BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 67 (71), Numărul 2, 2021
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ
DOI:10.2478/bipie-2021-0012

PREREQUISITES TO DESIGN A COLLISION FREE TRAJECTORY IN A 3D DYNAMIC ENVIRONMENT FOR AN UAV

BY

SOFIA HUŞTIU[1,2,*]

[1]“Gheorghe Asachi” Technical University of Iasi,
Faculty of Automatic Control and Computer Engineering
[2]University of Zaragoza, Spain
Engineering Research Institute of Aragon (I3A)

Received: October 11, 2021
Accepted for publication: December 5, 2021

Abstract. This research presents the main steps needed for designing a piece-wise linear trajectory which grants an unmanned aerial vehicle (UAV) to reach a destination pose in a workspace with dynamic obstacles. The first objective is characterized by the examination of kinematics and dynamics of a quadcopter. For this purpose, a nonlinear mathematical model was developed. The validation of the mathematical model was confirmed by MATLAB and real time experiments. The second step aims to properly map the 3D environment. For this objective, an algorithm for a cuboid rectangular decomposition was developed and implemented, by extending a 2D decomposition technique. The evaluation of the proposed path planning algorithm occurred for different scenarios and the validation of the results was established through numerical simulations. In the end, a comparison for two path planning scenarios is shown: for a 3D static environment and for a 3D dynamic environment, where the movement of a dynamic obstacle is known.

Keywords: path planning; unmanned aerial vehicle; dynamic environment; cell decomposition.

---

*Corresponding author; *e-mail*: sofia.hustiu@academic.tuiasi.ro

1. Introduction

In the last years, Unmanned Aerial Vehicles (UAVs), also known as drones, have been used frequently in different applications such as military and surveillance, aerial photography, mapping the environment, and entertainment domain, *e.g.*, drone racing. All these applications have in common the understanding of kinematic and dynamic of drone. This is mandatory for a collision free path planning.

One challenge in 3D path planning with UAVs is represented by studying their dynamic. This is a complex problem, caused by the presence of six degrees of freedom of the drone. In this work, we have analyzed this problem by selecting a particular type of drone: quadcopter. After this study, the next step is captured by developing a control law for the trajectory. Several papers tackle issue (Gheorghiță *et al.*, 2015; Tagay *et al.*, 2021). In (Grieff, 2017) is described a control planning method for a nano quadcopter Crazyflie. Analyzing the latter paper, we developed a mathematical model for Crazyflie 2.0, which was validated in real indoor workspace. This contribution defines the first prerequisite described in our work.

Another critical topic is defined by a collision free trajectory for the UAV, independent of the 3D space: static or dynamic. The collision free techniques can be based on classical approaches, *e.g.*, Rapidly Exploring Random Trees (RRT) (Yao *et al.* 2015), Voronoi partitioning (Fang *et al.*, 2017), Artificial potential (Liu *et al.*, 2016), or based on particular approaches, *e.g.*, (Goel *et al.*, 2018) - using an optimal flight routine in real environments, (Paranjape *et al.*, 2015) - considers a time-delay for 3D circular path combined with aggressive turn-around maneuvers (ATA).

Other methods are represented by swarm intelligence algorithms, simulating the nature's behavior, *e.g.*, (Duan *et al.*, 2014) – based on pigeon behavior, (Ge *et al.*, 2020) – based on fruit fly behavior. A different technique is represented by discrete models *e.g.*, using Petri Net Toolbox in MATLAB (Păstrăvanu *et al.*, 2004). In (Mahulea *et al.*, 2018) the authors used Petri Net model the movements of a team of mobile robots, given a Boolean-based specification.

The collision free path is especially difficult to be acquired in dynamic environments, especially when the movement of the obstacles is uncertain. Usually, the UAV requires an accurate localization in the 3D space, the on-board sensors must be quick enough to detect and to monitor the movements in its surroundings, and the actuators should act fast to avoid the obstacles, based on a robust control law algorithm. Therefore, the complexity for this problem is increased. The work from this paper captures the study on a collision free trajectory in a known environment, this representing the second prerequisite for the UAV path planning area. We assume that this study will give us a valuable base of knowledge, before tackling the collision avoidance topic in dynamic and uncertain environments.

In the given context, the purpose of this paper is to present the main steps which are necessary to develop a collision free trajectory for an UAV from an initial to a desired final point, evolving in a dynamic 3D environment. The movement of the dynamic obstacle is assumed to be known. This work combines the results from (Huștiu *et al.*, 2018; Lupașcu *et al.*, 2019, Huștiu *et al.*, 2020), adding new examples, explications and interpretations. The benefit of these precondition is evaluated through numerical simulations, by achieving a collision free trajectory in a known dynamic workspace.

