsymbol{y}}^2 \, dt + \cdots + \int_{t_3}^{t_4} \dddot{\boldsymbol{y}}^2 \, dt \right) + \\ &\underset{\boldsymbol{z}(\boldsymbol{t})}{\arg\min} \left(\int_{t_0}^{t_1} \dddot{\boldsymbol{z}}^2 \, dt + \cdots + \int_{t_3}^{t_4} \dddot{\boldsymbol{z}}^2 \, dt \right),\end{aligned} \tag{6.2}$$

6.4 Estimation of energy consumption when carrying out missions with planned flight trajectory, in different scenarios to minimize variables related to quadcopter dynamics

This sub-chapter aims to estimate the energy consumption of the quadcopter when carrying out different flight missions under the minimisation conditions studied in the previous sub-chapters (minimizing the acceleration of translation, i.e. minimizing its variation and timing). If energy consumption increases when the flight path is calculated by minimizing $\overset{(4)}{\boldsymbol{r}}$. This occurs because, in order to keep the quadcopter on the trajectory (i.e. the deviation in position, speed and acceleration is as low as possible), the value of the force vector $\boldsymbol{F}$ acting on the quadcopter increases. Thus, even if the energy requirement is the lowest in the case of minimisation $\ddot{\boldsymbol{r}}$ or $\dddot{\boldsymbol{r}}$, the accuracy with which the trajectory is tracked by the quadcopter is greater in the event of its minimisation $\overset{(4)}{\boldsymbol{r}}$.

Table 6.1: Estimate of energy consumption for flight scenarios in the fig. 5.12, fig. 5.6 and fig. 5.10

Limit	**Trajectory**	E_{WP3} [Wh]	E_{WP4} [Wh]	E_{WP6} [Wh]
$\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}} = 0.1m/s^2$	$\min \dot{\boldsymbol{r}}$	196.12	197.79	-
	$\min \ddot{\boldsymbol{r}}$	195.18	196.31	196.11
	$\min \dddot{\boldsymbol{r}}$	195	195.46	195.27
	$\min \overset{(4)}{\boldsymbol{r}}$	**194.88**	**195.23**	**195.14**
$\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}} = 0.5m/s^2$	$\min \dot{\boldsymbol{r}}$	-	-	-
	$\min \ddot{\boldsymbol{r}}$	195.96	197.23	196.42
	$\min \dddot{\boldsymbol{r}}$	195.79	198.97	196.35
	$\min \overset{(4)}{\boldsymbol{r}}$	**195.29**	**197.07**	**196.23**
$\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}} = 1m/s^2$	$\min \dot{\boldsymbol{r}}$	-	-	-
	$\min \ddot{\boldsymbol{r}}$	198.92	**203.24**	199.8
	$\min \dddot{\boldsymbol{r}}$	198.96	206.88	**199.35**
	$\min \overset{(4)}{\boldsymbol{r}}$	**198.16**	-	200.47
$\ddot{\mathbf{r}}_{\mathbf{d,3D_{max}}} = 2m/s^2$	$\min \dot{\boldsymbol{r}}$	-	-	-
	$\min \ddot{\boldsymbol{r}}$	202.94	**222.55**	212.62
	$\min \dddot{\boldsymbol{r}}$	**202.24**	-	**212.59**
	$\min \overset{(4)}{\boldsymbol{r}}$	202.27	-	212.79

6.5 Conclusions

This chapter achieved the fulfillment of the objectives **OB7**, **OB8** and **OB9**, of the doctoral thesis. Based on the results achieved up to this point, the following chapter will present a study on the extension of the flight path planning problem by adding static (workspace objects) and dynamic obstacles represented by the other members of the flight group.

Chapter 7

Flight Path Planning with Energy Consumption Minimization in the Presence of Obstacles

This chapter aims to:

- Develop a method of planning the flight path with the minimisation of energy consumption for a quadcopter in the presence of obstacles;
- Validate by numerical simulations and demonstration of the performance of the flight path planning algorithm with minimizing energy consumption for a quadcopter in the presence of obstacles;
- Develop of a method of planning the flight path with the minimisation of energy consumption for two members of the flight group in the absence of obstacles;
- Validate by numerical simulations and demonstration of the performance of the flight path planning algorithm with the minimisation of energy consumption for two members of the flight group in the absence of obstacles;
- Develop a method of planning the flight path with the minimisation of energy consumption for two members of the flight group in the presence of obstacles;
- Validate by numerical simulations and demonstration of the performance of the flight path planning algorithm with the minimisation of energy consumption for two members of the flight group in the presence of obstacles.

All this coincides with the total achievement of the objectives **OB10−OB15**. The following sub-chapters deal successively with the challenges posed by:

- Flight path planning for a UAV avoiding static obstacles;
- Flight path planning for two UAVs with inter-member collision avoidance;
- Planning the flight path for two UAVs with inter-member and static obstacles collision avoidance.

7.1 Theoretical considerations on convex optimization

Starting from the general form of a written optimization problem of the

$$\begin{aligned} &\min \ f_0(x) \\ &\text{evaluată la } \ f_i(x) \leq \mathrm{b_i}, \ \ \mathrm{i} = 1, \ldots, \mathrm{m}, \end{aligned} \tag{7.1}$$

where $x = (x_1, \ldots, x_n)$ and represents the variable in relation to which optimisation is carried out, $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the objective function, $f_i : \mathbf{R}^n \to \mathbf{R}$ represent the constraints and $\mathrm{b}_1, \ldots, \mathrm{h}_\mathrm{m}$ are the constraints limits. General form of optimization problem (7.1) becomes a linear problem if the objective function as well as the functions describing the restrictions are linear, i.e. satisfy

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y), \tag{7.2}$$

$\forall x, y, \in \mathbf{R}^n$, iar $\alpha, \beta \in \mathbf{R}$.

