aerospace

MDPI

Article

Stochastic Trajectory Generation Using Particle Swarm Optimization for Quadrotor Unmanned Aerial Vehicles (UAVs)

Babak Salamat * and Andrea M. Tonello

EcoSys Lab, Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria; andrea.tonello@aau.at

* Correspondence: babaksa@edu.aau.at; Tel.: +43-(0)-463-2700-3661

Academic Editors: David Anderson, Javaan Chahl and Michael Wing
Received: 22 March 2017; Accepted: 4 May 2017; Published: 8 May 2017

Abstract: The aim of this paper is to provide a realistic stochastic trajectory generation method for unmanned aerial vehicles that offers a tool for the emulation of trajectories in typical flight scenarios. Three scenarios are defined in this paper. The trajectories for these scenarios are implemented with quintic B-splines that grant smoothness in the second-order derivatives of Euler angles and accelerations. In order to tune the parameters of the quintic B-spline in the search space, a multi-objective optimization method called particle swarm optimization (PSO) is used. The proposed technique satisfies the constraints imposed by the configuration of the unmanned aerial vehicle (UAV). Further particular constraints can be introduced such as: obstacle avoidance, speed limitation, and actuator torque limitations due to the practical feasibility of the trajectories. Finally, the standard rapidly-exploring random tree (RRT*) algorithm, the standard (A*) algorithm and the genetic algorithm (GA) are simulated to make a comparison with the proposed algorithm in terms of execution time and effectiveness in finding the minimum length trajectory.

Keywords: stochastic trajectory; unmanned aerial vehicle (UAV); multi-objective optimization; obstacle avoidance; particle swarm optimization (PSO)

1. Introduction

In the last decade, unmanned aerial vehicles (UAVs), mostly known as an autonomous aerial vehicles, have been used in numerous military, aerial photography, agricultural and surveillance applications. UAVs can be classified into three significant groups: fixed-wing UAVs, rotary-wing UAVs and hybrid-layout UAVs [1].

The advantages of fixed-wing UAVs are the high-speed and the ability to fly for long distances. However, mechanical systems for landing and take-off, for example the landing gear, have to be installed. Furthermore, spacious structures, e.g., landing strips, have to be built. In small areas with obstacles, vertical take-off and landing (VTOL) and the ability to hover in a motionless spot are crucial. Therefore, rotary-wing UAVs have more applications. In addition, VTOL UAVs have greater maneuverability in indoor flight. Hybrid-layout UAVs have both long distance flight and VTOL properties. On the other hand, they have complex mechanisms for changing from rotary wing to fixed wing during the flight.

Considering harsh scenarios characterized by limited area [2], the presence of obstacles and possible dynamic constraints, rotary-wing UAVs have been shown to give the best performance [3]. This is why we consider these specific types in this paper. In particular, a quadrotor UAV is assumed. Its model structure is depicted in Figure 1. The aim is to derive a model to represent realistic trajectories that such a UAV can follow. In particular, there exist several methods to generate trajectories, including splines [4], rational functions [5], polynomial [6] and Bézier curves [7,8]. Likewise, another technique for trajectory generation is to use analytical functions such as higher-order piecewise polynomials to represent the evolution of the position over time [9]. Also, there are remarkable works in which a collision-free path can be obtained using the standard rapidly-exploring random tree (RRT*) and sampling-based algorithms [10,11]. In order to provide a method that can represent the variability of the ensemble of flight trajectories followed by a UAV, it is of interest to define a stochastic approach. In this respect, the Confined Area Random Aerial Trajectory Emulator (CARATE) proposed in [12] stochastically generates a 3D path obtained from a variable length previous history of the trajectory and a tunable set of random variables. CARATE is specifically designed to emulate stochastic trajectories with a limited flight area. However, it does not consider the dynamics of the UAV and it is not optimized to cope with harsh environmental constraints. In this paper, we tackle the limitations above and we propose a novel stochastic trajectory generation method that: (1) takes into account the dynamics of a quadrotor UAV, and (2) copes with environmental limitations. To do this, the dynamic model of a quadrotor [13,14] is considered. Moreover, an analytic function (quintic B-spline) is used to generate candidate trajectories. Nonetheless, the disadvantage of the deterministic functions is that they are not able to satisfy all the constraints at the same time when planing the trajectory in the search space. Therefore, the genetic algorithm (GA) may be used to obtain an optimized trajectory [15]. Another approach uses swarm intelligence. In particular, particle swarm optimization (PSO) is based on a simple mathematical model, extended by Kennedy and Eberhart in 1995 [16], to describe the social behavior of birds and fish. PSO follows the basic principle of self-organization which is used to characterize the dynamics of complex systems. PSO uses a model of social behavior to unravel the optimization problems, in a cooperative and sagacious skeleton [17]. PSO is easy to apply and is powerful in solving a diverse range of multi-objective optimization problems where GA can not be used. Some other related papers based on particle swarm optimization for path planning include [18,19]. Our approach, on the other hand, enjoys a specific method for solving the optimization problem with constraints. We change the problem into one that considers a cost metric that includes a violation function. The constraints, namely position, velocity, acceleration, Euler angles, torques and obstacle avoidance are added in the violation function.