The next section describes the prerequisites for a piece-wise trajectory, divided into two parts: computation of a mathematical model for a quadcopter and mapping the 3D workspace to allow the drone to evolve in a dynamic space. The section 3 emphasizes the algorithm used for path planning, and section 4 captures the results of the proposed algorithm, visualized through numerical simulations. Also, a comparison between two different trajectories is commented (one trajectory considers only the static environment and the second trajectory considers the movement of the dynamic obstacle, thus being able to avoid the collision with it). At the end, the conclusions are presented in section 5.

2. Prerequisites for a 3D Trajectory

This section describes the prerequisites necessary to develop a piece-wise linear trajectory for an omnidirectional drone in a dynamic known environment. For this reason, the problem formulation is defined, alongside with several mathematical notations used throughout this paper.

This work includes a comparison between two UAV paths, in different scenarios: static and dynamic. For this reason, let us consider first a 3D environment in a shape of a rectangular cuboid with fixed limits, denoted as $E = [x_{min}, x_{max}]x[y_{min}, y_{max}]x[z_{min}, z_{max}]$, with $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max} \in \mathbb{R}$. The environment contains n static obstacles (represented with purple color), having flat polygonal facets and flat base (z = 0). The obstacles are characterized as a convex and bounded polyhedron, defined in the set $O = \{O_1, O_2, \ldots O_n\}$. Fig. 1 illustrates a representation of the defined 3D space, including an initial position $(x_0, y_0, z_0)^T \in \mathbb{R}^3$(red circle) and a desired final position $\left(x_g, y_g, z_g\right)^T \in \mathbb{R}^3$ (blue star). In addition to this specified environment, one dynamical is defined $DOb = \left[x_{Dyn_{min}}, x_{Dyn_{max}}\right]x\left[y_{Dyn_{min}}, y_{Dyn_{max}}\right]x\left[z_{Dyn_{min}}, z_{Dyn_{max}}\right]$, with $x_{Dyn_{min}}, x_{Dyn_{max}}, y_{Dyn_{min}}, y_{Dyn_{max}}, z_{Dyn_{min}}, z_{Dyn_{max}} \in \mathbb{R}$. The shape of it is pictured as a rectangular parallelepiped (orange color) and the movement is represented with the orange dashed line, depicted in Fig. 1. The velocity of the dynamic obstacle is denoted with vel_{DOb} and the direction is given by dir_{DOb}.

In both scenarios, the drone evolves with a constant velocity, denoted as vel_{drone}.

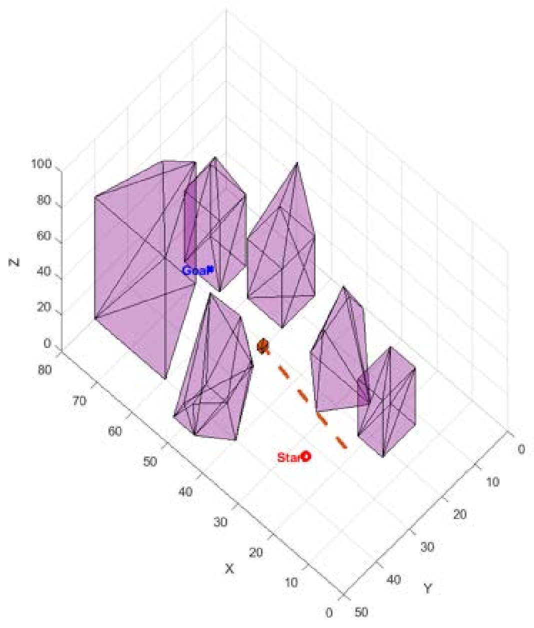

Fig. 1 – Environment *E* with static obstacles (purple), initial position of *DOb* (orange) and movement direction (dashed orange line).

The mathematical description of a 3D convex polyhedral is essential for the proposed path planning algorithm. Therefore, two representations are given, as follows:

- *V-representation*- based on the coordinates of the vertices which describes the shape. P denotes the convex hull of the vertices (Eq. (1)). This representation is valid only when the number of vertices is at least equal with three. With $\lambda_i \in [0, 1]$ was represented the constraints.

$$P = convhuull(v_1, \dots v_p) = \{q \in \mathbb{R}^3 | \quad q = \sum_{i=1}^{p} \lambda_i v_i \, \forall \lambda_i \in [0,1] \, with \sum_{i=1}^{p} \lambda_i = 1\} \tag{1}$$

- *H-representation*- based on the intersection of half spaces. The inequalities in Eq. (2) describes the half space, where $A \in \mathbb{R}^{mx3}$ $and\ b \in \mathbb{R}^m$.

$$P = \{q \in \mathbb{R}^3 | \, Aq \le b\} \tag{2}$$

The following subsection, in Algorithm 2, the importance of these two representations is emphasized. A conversion between them is required (from V-representation to H-representation) when it is needed to test if two polyhedral are intersected, *e.g.*, a section of the discretized environment, denoted as a cell, with the dynamic obstacle.