The issue of flight path planning has so far been treated as a linear problem. Linear optimization issues are part of an important class of convex optimization issues [45]. Starting from the expression (7.2), a convex optimization problem requires that the objective or functional function as well as the restrictions be convex, which mathematically means:

$$\begin{aligned} f_i(\alpha x + \beta y) &\leq \alpha f_i(x) + \beta f_i(y) \\ \alpha + \beta &= 1 \\ \alpha \geq 0, \beta \geq 0 \quad \alpha, \beta &\in \mathbf{R}, \end{aligned} \tag{7.3}$$

$\forall x, y, \in \mathbf{R}^n$. It is observed from (7.2) and (7.3) that we can consider convex optimization, a generalization of the linear optimization problem.

7.2 Planning the trajectory avoiding static obstacles, for an UAV performing solitary flight

We approximate the dynamics of the UAVs band members using the dual integrator model, and according to [46], the general expression of the linear system of order II invariant over time for a discrete model can be approximated as

$$\begin{aligned} \boldsymbol{x_{t+\delta t}} &= \mathbf{A_d} \boldsymbol{x_t} + \mathbf{B_d} \boldsymbol{u_t} \\ \mathbf{x_0} &= \begin{bmatrix} \mathbf{r_0} \\ \dot{\mathbf{r}}_\mathbf{0} \end{bmatrix}, \end{aligned} \tag{7.4}$$

in which $\boldsymbol{u_t} = \ddot{\boldsymbol{r}}_{\boldsymbol{t}}$ column vector with translation acceleration components at time $t\,\forall t \in [0, \mathrm{T}]$. The initial condition is specified by $\mathbf{x_0}$. In extended form, (7.4) becomes

$$\begin{bmatrix} \boldsymbol{x_{t+\delta t}} \\ \boldsymbol{y_{t+\delta t}} \\ \dot{\boldsymbol{x}}_{\boldsymbol{t+\delta t}} \\ \dot{\boldsymbol{y}}_{\boldsymbol{t+\delta t}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x_t} \\ \boldsymbol{y_t} \\ \dot{\boldsymbol{x}}_{\boldsymbol{t}} \\ \dot{\boldsymbol{y}}_{\boldsymbol{t}} \end{bmatrix} + \begin{bmatrix} \frac{\delta t^2}{2} & 0 \\ 0 & \frac{\delta t^2}{2} \\ \delta t & 0 \\ 0 & \delta t \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}}_{\boldsymbol{t}} \\ \ddot{\boldsymbol{y}}_{\boldsymbol{t}} \end{bmatrix}. \tag{7.5}$$

Thus, in the context of low energy consumption, as demonstrated in the previous chapter, we aim that the magnitude of the forces (which will generate the translation accelerations) acting on the structure of the UAV must be as small or even minimal as possible. The solution of pursuing this objective is to minimize the cost function, written in discrete form for the 2D case

$$J = \sum_{t=0}^{T} (||\ddot{\boldsymbol{x}}_{\boldsymbol{t}}{}^2|| + ||\ddot{\boldsymbol{y}}_{\boldsymbol{t}}{}^2||). \tag{7.6}$$

We write the explicit form of (7.1) as

$$\begin{aligned} &\min_{\ddot{\boldsymbol{u}}} \sum_{t=0}^{T} ||\ddot{\boldsymbol{u}}_{\boldsymbol{t}}{}^2|| \\ &\mathbf{x_0} = \begin{bmatrix} \mathbf{r_0} & \dot{\mathbf{r}}_{\mathbf{0}} \end{bmatrix}^T \\ &\begin{bmatrix} \boldsymbol{r_{t+\delta t}} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t+\delta t}} \end{bmatrix} = \begin{bmatrix} \mathbf{I_2} & \delta t \mathbf{I_2} \\ \mathbf{O_2} & \mathbf{I_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{r_t} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t}} \end{bmatrix} + \begin{bmatrix} \frac{\delta t^2}{2} \mathbf{I_2} \\ \delta t \mathbf{I_2} \end{bmatrix} \ddot{\boldsymbol{u}}_{\boldsymbol{t}} \\ &\begin{cases} x_t - x_{l,min} \leq \mathrm{R} b_{t,1} - d_l \\ y_t - y_{l,min} \leq \mathrm{R} b_{t,2} - d_l \\ x_{l,max} - x_t \leq \mathrm{R} b_{t,3} - d_l \\ y_{l,max} - y_t \leq \mathrm{R} b_{t,4} - d_l \\ \sum_{o=1}^{4} b_{t,o} \leq 3 \end{cases} , \forall l = [1, \mathrm{L}] \\ &||\dot{\boldsymbol{r}}_{\boldsymbol{t}}|| \leq ||\dot{\mathbf{r}}_{\mathbf{max}}|| \\ &||\ddot{\boldsymbol{r}}_{\boldsymbol{t}}|| \leq ||\ddot{\mathbf{r}}_{\mathbf{max}}|| \\ &\mathbf{x_T} = \begin{bmatrix} \mathbf{r_T} & \dot{\mathbf{r}}_{\mathbf{T}} \end{bmatrix}^T, \end{aligned} \tag{7.7}$$

in which d_l is the safety distance d around obstacle l, and with L the number of obstacles. We aim minimization by satisfying the constraints of equality and inequality within the brace. Restrictions on inequality within the smaller brace are met at each δt iteration L times. We are customizing the problem described by (7.7), hereinafter referred to as OPTimization ACCeleration (OPTACC), in relation to the actual working conditions described by the scenario in the fig. 7.1:

- δt has been fixed one at a time at 1s, 0.5s, 0.2s, which is the time intervals most often used in the reception of GPS coordinates;