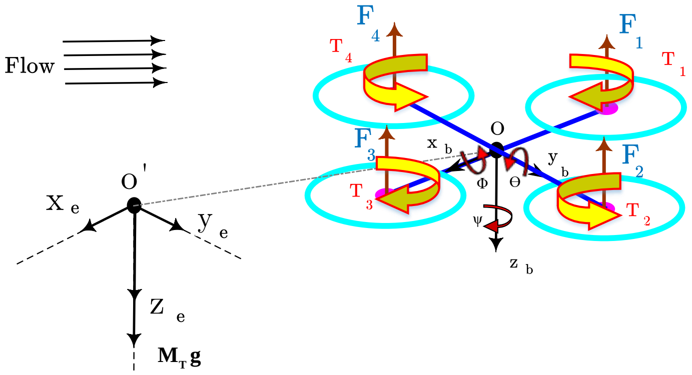

Figure 1. Reference frames of the unmanned aerial vehicle (UAV).

To validate the proposed method, three scenarios are defined:

1. Flight level (FL) trajectory;
2. Take-off → Mission → Landing (TML);
3. Complex maneuvering in 3D space (CMS).

In each scenario, random trajectories are generated under certain constraints that can be imposed with the given initial point (IP) and target point (TP) in three-dimensional coordinates.

We organize the contents of this paper as follows. Section 2 describes the world frame, body frame, dynamic model and the kinematics of the quadrotor. The trajectory modeling approach and the description of the three considered scenarios is given in Section 3. In Section 4, we define the cost function and the violation function related to the constraints and obstacle avoidance. Finally, examples of realized trajectories are reported in Section 5. The conclusions follow.

2. Dynamic Model

In order to represent the dynamic model of a quadrotor, two coordinate systems will be introduced. They describe the orientation and the motion of a quadrotor in the world frame and in the body frame. The world frame is referred to as the inertial earth coordinate $\acute{o}$ (north east down, NED) system. As can be observed in Figure 1, the earth is assumed to be fixed. Therefore, the inertial earth frame can be defined. The body coordinate o system is installed at the quadrotor's center of gravity (CoG). Each frame follows the right hand rule [20]. The notations used in this paper are reported in Table 1.

Table 1. List of notations.

Symbol	Definition
$\acute{o}(x_e, y_e, z_e)$	Earth axes
NED	North east down
$o(x_b, y_b, z_b)$	Body axes
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
Euler angles	$[\phi, \theta, \psi]$
$[\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$	Rate of change of Euler angles
$[p, q, r]^T$	Body angular rates
$M_T = M + M_r$	Overall mass of the UAV
I_{xx}, I_{yy}, I_{zz}	Moment of inertia
R	Radius of central sphere
g	Gravitational acceleration
F_i	Force generated by i_{th} rotor
T_i	Torque generated by i_{th} rotor
Ω_i	Angular velocity of i_{th} rotor
U_1, U_2, U_3, U_4	Control inputs

The Euler angles are used to represent the orientation of the quadrotor which is considered as a point mass, with respect to a fixed reference frame (body frame). The euler angle rates $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ can be represented in matrix form as follows [21]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{1}$$