2.1. Mathematical Model of a Quadcopter

The first prerequisite in regards to a collision free trajectory for an UAV is represented by analysing the physics of the drone. First, the mathematical model of a quadcopter was proposed in (Huștiu *et al*., 2018). The current section provides more details regarding the construction of it. The model contributes to understand the kinematics and dynamics of a drone, by studying the physical laws and forces applied for its motion. For this reason, a particular type of drone was selected for this analysis, represented by a quadcopter (Fig. 2).

The mathematical model was computed based on Newton-Euler equations, which considers the following assumptions about the drone: rigid symmetrical body, constant mass, and the Center of Gravity (CoG) is positioned in the middle of the symmetrical structure. These assumptions help in simplifying the mathematical model of the quadcopter. Fig. 2 presents both the inertial and body frames, essential for expressing the states of this system in the inertial frame. For this reason, several rotational transformations of the coordinates are computed, captured in the following equations.

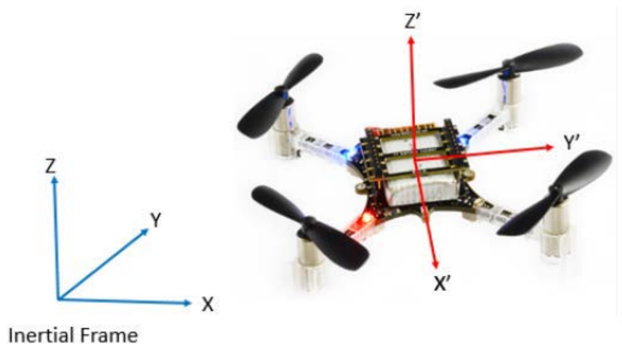

Fig. 2 – Inertial frame (left) and body frame (right).

Eq. (3) describes the nonlinear mathematical model of the quadcopter, based on several inputs:

- basic movements of an UAV, such as: thrust (the force necessary to lift the drone from the ground) and orientation around each axis (roll, pitch and yaw).
- force and momentum equilibrium.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{r} \\ \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} w(s_\psi s_\phi + c_\psi c_\phi s_\theta) - v(s_\psi c_\phi - c_\psi s_\phi s_\theta) + u c_\psi c_\theta \\ v(c_\psi c_\phi + s_\psi s_\phi s_\theta) - w(c_\psi s_\phi - s_\psi c_\phi s_\theta) + u s_\psi c_\theta \\ w c_\theta c_\phi - u s_\theta + v s_\phi c_\theta \\ p + r c_\phi t_\theta + q s_\phi t_\theta \\ q c_\phi - r s_\phi \\ r\dfrac{c_\phi}{c_\theta} + q\dfrac{s_\phi}{c_\theta} \\ rv - qw + g s_\theta \\ pw - ru - g s_\phi c_\theta \\ qu - pv + \dfrac{U_1}{m} - g c_\theta c_\phi \\ \dfrac{U_2 + pq(I_{xx} - I_{yy})}{I_{zz}} \\ \dfrac{U_3 + pr(I_{zz} - I_{xx})}{I_{yy}} \\ \dfrac{U_4 + pq(I_{zz} - I_{yy})}{I_{xx}} \end{bmatrix} \tag{3}$$

The first six states represent the pose of the quadcopter: *x*, *y*, *z* for positions, and roll ϕ, pitch θ and yaw ψ angles for orientation. The next six states captures the linear *u*, *v*, *w* and angular *p*, *q*, *r* velocities. Other notations used and observed in Eq. (3) are: s_x = sin(x), c_x= cos(x), t_x = tg(x), m – mass of quadcopter, g – gravitational accelearation, I_{xx}, I_{yy}, I_{zz} – moments of inertia around each axis.

The generic inputs of a quadcopter depend on the angular velocity of each rotor ω_i, where $i=\overline{1,4}$, in different combinations, based on the required movement. In order to describe these inputs, visualized in Eq. (4), three coefficients were defined: C_T – thrust coefficient, C_D – drag coefficient, *d* – arm length of the quadcopter:

$$\begin{cases} U_1 = C_T(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 = dC_T\sqrt{2}(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_3 = dC_T\sqrt{2}(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \\ U_4 = C_D(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \tag{4}$$

This mathematical nonlinear model with 12 states was developed in Matlab – Simulink and validated in real experiments in open loop (Fig. 3), by considering the same input in both cases and the numerical coefficients of the nano-quadcopter Crazyflie 2.0. The input command is represented by the thrust force which is given as a 16 bit number, between 0 and 65535. For validation, it was assigned the value of thrust force 55000, and afterwards the value decreased to 35000. The graphic from Fig. 3 captures the behavior of drone both in simulation and in experimental results (the z position of drone is rising for 1.6 seconds, and then decreases). The drone ascends between 0.7 – 1.6 seconds due to inertia.