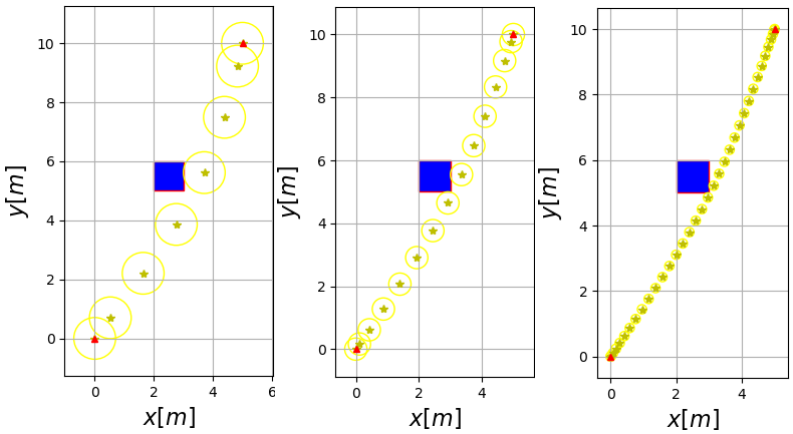
- T = $7s$, L = 1;
- Start point coordinates: $(x, y) = (0m, 0m)$; The UAV achieves a fixed point at a certain height z;
- Endpoint coordinates: $(x, y) = (5m, 10m)$. UAV keeps z constant altitude;
- Obstacle coordinates: $(x_{1,min}, y_{1,min}, x_{1,max}, y_{1,max}) = (2m, 5m, 3m, 6m)$;
- R = $1000000m$
- $||\dot{\mathbf{r}}_{\mathbf{max}}|| = 2m/s$, $||\ddot{\mathbf{r}}_{\mathbf{max}}|| = 2m/s^2$;
- d_{max} is obtained when $\gamma = \pi/4$.

Numerical resolution of the problem

$$
\begin{aligned}
&\min_{\ddot{\boldsymbol{u}}} \sum_{t=0}^{7s} ||{\ddot{\boldsymbol{u}}_{\boldsymbol{t}}}^2|| \\
&\mathbf{x_0} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \\
&\begin{bmatrix} \boldsymbol{r_{t+\delta t}} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t+\delta t}} \end{bmatrix} = \begin{bmatrix} \mathbf{I_2} & \delta t \mathbf{I_2} \\ \mathbf{O_2} & \mathbf{I_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{r_t} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t}} \end{bmatrix} + \begin{bmatrix} \frac{\delta t^2}{2} \mathbf{I_2} \\ \delta t \mathbf{I_2} \end{bmatrix} \ddot{\boldsymbol{u}}_{\boldsymbol{t}} \\
&\begin{cases} x_t - 2 \leq 1000000 b_{t,1} - \frac{\dot{\mathbf{r}}_{\mathbf{t,max}} \delta t}{2} \sin(\pi/4) \\ y_t - 5 \leq 1000000 b_{t,2} - \frac{\dot{\mathbf{r}}_{\mathbf{t,max}} \delta t}{2} \sin(\pi/4) \\ 3 - x_t \leq 1000000 b_{t,3} - \frac{\dot{\mathbf{r}}_{\mathbf{t,max}} \delta t}{2} \sin(\pi/4) \\ 6 - y_t \leq 1000000 b_{t,4} - \frac{\dot{\mathbf{r}}_{\mathbf{t,max}} \delta t}{2} \sin(\pi/4) \\ \sum_{o=1}^{4} b_{t,o} \leq 3 \end{cases} \\
&||\dot{\mathbf{r}}_{\mathbf{t,max}}|| \leq 2 \\
&||\ddot{\mathbf{r}}_{\mathbf{t,max}}|| \leq 2 \\
&\mathbf{x_T} = \begin{bmatrix} 5 & 10 & 0 & 0 \end{bmatrix}^T,
\end{aligned}
\tag{7.8}
$$

took place through the hardware and software resources of the tab. 2.1. We want to compare these results with a new flight path planning technique using the IRRT* method and the expression (7.7), specific to the MILP method. This idea stems from the fact that we want to 'inform' the MOSEK component on the existence of the IRRT* trajectory, in order to avoid searching for all solutions to the optimization problem in 2D space. Mathematically, we say we want to solve the optimization problem

$$
\begin{aligned}
&\min_{\ddot{\boldsymbol{u}}} \sum_{t=0}^{T} ||{\ddot{\boldsymbol{u}}_{\boldsymbol{t}}}^2|| \\
&\mathbf{x_0} = \begin{bmatrix} \mathbf{r_0} & \dot{\mathbf{r}}_{\mathbf{0}} \end{bmatrix}^T \\
&\begin{bmatrix} \boldsymbol{r_{t+\delta t}} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t+\delta t}} \end{bmatrix} = \begin{bmatrix} \mathbf{I_2} & \delta t \mathbf{I_2} \\ \mathbf{O_2} & \mathbf{I_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{r_t} \\ \dot{\boldsymbol{r}}_{\boldsymbol{t}} \end{bmatrix} + \begin{bmatrix} \frac{\delta t^2}{2} \mathbf{I_2} \\ \delta t \mathbf{I_2} \end{bmatrix} \ddot{\boldsymbol{u}}_{\boldsymbol{t}} \\
&||\boldsymbol{r_t} - \boldsymbol{r_{t,irrt}}|| \leq d_{l,irrt} - d_l, \, \forall l = [1, \mathrm{L}] \\
&||\dot{\boldsymbol{r}}_t|| \leq ||\dot{\mathbf{r}}_{\mathbf{max}}||
\end{aligned}
\tag{7.9}
$$