The direction cosine matrix that allows to obtain a transform from the body frame to the world frame is denoted with C_{bn}. Hence, C_{bn} consists of three sequential single-axis rotations through the basic Euler angles $R(x, \phi)$, $R(y, \theta)$ and $R(z, \psi)$ that represent the roll, pitch and yaw, respectively. The direction cosine matrix C_{bn} can be obtained according to the relations below [22]:

$$R(x, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2}$$

$$R(y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3}$$

$$R(z,\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4}$$

$$C_{bn} = R(x,\phi)R(y,\theta)R(z,\psi)\,. \tag{5}$$

2.1. The Moment of Inertia

In this paper, the quadrotor is assumed to be a symmetric body. To define the moment of inertia, we assume the quadrotor to consist of four rotors and a main body. The main body is a sphere with radius R and a mass M. The four rotors are assumed to be a point mass M_r. Then, the point masses are connected to the center of the sphere with radius l. Therefore, the inertia matrix is a diagonal symmetric matrix:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{6}$$

with,

$$I_{xx} = I_{yy} = \frac{2MR^2}{5} + 2M_r l^2 \tag{7}$$

$$I_{zz} = \frac{2MR^2}{5} + 4M_r l^2\,. \tag{8}$$

2.2. Thrust and Torques

The translational position, velocity and acceleration of the point mass o represented in the inertial frame $\acute{o}$, are:

$$\xi = [x,y,z]^T \tag{9}$$

$$\dot{\xi} = [\dot{x},\dot{y},\dot{z}]^T \tag{10}$$

$$\ddot{\xi} = [\ddot{x},\ddot{y},\ddot{z}]^T\,. \tag{11}$$

According to Figure 1 the speed of each propeller can affect the motion of the quadrotor, so the relation between the thrust and torque generated by each rotor can be expressed as follows [13]:

$$F_i = b\dot{\Omega}_i^2, \quad for \quad i = 1,2,3,4 \tag{12}$$

$$T_i = d\dot{\Omega}_i^2, \quad for \quad i = 1,2,3,4\;, \tag{13}$$

while b and d are coefficients related to the Mach number, the Reynolds number and the angle of attack. Also, the input torques ς_i can be defined by:

$$\varsigma_i = (-1)^{i+1} d\dot{\Omega}_i^2, \quad for \quad i = 1,2,3,4 \tag{14}$$

2.3. Dynamic Model

The dynamic model of the quadrotor can be defined [13,14] according to the following relations:

$$\begin{cases} \ddot{x} = U_1(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \ddot{y} = U_1(\sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi)) \\ \ddot{z} = U_1(\cos(\theta)\cos(\psi)) - g \\ \ddot{\psi} = U_2 + \dot{\phi}\dot{\theta}I_1 \\ \ddot{\theta} = U_3 + \dot{\phi}\dot{\psi}I_2 \\ \ddot{\phi} = U_4 + \dot{\psi}\dot{\theta}I_3 \end{cases} \tag{15}$$

where,

$$\begin{cases} U_1 = F_1 + F_2 + F_3 + F_4 \\ U_2 = F_4 - F_2 \\ U_3 = F_3 - F_1 \\ U_4 = T_2 + T_4 - T_1 - T_3 \end{cases} \tag{16}$$

In Equation (15), I_1, I_2 and I_3 are equal to:

$$\begin{cases} I_1 = (I_{yy} - I_{zz})/I_{xx} \\ I_2 = (I_{zz} - I_{xx})/I_{yy} \\ I_3 = (I_{xx} - I_{yy})/I_{zz} \end{cases} \tag{17}$$

The inverse kinematics of our model that are referred to as control inputs of the quadrotor according to Equation (15) are:

$$\begin{cases} U_1 = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2} \\ U_2 = \ddot{\psi} - \dot{\phi}\dot{\theta}I_1 \\ U_3 = \ddot{\theta} - \dot{\phi}\dot{\psi}I_2 \\ U_4 = \ddot{\phi} - \dot{\psi}\dot{\theta}I_3 \end{cases} \tag{18}$$