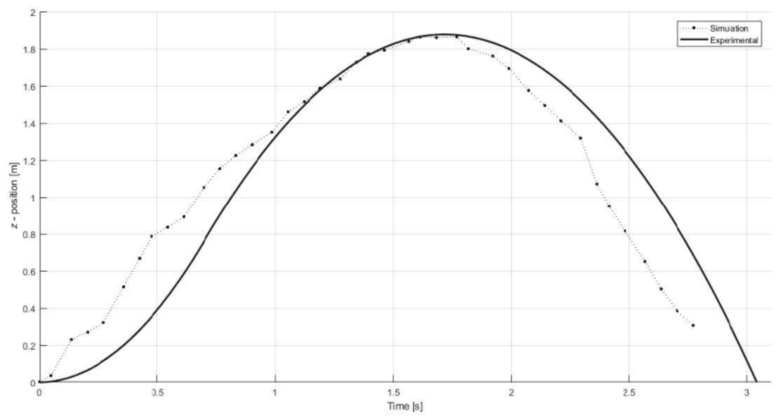

Fig. 3 – Behavior of position z (experimental results – black points, numerical simulation – black line).

As a conclusion, the model validated in real experiments demonstrates a good value to the proposed mathematical model. In the next simulation for path planning, the drone is assumed omnidirectional, based on its 6 degrees of freedom, represented by the linear and angular movement of each axis.

2.2. Mapping the 3D Environment

The second precondition necessary for developing a collision free trajectory in a 3D environment with obstacles is given by the mapping of the free space. For this purpose, an extension of a classical 2D decomposition technique was made for a 3D approach. The decomposition chosen was based on rectangular shape type. The main idea of this decomposition is to split iteratively the environment into cells until a certain precision is reached ε, these being labelled as: *occupied* if it lies entirely in at least one obstacle, *free* if it does not intersect any obstacle and *mixed* if is neither free nor occupied. All the cells have the same size ratio as the size of the environment because the method divides all axis in half for each mixed cell.

After the full environment is divided into cells, each free cell becomes a node in a graph *G*, and the edges are given by an adjacency matrix *Adj*. This transformation helps in computing a collision free trajectory, described in the next subsection. Two cells *i*, *j*, with *i*≠*j*, are adjacent if they share a facet in common (4 vertices). This definition is relevant only for the 3D approach. In 2D decomposition, the relation of adjacency has other definition. Thus, when two cells are adjacent, *Adj*(*i*,*j*) = 1, otherwise *Adj*(*i*,*j*) = 0. For an easier understanding of the proposed technique, a pseudocode is presented below:

Algorithm 1 Procedure *label_split_cuboid*

Input: Current decomposition C, current cuboid bounds $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$, set of obstacles *O*, volumetric threshold ε

Output: Update decomposition C

```
1  RC = [$x_{min}, x_{max}$] x [$y_{min}, y_{max}$] x [$z_{min}, z_{max}$] is the current rectangular cuboid
2  $volume_{intersection} = \sum_{i=1}^{n} volume(RC \cap O_i)$
3  if $volume_{intersection}$= 0, then
4      /* RC is free */
5      C = C $\cup$ {$RC$}return C
6  else
7      if $volume_{intersection} < volume(RC)$ and (volume(RC) $\leq \varepsilon$), then
8          /* RC is mixed and should be split */
9          C = label_split_cuboid (C,$x_{min}, \frac{x_{min}+x_{max}}{2}, y_{min}, \frac{y_{min}+y_{max}}{2}, z_{min}, \frac{z_{min}+z_{max}}{2}, \varepsilon, 0$)
10         C = label_split_cuboid (C,$x_{min}, \frac{x_{min}+x_{max}}{2}, \frac{y_{min}+y_{max}}{2}, y_{max}, z_{min}, \frac{z_{min}+z_{max}}{2}, \varepsilon, 0$)
11         C = label_split_cuboid (C,$\frac{x_{min}+x_{max}}{2}, x_{max}, y_{min}, \frac{y_{min}+y_{max}}{2}, z_{min}, \frac{z_{min}+z_{max}}{2}, \varepsilon, 0$)
12         C = label_split_cuboid (C,$\frac{x_{min}+x_{max}}{2}, x_{max}, \frac{y_{min}+y_{max}}{2}, y_{max}, z_{min}, \frac{z_{min}+z_{max}}{2}, \varepsilon, 0$)
13         C = label_split_cuboid (C,$x_{min}, \frac{x_{min}+x_{max}}{2}, y_{min}, \frac{y_{min}+y_{max}}{2}, \frac{z_{min}+z_{max}}{2}, z_{max}, \varepsilon, 0$)
14         C = label_split_cuboid (C,$x_{min}, \frac{x_{min}+x_{max}}{2}, \frac{y_{min}+y_{max}}{2}, y_{max}, \frac{z_{min}+z_{max}}{2}, z_{max}, \varepsilon, 0$)
15         C = label_split_cuboid (C,$\frac{x_{min}+x_{max}}{2}, x_{max}, y_{min}, \frac{y_{min}+y_{max}}{2}, \frac{z_{min}+z_{max}}{2}, z_{max}, \varepsilon, 0$)
16         C = label_split_cuboid (C,$\frac{x_{min}+x_{max}}{2}, x_{max}, \frac{y_{min}+y_{max}}{2}, y_{max},, \frac{z_{min}+z_{max}}{2}, z_{max}, \varepsilon, 0$)
17     else
18         /* RC is either occupied, or mixed but too small */
19         return C
```