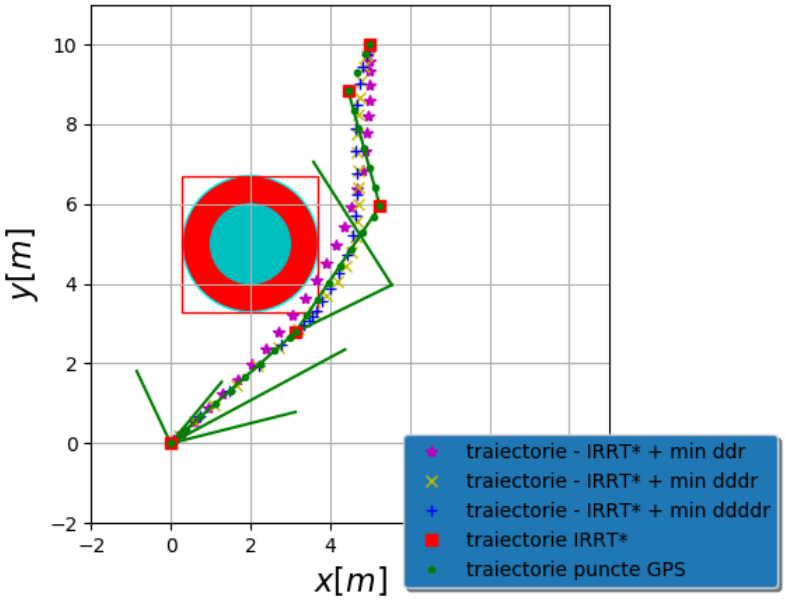


(a) $\delta t = 1s$ (b) $\delta t = 0.5s$ (c) $\delta t = 0.2s$

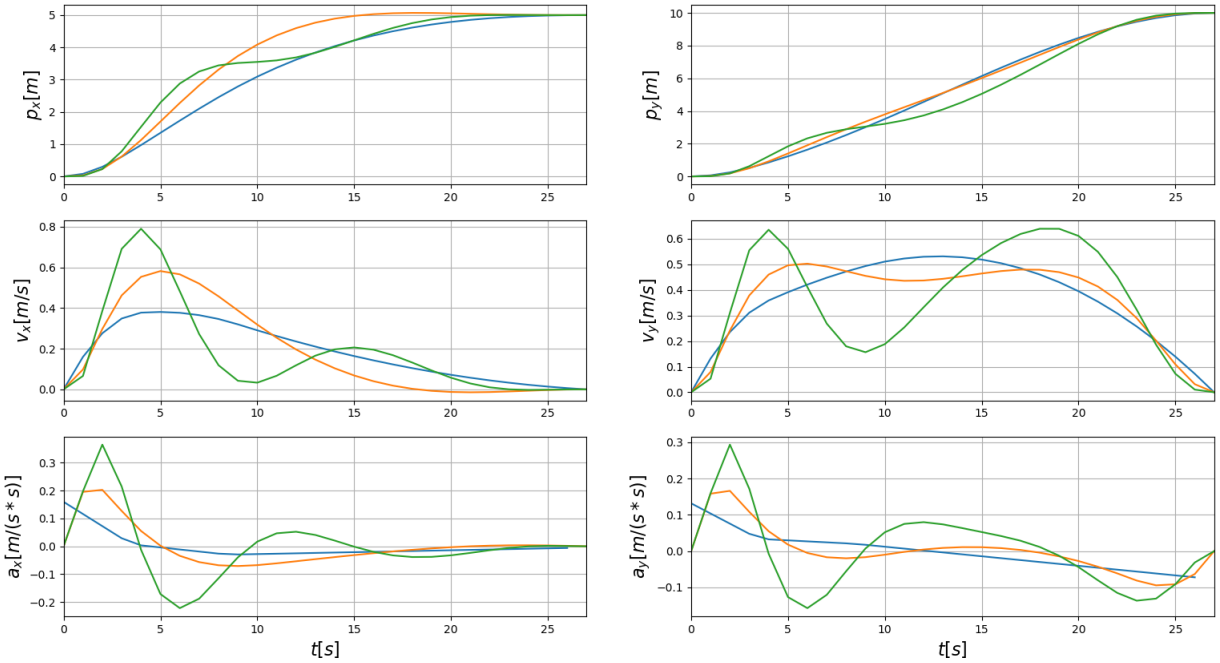
Figure 7.1: The 2D trajectory of the quadcopter avoiding an obstacle, obtained after applying OPTACC with the time step δt

$$||\ddot{\boldsymbol{r}}_t|| \leq ||\ddot{\mathbf{r}}_{\mathbf{max}}||$$
$$\mathbf{x_T} = \begin{bmatrix} \mathbf{r_T} & \dot{\mathbf{r}}_{\mathbf{T}} \end{bmatrix}^T,$$

hereinafter referred to as IRRT star OPTACC (OPTDACC), in which $d_{l,irrt}$ is the distance from the quadcopter, at which samples are generated within the IRRT* algorithm and has been fixed at $d_{l,irrt} = \mathbf{r_l} + s_l/2 + \dot{\mathbf{r}}_{\mathbf{max}}\delta t$ ($\mathbf{r_l}$ represents the position of the obstacle l and s length of its edge). In the previous chapter, we demonstrated that minimising $\dddot{\boldsymbol{r}}$ and $\overset{(4)}{\boldsymbol{r}}$ for the execution of a flight path described via GPS points, leads to a reduction in energy consumption. This motivation involved customizing the expression (7.7), process resulting in the Derivative OPTACC (OPTDACC), IRRT star Derivative OPTACC (IRRTOPTDACC), and respectively Double Derivative OPTACC (OPTDDACC), IRRT star Double Derivative OPTACC (IRRTOPTDDACC).