Indeed, the quadrotor has four motors actuating the four rotors while our system has six degrees of freedom. It can be deduced that our system is underactuated. In order to resolve the issue, we have to identify the dependency in Equation (15), as proposed by [23]:

$$\begin{cases} \ddot{x}\cos(\phi) - (\ddot{z} + g)\tan(\theta) = 0 \\ \ddot{x}\sin(\phi) - \ddot{y}\cos(\phi) - (\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2})\sin(\psi) = 0\,. \end{cases} \tag{19}$$

Equation (19) shows two nonholonomic constraints existing in the kinematics of the quadrotor. Therefore, the angles θ and ψ can be derived from these nonlinear equations:

$$\begin{cases} \theta = \text{atan2}(\ddot{x}\cos(\phi) + \ddot{y}\sin(\phi), \ddot{z} + g) \\ \psi = \arcsin(\frac{\ddot{x}\sin(\phi) - \ddot{y}\cos(\phi)}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2}}) \end{cases} \tag{20}$$

For calculating the torques, Equation (17) can be rearranged in matrix form so that:

$$U = \Xi\,\Omega \tag{21}$$

where Ξ is:

$$\Xi = \begin{bmatrix} b & b & b & b \\ 0 & -bl & 0 & bl \\ -bl & 0 & bl & 0 \\ -bl & bl & -bl & bl \end{bmatrix}. \tag{22}$$

The input torques U_i can be obtained by the actuators of the quadrotor. Exploiting Equation (20), we can define the fitness function and boundary conditions in our model by evaluating $\Gamma(t) = (x(t), y(t), z(t), \phi(t))$, for $t \in [0\,T]$ and its time derivatives. We refer to $\Gamma(t)$ as the trajectory evolution.

3. Modeling of Three Representative Scenarios

Our goal is to develop a stochastic model for the trajectory evolution. Indeed, at a first glance this can appear a difficult task since the UAV can follow a number of different trajectories and be in various flying situations. Therefore, we propose to consider certain scenarios that can be representative of certain flying operations. In each scenario, the trajectory evolution $\Gamma(t)$ is represented with a B-spline drawn from a certain random model. In more detail (Figure 2), the B-spline has order k and it is defined by $n+1$ control points [4]. In our model, the first and the last control point are fixed and they correspond to the start and the end of the mission, respectively. In the search space between these two points, additional control points are placed in order to generate the segments of the B-spline curve. Each one of the control points is defined with coordinates in the search space, which are $\{x(t), y(t), z(t), \phi(t)\}$ and each of them is generated stochastically as a uniform random variable ranging between a lower and an upper limit that are distributed over time by the user. This configuration enables a stochastic behaviour for $\Gamma(t)$. Indeed, by changing the location of the control points in the search space, a new path can be realized.

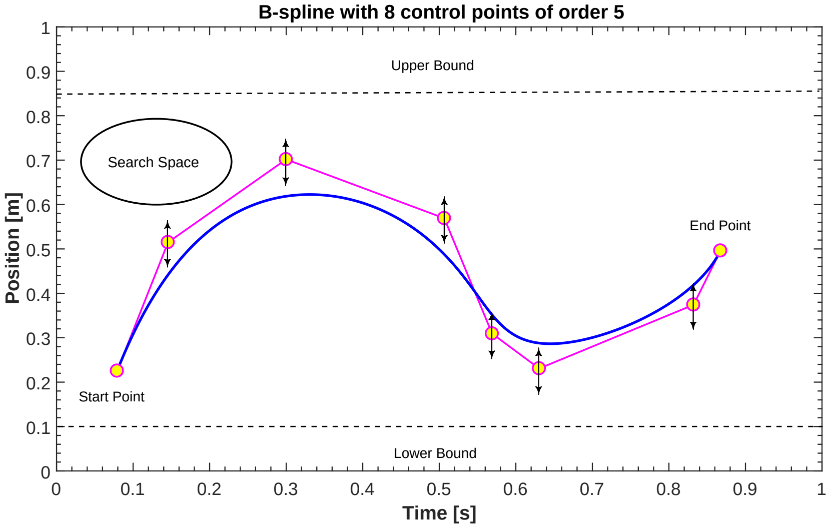

Figure 2. Evolution of the B-spline.