The inputs required for this procedure are the set of obstacles *O*, the precision ε – impacting the size of the cells, the boundary of the environment and the set of cells *C*, which in the beginning is empty. The output of the procedure returns the updated set of cells *C*, due to the recursive process. In line 2 is computed the intersection of the environment *RC* considered as the first cell in the set *C* with each obstacle O_i, $i = \overline{1, n}$, by using H-representation. If this intersection is empty, then the cell is not divided furthermore (lines 3-5). On the other hand, if the current cell intersects at least one obstacle (line 7), the cell is labeled as mixed and is further split into 8 smaller and equal cuboids by cutting in half each axis, until a certain precision ε is reached (lines 8-18).

3. Collision Free Trajectory

This section is dedicated to the collision free trajectory algorithm proposed for this paper, based on the work captured in (Huștiu *et al.*, 2020), considering another environment scenario with different numerical values for the result part. Based on the previously presented prerequisites, two different trajectories were computed. The first one considers only the static environment, denoted with *StatTraj*, while the seconds one considers the movement of the dynamic obstacle, trajectory denoted with *DynTraj*. The latter trajectory is able to avoid the collision with *DynTraj*. Therefore, these two paths represent the output of the proposed algorithm.

In Algorithm 2, the first focus is directed towards *StatTraj*. Lines 2-9 includes the correlation between the set of free cells *C* and graph *G* mentioned in subsection 2.2. The environment is decomposed into cells, and the adjacency matrix is computed based on the verification illustrated in lines 4-7. For this reason, the intersection between two cells is converted from a H-representation to a V-representation and it is checked if two cuboids i, j, with $i \neq j$, have four vertices in common. If this condition is true (line 7), then the value for *Adj*(*i*,*j*) is modified to 1 and the centroid of the common facet is saved in $Waypoint_{i,j}$. The path is computed by the union of these waypoints, together with the start and goal position.

The next step consists in finding the cells which includes the starting and the goal position given as inputs for the drone (line 10). Based on these points, together with the graph G corresponding to the environment model, Dijkstra algorithm is applied for computing the minimum path (line 11). The returned cells of the procedure *shortest_path*, are translated into way points, expressing the positions in which the drone is allow to evolve into 3D space towards the goal position (lines 12-15).

The second part of the path planning algorithm concentrates on computing the trajectory denoted with *DynTraj*. For this reason, the global time *Time* is considered in the execution of the algorithm, together with a defined sample time T_s. These impact the estimation for the location of *DOb*at each moment in time T_s, based on its direction dir_{DOb} and its velocity vel_{DOb}. This estimation can be visualized on line 23. In order to develop the trajectory for the dynamic environment, the mapping of the workspace needs to be updated at each moment in time T_s. Therefore, the cells which intersect the dynamic obstacle are inhibited (lines 24-25). This way, the drone has the information that those cells will be avoided and will not be considered in path planning algorithm. The collision avoidance is verified by computing the distance between each cell and the current position of the dynamic obstacles $dist_{DOb,RC_i}$, with $i = \overline{1, |RC|}$. If this distance is smaller or equal with the sum between diagonal of *DOb* and diagonal of the current cell RC_i, then the movement of the *DOb* intersects the current cell. As a result, the input transitions to that cell are inhibited, thus forbidding the drone to pass through it.

Based on the same search graph Dijkstra algorithm, the sequence of cells of *DynTraj* are returned. The method used for this approach is updating iteratively. In the lines 29-34 is described the construction of *DynTraj* when the time to reach the desired position is smaller than the considered global time. On the lines 35-38 is expressed the path planning when the dynamic obstacle leaves the limits of the environment, and the drone hasn't arrived at the goal position.

Algorithm 2 Cell decomposition planning

Input: Environment E, obstacles O, precision E, Drone velocity vel_{Drone}, initial and goal positions $(x_0,y_0,z_0)^T$, $(x_g,y_g,z_g)^T$, Dynamic Object DOb, Dynamic Object velocity vel_{DOb}, Dynamic Object direction dir_{DOb}, Sample time T_S