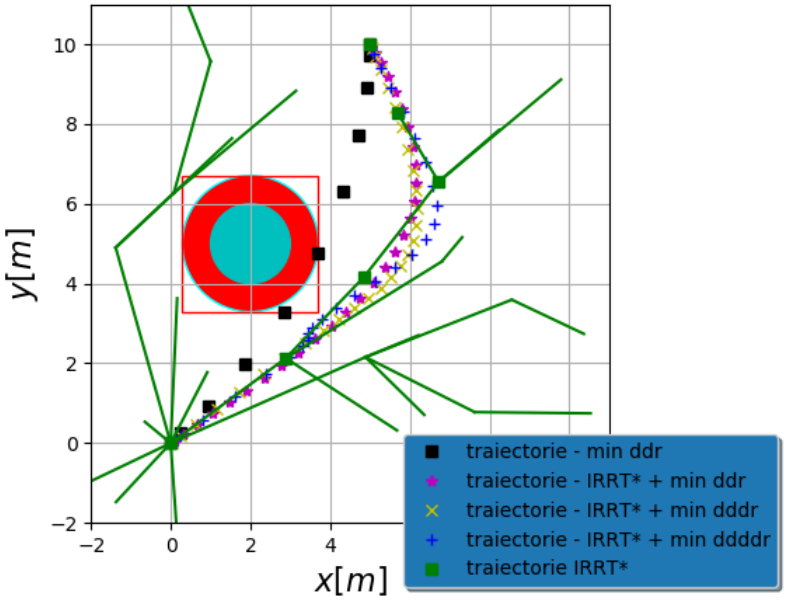


(a) IRRTOPT* trajectories

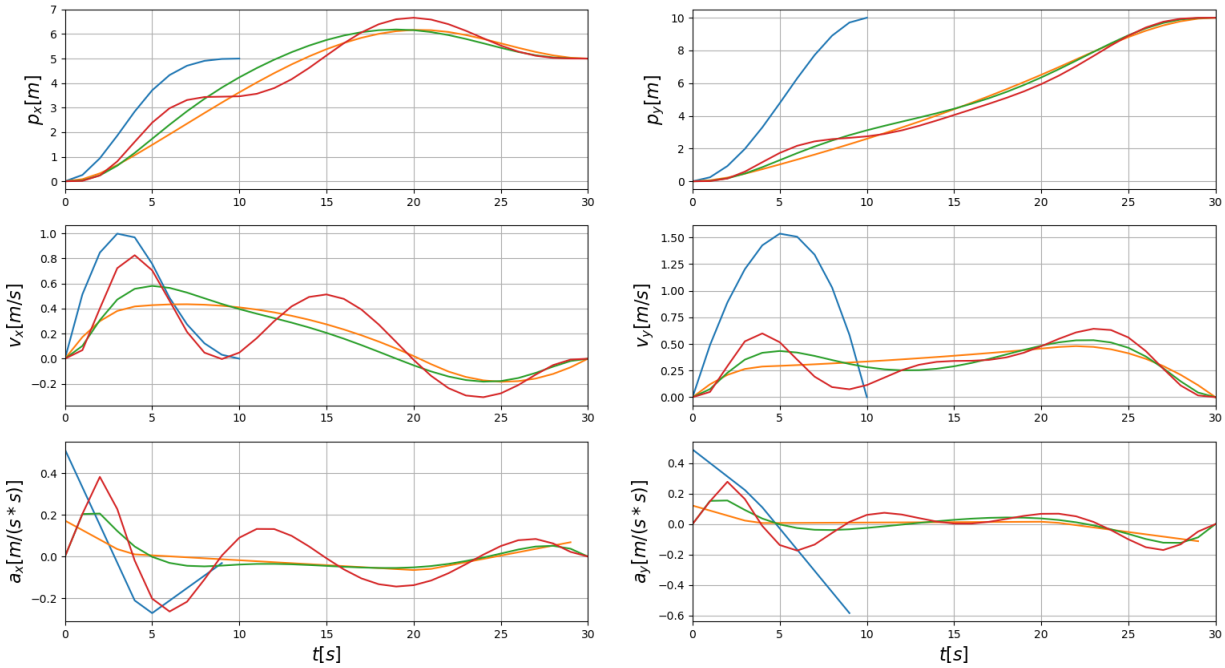


(b) Numerical results of the IRRTOPT* trajectories

Figure 7.2: 2D trajectory and evolution over time of the position, speed and acceleration of the quadcopter's translation avoiding a circle-shaped obstacle, obtained after the application of IRRTOPTACC (blue), IRRTOPTDACC (orange) and IRRTOPTDDACC (green) with the time step $\delta t = 1s$



(a) OPTACC and IRRTOPT* trajectories



(b) Numrical results of for OPT and IRRTOPT* trajectories

Figure 7.3: 2D trajectory and evolution over time of the position, speed and acceleration of the quadcopter's translation avoiding a circle-shaped obstacle, obtained after the application of OPTACC (blue), IRRTOPTACC (orange), IRRTOPTDACC (green) and IRRTOPTDDACC (red) with the step of time $\delta t = 1s$

7.3 Planning the trajectory for a two member flight formation, with inter-collision avoidance

Solving the flight path planning problem for two UAVs with inter-collision avoidance requires modifying/completing the OPT* and IRRTOPT* problem set in a similar way to how the obstacle avoidance problem was solved. We consider that each UAV is surrounded by a circle, which is located during the flight in the center of the circle. Thus, the radius of the circle h will establish the area around the obstacle as a safe zone, where the access of the other members of the flight group is restricted. Mathematically, we can write that

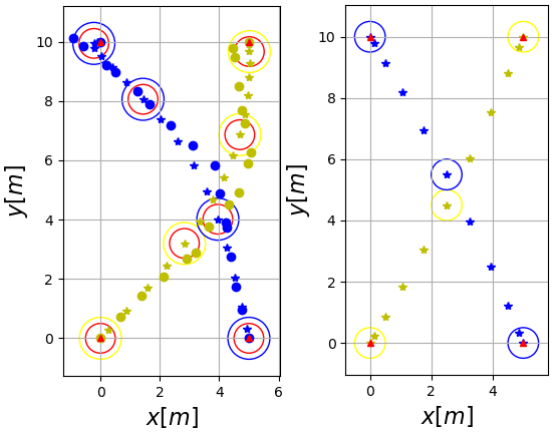
$$\begin{aligned} & x_t^i - x_t^j \geq dx_{ij} \\ sau \quad & x_t^j - x_t^i \geq dx_{ij} \\ sau \quad & y_t^i - y_t^j \geq dy_{ij} \\ sau \quad & y_t^j - y_t^i \geq dy_{ij}, \end{aligned} \tag{7.10}$$

expression in which, by using the superscript i and j, specify the position on both the axis x and the y of the UAV i at the time of the t and the UAV j, $\forall i, j \in [1, \mathrm{N}]$, $i \neq j$ and N represents the total number of UAVs. The value of the safe distance on the x and y axes is

$$d_{ij} = (||\dot{\mathbf{r}}^{\mathbf{i}}_{\mathbf{t,max}}|| + ||\dot{\mathbf{r}}^{\mathbf{j}}_{\mathbf{t,max}}||)\delta t. \tag{7.11}$$