The mathematical model for a B-spline curve of degree k is defined by:

$$S(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i \tag{23}$$

where $(P_0, P_1, ...P_n)$ are control points and $N_{i,k}(t)$ are the basis functions defined using the Cox-de Boor recursion formula [4]:

$$N_{i,0}(t) = \begin{cases} 1, & \text{if } t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$
$$N_{i,j}(t) = \frac{t - t_i}{t_{i+1} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t) \ . \tag{24}$$

The reason for using the B-spline is that unlike the global property of the Bezier curve, it has the local property [24]. Furthermore, the order of the B-spline is very important to generate feasible trajectories that fulfill constraints and obtain trajectory continuity and smoothness. To do so, we use fourth-order B-splines (quadratic) and fifth-order B-splines (quintic) in the considered scenarios.

3.1. Scenario One (FL)

According to the aeronautical literature [25], the flight level (FL) is expressed as a vertical altitude with zero roll, pitch and yaw at sea level pressure. For this scenario a fourth-order B-spline with four control points is considered. More details on the FL can be found in Section 4.

3.2. Scenario Two (TML)

In all kind of UAVs, take-off, mission and landing (TML) are crucial. The challenge in modeling this scenario is the tuning of the parameters of the B-spline in a way that: (1) the velocity profiles of the trajectories during take-off and landing are zero (rest-to-rest manoeuvre); (2) the absolute value of the velocity should be less than 1.5 m/s; (3) the Euler angles during take-off and landing are zero (rest-to-rest manoeuvre); and (4) the quadrotor reaches the specific altitude (10 m) during the flight time. Clearly, other constraints can be defined. For simplicity, the surface of the ground is assumed to be horizontal. The length and duration of the trajectory between take off and landing can be made variable. Further details about modeling this scenario are reported in Section 4 and numerical examples in Section 5.

3.3. Scenario Three (CMS)

The third considered scenario highlights the ability of our approach to model trajectories that comprise obstacle avoidance procedures. Static obstacles with different size can be defined in the flight space. In particular, we consider three obstacles. Finding a safe and short trajectory in the search space with obstacles becomes a challenging problem in path planning. On the other hand, according to the configuration of our quadrotor, the feasibility of these trajectories is also significant. Boundary constraints for complex maneuver in 3D space are defined in Equation (25). More details are reported in Section 5.

4. Trajectory Realization under Practical Constraints

As discussed previously, the trajectory evolution is $\Gamma(t) = (x(t), y(t), z(t), \phi(t))$ with input controls $\Lambda(t) = (\varsigma_1(t), \varsigma_2(t), \varsigma_3(t), \varsigma_4(t))$. Also, the dynamics of a quadrotor and environmental constraints add mathematical constraints to the generation of feasible quadrotor trajectories. In the following discussion, the constraints will be defined. Therefore, our problem is to determine the optimal coefficients of the B-spline under particular constraints. The boundary constraints in our scenarios are:

$$\begin{cases} \xi(0) = \Gamma(start) & and \quad \xi(T) = \xi(end) \\ \dot{\xi}(0) = 0 & and \quad \dot{\xi}(T) = 0 \\ \ddot{\xi}(0) = 0 & and \quad \ddot{\xi}(T) = 0 \\ \Lambda_i^{min} \;\leq\; \Lambda_i^t \;\leq\; \Lambda_i^{max} & \\ d_{min} = min(d) & \end{cases} \tag{25}$$

where d_{min} is the minimum Euclidian distance between the UAV and a given a generic obstacle. It means that for $d_{min} \geq 0$, the UAV will not crash during the flight. We solve this problem with the PSO algorithm in Section 4.2.

4.1. Design of Trajectory with Minimum Length

The aim of this paper is generating stochastic trajectories. However, in path planning when the search space is populated with generic obstacles, finding the short and safe path among these trajectories becomes significant. We can therefore consider the problem of designing the shortest path trajectory. This translates into the definition of a cost function that equals the length of the path as follows:

$$J_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{26}$$

where we have discretized the time evolution so that the coordinates at time instant i are denoted with $x_i = x(iT), y_i = y(iT), z_i = z(iT)$ and the sampling period is T. Our problem is a multi-objective optimization problem that has a cost function with boundary constraints. To proceed, we write it as a problem without constraints, however it includes a violation function $P(\Upsilon)$ to obtain the new cost function.