Output: Free cells C, graph G, drone static trajectory $StatTraj$, drone dynamic trajectory $DynTraj$

```
1  C = $label\ split\ cuboid(E,\varepsilon,O)$
2  Graph $G = (N,Adj)$, with nodes $N = \{1,2,\dots,|C|\}$
3  $Adj = 0_{|C|\times|C|}$;
4  for $RC_i, RC_j \in C$, $i \neq j$ do
5      $\mathbf{Int_{i,j}} = \left\{ \boldsymbol{q} \in \mathbf{R}^3 \,\middle|\, \begin{bmatrix} \mathbf{A}_{\mathbf{RC_i}} \\ \mathbf{A}_{\mathbf{RC_j}} \end{bmatrix} \mathbf{q} \le \begin{bmatrix} \mathbf{b}_{\mathbf{RC_i}} \\ \mathbf{b}_{\mathbf{RC_j}} \end{bmatrix} \right\}$
6      Convert $Int_{i,j}$ to V-representation $VInt_{i,j}$
7      if $|VInt_{i,j}| = 4$ then
8          $Adj(i,j) = 1$
9          $Waypoint_{i,j} = centroid(VInt_{i,j})$
10 Find indexes $start, stop \in N$ such that $(x_0,y_0,z_0)^T \in RC_{start}$, $(x_g,y_g,z_g)^T \in RC_{stop}$
11 $Stat\_Cell\_Seq = shortest\ path(G, Adj, RC_{start}, RC_{stop})$
12 Denote $Stat\_Cell\_Seq = \{RC_0, RC_1, \dots, RC_{n_i}\}$
13 $StatTraj = (x_0,y_0,z_0)^T$
14 Append $Waypoint_{i,i+1}$ to $StatTraj$, $\forall i=0,\dots,n_i-1$
15 Append $(x_g,y_g,z_g)^T$ to $StatTraj$

16 Let $final_{time}$ be the time moment when obstacle $Dob$ leaves environment $E$
17 $Time = \{0, T_S, 2*T_S, \dots, final_{time}\}$
18 Initialize $Adj_t = Adj$, $\forall\ t \in Time$
19 Denote with $Obs_t$ the vertices of $Dob$ at time instant $t$ and with $Obs\ center_t$ its center
20 $DynTraj = (x_0,y_0,z_0)^T$ (current drone position)
21 for $t \in Time$ do
22     if $t > 0$ then
23         $Obs\_center_t = Obs\_center_{t-1} + dir_{DOb} * vel_{DOb} * T_S$
24     for $RC_i \in C$ such that $RC_i \cap Obs_t \neq \emptyset$ do
25         $Adj_t(:,i) = 0$
26     Let $RC_{current}$ be the cell containing the drone at time $t$
27     $Dyn\_Cell\_Seq = shortest\ path(G, Adj_t, RC_{current}, RC_{stop})$
28     Denote $Dyn\_Cell\_Seq = \{RC_{t_1}, RC_{t_2}, \dots, RC_{t_n}\}$
29     if $t+T_S < final_{time}$ then
30         Move drone with speed $vel_{Drone}$ along trajectory linking the current position
   with $Waypoint_{t_i,t_{i+1}}$ where $i=1,2,\dots$ until time moment $t+T_S$ or until cell $RC_{stop}$ is reached
```

```
            Append $Waypoint_{t_i,t_{i+1}}$ to $DynTraj$
            if $RC_{stop}$ was reached then
                Append $(x_g,y_g,z_g)^T$ to $DynTraj$
                Return
    else
            Move drone with speed $vel_{Drone}$ along trajectory linking the current position
            with $Waypoint_{t_i,t_{i+1}}$, where $i$=1,2,…
            Append $Waypoint_{t_i,t_{i+1}}$ to $DynTraj$
            Append $(x_g,y_g,z_g)^T$ to $DynTraj$
```

The method described in this subsection has a good base for an offline path planning algorithm, the idea being based on the model predictive control strategy, by examine the case with the look-ahead horizon of T_s. By predicting the position of the dynamic obstacle for each T_s moment based on its direction and velocity, the drone would know the forbidden cells in a selected future horizon. At the end, the drone could optimize the returned trajectory, based on the remaining free space and the selected future horizon.

4. Simulation Results

The evaluation of the proposed algorithm in regards to path planning for a drone, was done in MATLAB simulations, implemented on a mobile computer unit with Intel® Core™ i7-8750H CPU @ 2.20Ghz and 8GB RAM.

Let us consider a 3D environment with size 80 x 50 x 100 length unit (denoted as *lu*), visualized in Fig. 1, with 6 convex polyhedral static obstacles with flat base, simulating a city scenario. As mentioned previously, the first step for the path planning algorithm is represented by the mapping of the free space. Using a precision ε = 16, the assumed workspace was divided into 764 free cuboids in 2.97 seconds, as it can be seen in Fig. 4.