In the case of two identically constructed quadcopter UAVs, $||\dot{\mathbf{r}}^{\mathbf{i}}_{\mathbf{t,max}}||=||\dot{\mathbf{r}}^{\mathbf{j}}_{\mathbf{t,max}}||$ with (7.11), we can write:

$$d_{ij} = 2||\dot{\mathbf{r}}^{\mathbf{i}}_{\mathbf{t,max}}||\delta t, \ \forall i \in [1, \mathrm{N}]. \tag{7.12}$$

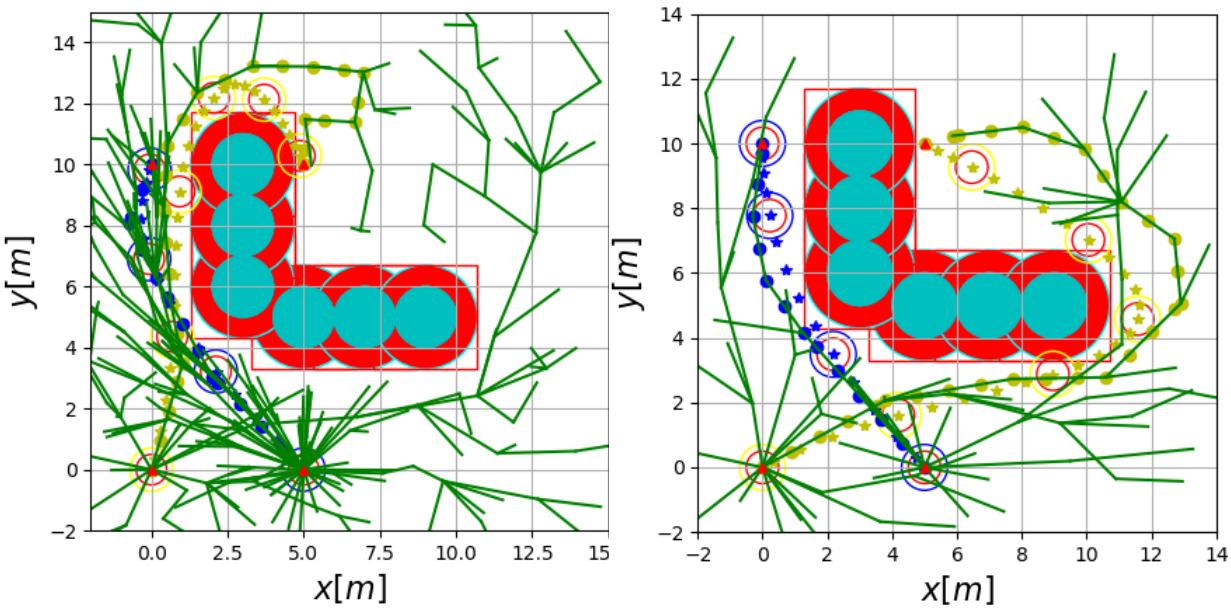


(a) IRRTOPTACC (b) OPTACC

Figure 7.4: 2D trajectory of two quadcopters in the absence of obstacles, using IRRTOTTACC and OPTACC

7.4 Planning the trajectory for a two member flight formation, with inter-collision and static obstacles collision avoidance

In this sub-chapter we will piece together those studied on avoiding collisions with both static and dynamic obstacles. Thus, we aim to formulate and solve the problem of flight path planning that corresponds to the applications as close to reality as possible. Flying in formation involves avoiding collisions between the band members and their existing obstacles. Achieving this objective requires integrating all the restrictions made in the previous chapters into the flight path planning issue to form the OPT* set as well as the IRRTOPT* set. The probabilistic nature of the IRRT* algorithm involved in the IRRTOTTACC method may also lead to unfavourable results in energy consumption. Such an example is captured in fig. 7.5. Even if the trajectories of the two UAVs in the figure 7.5b have been determined in approximately 0.84s, the yellow coloured one is far from the optimal solution, its length being approximately 24.63% higher than that calculated and highlighted in the figure 7.5a, which is 6.6% longer than the optimal one.



(a) 500 iterations for IIRT*

(b) 200 iterations for IRRT*

Figure 7.5: 2D trajectory of two quadcopters in the presence of obstacles, using IRRTOTTACC

7.5 Conclusions

In this chapter, the problem of flight path planning in the presence of static obstacles for a two-member flight formation was solved, and the following objectives were met:

- **OB10, OB11**, subchapter 7.2;
- **OB12, OB13**, subchapter 7.3;
- **OB14, OB15**, subchapter 7.4.

Numerical resolution of the trajectory planning problem using IRRTOPT* and OPT* methods highlighted the following aspects:

- Using OPTACC, each time the optimal solution was obtained in the form of the minimum length trajectory;
- The execution time of the algorithm corresponding to the OPTACC method increases exponentially with the number of flight group members and the number of obstacles, and this is not comfortable with applications running in real time, such as, for example, the flight scenario illustrated in fig. 1.2, proposed to achieve the objectives of the thesis;
- The IRRTOPT* method family has led to a lower energy consumption compared to OPT*, as long as the number of samples corresponding to the IRRT* component of the developed method is set at a value that ensures a <1s execution time;
- It is necessary to develop a calculation relationship that generates the number of samples for the IRRT* method, depending on the layout of the obstacles as well as the ratio $C_{obst}/C_{spațiu}$.

Chapter 8

Results and Conclusions

8.1 General conclusions

In order to achieve the fifteen proposed objectives, the thesis is structured into eight chapters including an introductory chapter for familiarization with the issue of flight path planning, and a final chapter, reserved for results and conclusions, in which the main contributions of the author are also punctuated.

8.2 Contributions

The contributions are summarised in the following:

Chapter 1.3:

- Synthesis in the literature of information related to the flight path planning implications for a UAV formation;
- Presentation of cooperative and distributive flight path planning concepts for a vehicle formation;
- Proposal for resolution of a flight mission and related scenario for cooperative flight path planning for a formation of UAVs.