$$\hat{J} = \frac{1}{N}(\sum_{i=1}^{N} J_i + \sum_{i=1}^{N} \sum_{\Upsilon=1}^{M} P_i(\Upsilon)) \tag{27}$$

where N denotes the total number of samples considered in the overall trajectory. i represents the time index. J is the cost function that is equal to the length of the path. $P_i(\Upsilon)$ is the violation at time i for constraint Υ. The constraints Υ are the position, velocity, acceleration, Euler angles, torques and obstacle avoidance. Two important properties for the violation function are $P(0) = 0$ and $P(\Upsilon) \geq 0$. In order to satisfy the boundary constraints in Equation (25), the violation function for less, more and equal constraints can be defined respectively as follows:

$$P_i(\Upsilon) = \begin{cases} \max(\frac{f(\Upsilon)}{f_0(\Upsilon)} - 1, 0) & \text{for} \quad f(\Upsilon) \leq f_0(\Upsilon) \\ \max(1 - \frac{f(\Upsilon)}{f_0(\Upsilon)}, 0) & \text{for} \quad f(\Upsilon) \geq f_0(\Upsilon) \\ \mid (\frac{f(\Upsilon)}{f_0(\Upsilon)} - 1) \mid & \text{for} \quad f(\Upsilon) = f_0(\Upsilon) \end{cases} \tag{28}$$

where Υ refers to a certain constraint as already explained. In addition, $f(\Upsilon)$ and $f_0(\Upsilon)$ represent the physical quantity and target constraint respectively. Now with the violation functions definition in Equation (28), the boundary conditions in Equation (25) can be satisfied. For obstacle avoidance, the Euclidian distance between the edge of the object and the quadrotor is used [26]. The whole algorithm is sketched in Figure 3.

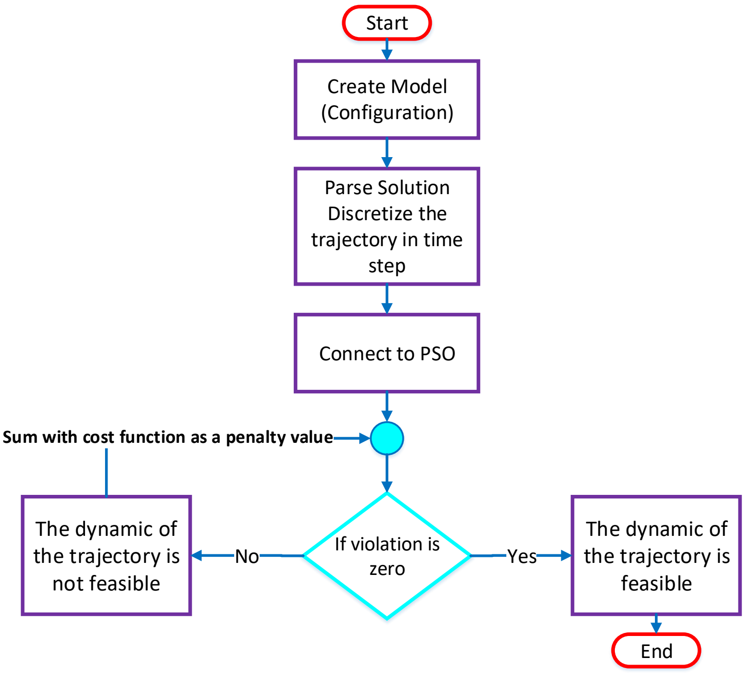
Start
Create Model (Configuration)
Parse Solution Discretize the trajectory in time step
Connect to PSO
Sum with cost function as a penalty value
The dynamic of the trajectory is not feasible
No
If violation is zero
Yes
The dynamic of the trajectory is feasible
End

Figure 3. Flowchart of trajectory generation algorithm. PSO: particle swarm optimization.

4.2. Basics of PSO

Particle swarm optimization (PSO) is an intelligent optimization algorithm. It belongs to the class of algorithms called metaheuristics. PSO is based on the paradigm of swarm intelligence and it is inspired by social behavior of animals like fish and birds [16]. PSO is a simple yet powerful optimization algorithm and it is successfully applied to multiple applications in various fields of science and engineering like machine learning, mesh processing, data mining and many other fields.