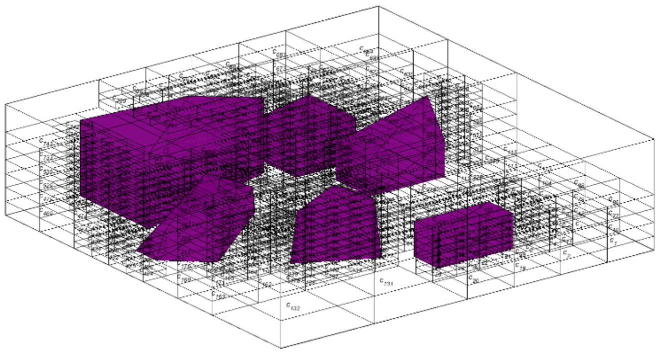

Fig. 4 – Rectangular cuboid decomposition for an environment with 6 static obstacles.

The size of the dynamic obstacle is 7 x 5 x 3 *lu*, moving with the velocity vel_{DOb} = 9 *lu/s*. On the other hand, the drone has the velocity vel_{drone} = 8 *lu/s*, departing from the initial position $(20,\ 40,\ 25)^T$, having the desired goal position with the coordinates $(50, 37, 73)^T$. For the iterative update of the environment model (consist in inhibiting the forbidden cells), it was considered T_s = 0.1 seconds. The Fig. 5 depicts the returned trajectories based on the algorithm presented in this paper. With black color is represented *StatTraj* – trajectory computed for the static environment and with green color is represented *DynTraj* – trajectory which considers the movement of the dynamic obstacles.

Through simulation it was observed that even though *StatTraj* reaches the goal position, after 4 seconds a collision occurs with the dynamic obstacles in the point with coordinates $(27.5,\ 37.5,\ 59.38)^T$.The collision is illustrated in Fig. 5. For a better visualization of this scenario, the reader can examine the animations in the suggested video, where additional numerical simulation was considered: https://www.youtube.com/watch?v=tZfjdH3Sf2U&ab_channel=LoRISwork.

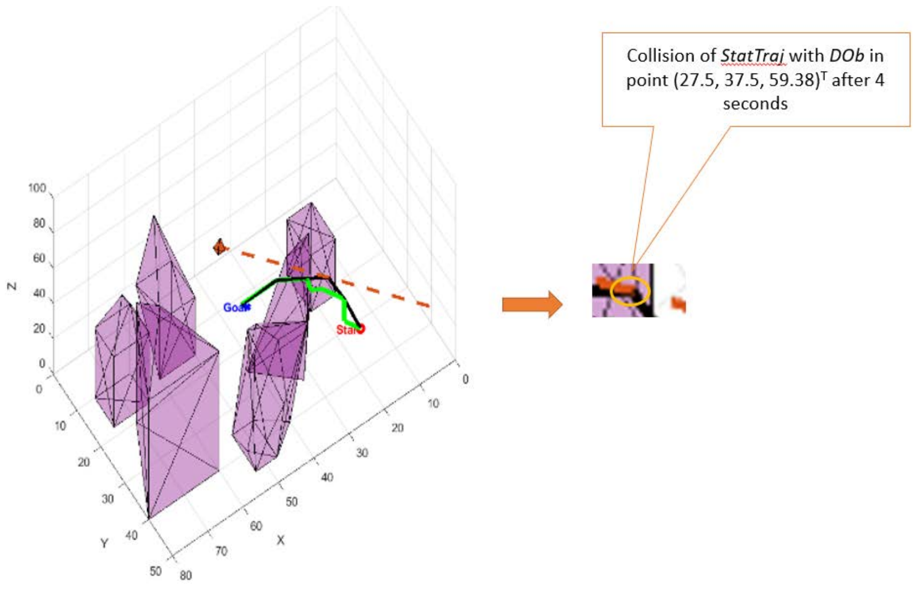

Fig. 5 – Drone trajectories (*DynTraj* with green, *StatTraj* with black) in dynamic environment. Collision avoidance between *StatTraj* with *Dob* (red dashed line).

Comparing the performances between these 2 trajectories, it was observed that the length of *StatTraj* is 62.53 *lu* with 8 changes of direction (or in other words, the drone traverse 8 cells to reach its destination) and the length of *DynTraj* is 68.04 *lu* with 9 changes of direction, therefore *StatTraj* has a smaller trajectory. Since in the calculation of *DynTraj* the map of the environment is updated at each sample time, the run time is longer compared with the run time for computing *StatTraj* (5.43 seconds, respectively 3.88

seconds). Nevertheless, the trajectory of *StatTraj* is only a hypothetical one in the context of dynamic 3D environment.

5. Conclusions

To conclude, this paper contains the preconditions necessary to develop a piece-wise collision free trajectory in a known 3D environment, with static and dynamic obstacles. The first prerequisite captures the analysis of kinematics and dynamics of a quadcopter, by modeling it with a mathematical nonlinear representation of 12 states. This model was validated in real time experiments, using nano quadcopter Crazyflie 2.0. The second prerequisite presented here assumes the mapping of a 3D environment based on a rectangular cuboid representation. Considering this mapping and the known movement of the dynamic obstacle which evolves in this workspace, the proposed algorithm generates effectively collision free trajectories for the drone, in both scenarios: static and dynamic spaces.