Chapter 2:

- Presentation of important categories of trajectory planning methods used in both land and air vehicle applications;
- Python language software for numerical simulation of methods/algorithms used within trajectory planning categories;
- Highlighting and analysing the advantages and disadvantages of trajectory planning methods on the basis of the results of their numerical simulations, but also on the basis of consultation of the literature;
- Formulate a proposal to improve the IRRT algorithm* by adding numeric optimization elements.

Chapter 3:
- Modeling a member of the flight group (rotating wing aircraft) using the Newton-Euler equations;
- Derivation of the nonlinear model of the quadcopter.

Chapter 4:
- Brief presentation of the advantages and disadvantages of the different control algorithms for the types of controllers studied for flight control;
- Formulation and use of specific assumptions 'fixed-point flight' for linearization of the proposed non-linear model of the quadcopter;
- Determination of control laws for flight on a 3D trajectory using the linear PD regulator;
- Making software in Matlab for altitude control, 2D movement and quadcopter movement, tracking two imposed trajectory profiles, the first consisting of interconnected right segments and the second in the shape of a spiral;
- Identification of coefficients in flight control laws by method 'trial and error';
- Highlighting and discussing the influence of coefficient values in control laws on the accuracy of the quadcopter's flight path tracking.

Chapter 5:
- Design of quadcopter aerial platform and presentation of components used for its construction;
- Hardware system architecture design that equips the quadcopter (electronic sensor data acquisition mode, electronic control and control mode, electronic radio communication mode, electronic DC voltage stabilization mode);
- Hardware realization of the automatic control system and the fast-up presentation of developed versions;
- Software implementation of PD control laws using the 'Embedded Coder' software package in Matlab to generate C/C++ files, required to compile and write the executable on an ARM microcontroller architecture, STM32F103;
- C-language software implementation of interface drivers with sensors in the composition of the automatic control system and serial communication drivers for communication with them as well as with the human operator;
- Software development in the C# language of a graphical interface for monitoring flight parameters and flight path planning (by fixing GPS points) for the built quadcopter;
- Conducting system ground testing procedures 'controlled quadcopter' for the completion of the controlled platform and flight preparation;
- Using the app GoogleEarth and the development of software in Python and Matlab languages for ground analysis of data transmitted from the quadcopter, after tracking 2D trajectories (constant maintenance of flight altitude) and 3D (flight mode implemented software 'Return to Home';
- Numerical and graphical analysis of the experimental data in order to validate the proper functioning of the projected control algorithm.

Chapter 6:

- Customize Euler-Lagrange equations to solve the flight path planning problem for the built quadcopter;
- Elaboration of flight path planning method for a quadcopter using Euler-Lagrange equations, while minimizing translation acceleration;
- Software development in Matlab language to implement the method of solving the flight path planning problem for a quadcopter while minimizing the translation acceleration;
- Validation by numerical simulations and demonstration of the performance of the flight path planning method with respect to the accuracy with which this trajectory is tracked by the quadcopter under conditions of minimisation of the translation acceleration;
- Demonstration of the influence of the variation of forces and moments acting on the structure of the quadcopter, on the order I and II derivatives of the translation acceleration;
- Adapting the flight path planning method for a quadcopter using the Euler-Lagrange equations to the conditions imposed by minimizing the variation of the translation acceleration and the moments acting on the structure of the quadcopter;
- Matlab software development for the calculation of flight trajectories required under the conditions of minimizing the translation acceleration, its variation and the moments acting on the structure of the quadcopter;
- Validation by numerical simulations and demonstration of the performance of flight path planning methods with regard to the accuracy with which these trajectories are tracked by the quadcopter, under conditions of minimizing the translational aceleration, its variation and the moments acting on the structure of the quadcopter;
- Deduction and software implementation, using the Matlab language, of the mathematical relationships of approximation of electricity, consumed by the quadcopter using the values of forces developed by the four motor-propeller assemblies, captured at each $\delta t = 0.004s$ (time of execution of the attitude control loop);
- Numerical simulation and energy consumption assessment of trajectory planning methods in various flight scenarios.

Chapter 7:

- The formulation of restrictions on avoiding collisions with a static obstacle in the shape of a square or circle, as well as several static obstacles in the shape of circles, for a UAV;
- Add constraints on avoiding collisions with static obstacles for a UAV and deriving the OPT* problem set, the resolution of which involves minimizing the translation acceleration and its order I and II derivatives;
- Proposal of the IRRTOPT* problem set to solve the trajectory planning problem by minimizing energy consumption under conditions of avoidance of obstacles to solitary flight, as a superior approach to that formulated by the OPT* set;
- Use Python language software to solve the OPT* and IRRTOPT* problem set;

- Numerical simulation evaluation of the performance offered by the OPT* and IRRTOPT* methods, related both to the accuracy of the tracking of the required trajectories and to the reduction of energy consumption, under conditions of avoiding collisions with static obstacles;
- Introducing restrictions on avoiding collisions between two members and customizing OPT* and IRRTOPT* methods for simulating a flight for two UAVs;
- Numerical simulation evaluation of the performance offered by the OPT* and IRRTOPT* methods, related both to the accuracy of the tracking of the required trajectories and to the reduction of energy consumption, under conditions of inter-member collision avoidance;
- Numerical simulation evaluation of the performance offered by the OPTACC versus IRRTOTTACC method, related both to the accuracy of the tracking of the required trajectories and to the reduction of energy consumption, under conditions of inter-member and static obstacles collision avoidance.

8.3 Prospects for further development

Among the further research directions, we mention the study on the processing of navigation data by developing complex algorithms to be evaluated in terms of the accuracy of tracking flight trajectories calculated by the developed OPT* and IRRTOPT* methods. This requires experimental testing of the developed OPT* and IRRTOPT* methods to plan the trajectory of the quadcopters, to minimize energy consumption and to avoid inter-member collisions and with static obstacles. In this respect, we will need to integrate both hardware and software, a differential GPS sensor to reduce the error in position.