Future work will consider movement of a team of drones in a 3D environment, maintaining the idea of collision avoidance between them, necessary in achieving one mission for the entire team.

REFERENCES

Duan H., Qiao P., *Pigeon-Inspired Optimization: A New Swarm Intelligence Optimizer for Air Robot Path Planning*, Intelligent Computing and Cybernetics, 2014, **7**, 24-37.

Fang Z., Luan C., Sun Z., *A 2d Voronoi-Based Random Tree for Path Planning in Complicated 3d Environments*, Intelligent Autonomous Systems, 2017, **31**, 433-445.

Ge F., Li K., Han Y., Xu W., Wang Y., *Path Planning for Oilfield Inspection in a Three-Dimensional Dynamic Environment with Moving Obstacles Based on Improved Pigeon-Inspired Optimization Algorithm*, Applied Intelligence, 1-18 (2020).

Gheorghiță D., Vîntu I., Mirea L., Brăescu C. *Quadcopter Control System*, 19th International Conference on System Theory, Control and Computing (ICSTCC), 2015, 421-426.

Goel U., Varshney S., Jain A., Maheshwari S., Shukla A., *Three Dimensional Path Planning for UAVs in Dynamic Environment Using Glow-Worm Swarm Optimization*, Procedia Computer Science, 2018, **133**, 230-239.

Greiff M., *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*, 2017, MSc. Thesis, Lund University.

Huștiu S., Lupașcu M., Popescu Ș., Burlacu A., Kloetzer M., *Stable Hovering Architecture for Nanoquadcopter Applications in Indoor Environments*, 22nd International Conference on System Theory, Control and Computing (ICSTCC), IEEE, 2018, 659-663.

Huştiu S., Kloetzer M., Burlacu A., *Collision Free Path Planning for Unmanned Aerial Vehicles in Environments with Dynamic Obstacles*, 24th International Conference on System Theory, Control and Computing (ICSTCC), IEEE, 2020, 520-525.

Liu L., Shi R., Li S., Wu J., *Path Planning for UAVs Based on Improved Artificial Potential Field Mwthoid Through Changing the Repulsive Potential Function*, IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), 2016, 2011-2015.

Lupaşcu M., Huştiu S., Burlacu A., Kloetzer M., *Path Planning for Autonoumous Drones Using 3d Rectangular Cuboid Decomposition*, 23rd International Conference on System Theory, Control and Computing (ICSTCC), IEEE, 2019, 119-124.

Mahulea C., Kloetzer M., *Robot Planning Based on Boolean Specifications Using Petri Net Models*, IEEE Transactions on Automatic Control, 2018, **63**, 2218-2225.

Paranjape A., Meier K., Shi X., Chung S.-J, Hutchinson S., *Motion Primitive and 3D Path Planning for Fast Flight Though a Forest*, The International Journal of Robotics Research, 2015, **34**, 357-377.

Pastravanu O., Matcovschi MH., Mahulea C., *Petri Net Toolbox – Teaching Discrete Event Systems Under MATLAB*, Advances in Automatic Control, 2004, 247-255.

Tagay A., Omar A., Ali M.H., *Development of Control Algorithm for a Quadcopter*, Procedia Computer Science. 2021, **179**, 242-251.

Yao P., Wang H., Su Z., *Hybrid UAV Path Planning Based on Interfered Fluid Dynamical System and Improved RRT*, 41st Annual Conference of the IEEE Industrial Electronics Society (IECON), 2015, 829-834.

CERCETĂRI ASUPRA DEZVOLTĂRII UNEI TRAIECTORII FĂRĂ COLIZIUNE ÎNTR-UN SPAŢIU DE LUCRU 3D DINAMIC PENTRU UN UAV

(Rezumat)

Această lucrare ştiinţifică prezintă principalii paşi în vederea dezvoltării unei traiectorii fără coliziune pentru o dronă. Obiectivul principal este atingerea unei poziţii finale, într-un spaţiu de lucru cu obstacole dinamice. Primul deziderat este caracterizat de examinarea cinematicii şi a dinamicii unui quadcopter. Pentru acest scop, un model matematic neliniar a fost construit. Validarea modelului matematic a fost realizată în MATLAB prin intermediul simulărilor, pentru ca după să poată fi validat şi prin experimente reale. Al doilea obiectiv presupune maparea spaţiului de lucru 3D. Pentru acest obiectiv, a fost implementată o reprezentare bazată pe tehnica decompoziţiei celulare, extinsă din mediul 2D. Evaluarea algoritmului de planificare a traiectoriei s-a realizat prin diferite simulări numerice în diverse scenarii. La sfârşit, o comparaţie între 2 scenarii diferite este prezentată: pentru un spaţiu de lucru 3D doar cu obstacole statice şi pentru un spaţiu de lucru 3D dinamic, unde mişcarea obiectului dinamic este cunoscută.