Bibliography

[1] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[2] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. D. Lee, A. Stewart, P. Vernaza, J. C. Derenick, J. R. Spletzer, and B. Satterfield, "Little ben: The ben franklin racing team's entry in the 2007 DARPA urban challenge," in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, ser. Springer Tracts in Advanced Robotics, M. Buehler, K. Iagnemma, and S. Singh, Eds., vol. 56. Springer, 2009, pp. 231–255. [Online]. Available: https://doi.org/10.1007/978-3-642-03991-1_6

[3] Y.-L. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, "Terramax™: Team oshkosh urban robot," *Journal of Field Robotics*, vol. 25, no. 10, pp. 841–860, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20267

[4] B. Patz, Y. Papelis, R. Pillat, G. Stein, and D. Harper, "A practical approach to robotic design for the darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 528–566, 2008.

[5] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *2012 IEEE Intelligent Vehicles Symposium*, 2012, pp. 383–388.

[6] A. Bacha, "Odin: Team victortango's entry in the darpa urban challenge," *J. Field Robot*, vol. 25, no. 8, pp. 467–492, Aug. 2008.

[7] R. Kala and K. Warwick, "Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization," *J Intell Robot Syst*, vol. 72, pp. 559–590, 2013.

[8] C. Urmson, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot*, vol. 25, no. 8, pp. 425–466, Aug. 2008.

[9] D. Ferguson, T. Howard, and M. Likhachev, "Motion planning in urban environments," *J. Field Robot*, vol. 25, no. 11, pp. 939–960, Nov./Dec. 2008.

[10] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res*, vol. 28, no. 8, pp. 933–945, Aug. 2009.

[11] M. Montemerlo, "Junior: The stanford entry in the urban challenge," *J. Field Robot*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[12] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005.

[14] J. Ren, K. McIsaac, and R. Patel, "Modified newton's method applied to potential field-based navigation for mobile robots," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 384–391, apr 2006.

[15] Y. bo Chen, G. chen Luo, Y. song Mei, J. qiao Yu, and X. long Su, "UAV path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, jun 2014.

[16] B. Di, R. Zhou, and H. Duan, "Potential field based receding horizon motion planning for centrality-aware multiple UAV cooperative surveillance," *Aerospace Science and Technology*, vol. 46, pp. 386–397, oct 2015.

[17] Y. Kuwata, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[18] D. Braid, A. Broggi, and G. Schmiedel, "The terramax autonomous vehicle," *J. Field Robot*, vol. 23, no. 9, pp. 693–708, Sep. 2006.

[19] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt$*$," in *Proc. IEEE ICRA*, 2011, pp. 1478–1483.

[20] J. Jeon, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *Proc. ACC*, Jun. 2013, pp. 188–193.

[21] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[22] Jonathan D. Gammell and Siddhartha S. Srinivasa and Timothy D. Barfoot, "Batch informed trees (BITstar): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2015.

[23] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRTstar: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2014.

[24] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. A. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4207–4214, 2016.

[25] L. Han, H. Yashiro, H. Tehrani Nik Nejad, Q. H. Do, and S. Mita, "Bézier curve based path planning for autonomous vehicle in urban environment," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 1036–1042.

[26] Z. Liang, G. Zheng, and J. Li, "Automatic parking path optimization based on bezier curve fitting," in *2012 IEEE International Conference on Automation and Logistics*, 2012, pp. 583–587.

[27] S. Thrun, "Stanley: The robot that won the darpa grand challenge," *J. Field Robot*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[28] A. Piazzi, C. G. Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic g/sup 2/-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002.

[29] P. Trepagnier, J. Nagel, P. Kinney, C. Koutsougeras, and M. Dooner, "Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain," *J. Field Robot*, vol. 23, no. 8, pp. 509–526, Aug. 2006.

[30] T. Gu and J. Dolan, *On-Road Motion Planning for Autonomous Vehicles*. Berlin, Germany: Springer-Verlag, 2012, pp. 588–597.

[31] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha-a local, continuous method," in *Proc. IEEE Intell. Veh. Symp*, 2014, pp. 450–457.

[32] R.-V. Mihai and M.-M. Bivolaru, "Cooperative distributed trajectory optimization for a heterogeneous UAV formation." Author(s), 2018.

[33] R.-V. Mihai, C. Vidan, A.-D. Radu, and A. Gavril, "Vision based cooperative distributed collision avoidance estimation in a dynamic environment for a heterogeneous UAV formation," in *CENTRAL EUROPEAN SYMPOSIUM ON THERMOPHYSICS 2019 (CEST)*. AIP Publishing, 2019.

[34] R.-V. Mihai and A.-M. Stoica, "On the development of an onboard real time system for the members of a heterogeneous UAV formation acting as LPWAN gateways," in *CENTRAL EUROPEAN SYMPOSIUM ON THERMOPHYSICS 2019 (CEST)*. AIP Publishing, 2019.

[35] Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.

[36] I. Noreen, A. Khan, and Z. Habib, "A comparison of rrt, rrt* and rrt*-smart path planning algorithms," 2016.

[37] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[38] J. Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, 2008, pp. 158–166.

[39] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, jun 1994.

[40] T. Schouwenaars, A. Richards, E. Feron, and J. How, "Plume avoidance maneuver planning using mixed integer linear programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, aug 2001.

[41] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. IEEE, 2002.

[42] N. P. Nguyen and S. K. Hong, "Sliding mode thau observer for actuator fault diagnosis of quadcopter UAVs," *Applied Sciences*, vol. 8, no. 10, p. 1893, oct 2018.

[43] P. Rucz, Z. Belso, B. Gati, I. Koller, and A. Turóczi, "Design and implementation of nonlinear control systems for rotary and fixed wing uavs," 2016.

[44] I. Grujić and R. Nilsson, "Model-based development and evaluation of control for complex multi-domain systems: Attitude control for a quadrotor uav," 2016.

[45] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, mar 2004.

[46] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*. Springer London, 2008.