PSO contains a population of candidate solutions called a swarm. According to Figure 4, every particle $p(n)$ is a candidate solution to the optimization problem. Any particle has a position in the search space of the optimization problem. The search space is the set of all possible solutions to the optimization problem. PSO allows for efficiently finding an optimal solution under a target function with multiple constraints in the problem at hand.

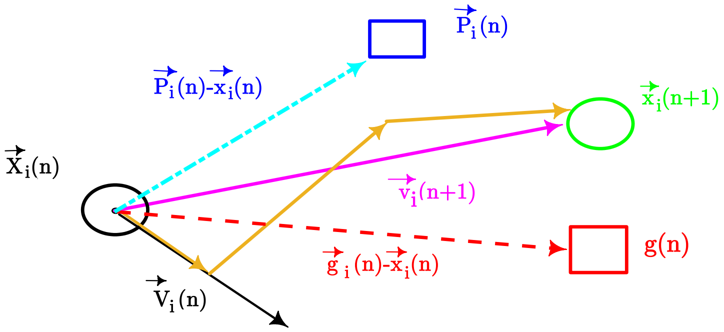

Figure 4. Mathematical model of PSO.

To proceed, we consider particles for the position vector $\mathbf{x}$ and the velocity vector $\mathbf{v}$. Therefore, the updating realization for the particles includes [16]:

$$\mathbf{x}_{i+1}(n) = \mathbf{x}_i(n) + \mathbf{v}_i(n+1) \tag{29}$$

$$\mathbf{v}_{i+1}(n) = \omega \mathbf{v}_i(n) + r_1 c_1(\mathbf{p}_i(n) - \mathbf{x}_i(n)) + r_2 c_2(\mathbf{g}_i - \mathbf{x}_i(n)) \tag{30}$$

where i denotes the time index. $n = \{1, ..., L\}$ is the particle index, and L is the number of considered particles. ω, c_1 and c_2 are real valued coefficients. c_1 is the personal learning coefficient and c_2 is the global learning coefficient. w is an inertial coefficient. c_1 and c_2 are acceleration coefficients. On the other hand, r_1, r_2 are random numbers uniformly distributed in the range from zero to one $(r_1, r_2 \sim U(0,1))$. $\mathbf{p_i}(n)$ is the best personal position of the particle of index n at time i, while $\mathbf{g}_i$ is the best collective position of all particles at time i in the swarm. It should be noted that in the original version of PSO proposed in 1995, there was no inertia term. However, in 1998 Shi and Eberhart added the inertia term to the standard PSO [27].

5. Numerical Generator and Results

In this section, first we report a summary of example parameters to be used to generate trajectories in the three considered scenarios. The proposed algorithm can be used for any configuration of UAVs, but herein, we focus on a quadrotor. A number of parameters is initially fixed as shown in Table 2 [13]. Furthermore, in each flight scenario the start point, end point and a number of control points are defined by the user. We also report the number of particles and the number of iterations run by the PSO for the generation of trajectories with a certain number of control points. The details for each scenario are given in Table 3. The number of iterations listed in Table 3 to obtain the minimum length path are needed for the convergence. The average cost metric as a function of the iterations will be shown in Sections 5.1–5.3 for Scenarios 1, 2 and 3, respectively. The execution time has been computed for a simulation setup with an Intel i5 CPU with 3.3 GHz, and RAM of 4GB using MATLAB 2016b. Finally, we demonstrate the performance of our method in comparison with the standard (A*) algorithm, the standard rapidly-exploring random tree (RRT*) algorithm and the genetic algorithm (GA) for obtaining the minimum length path in 2D.

Table 2. Physical parameters of the quadrotor.

Symbol	Unit
M_T	0.65 Kg
l	0.232 m
g	9.806 $\mathrm{m/s^2}$
R	0.15 m
I_{xx}	0.07582 $\mathrm{Kg\,m^2}$
I_{yy}	0.07582 $\mathrm{Kg\,m^2}$
I_{zz}	0.1457924 $\mathrm{Kg\,m^2}$
b	3.13×10^{-5} $\mathrm{Ns^2}$
d	7.5×10^{-7} $\mathrm{Nms^2}$
$\Lambda^{max} = -\Lambda^{min}$	0.04 Nm
$\phi^{max} = -\phi^{min}$	0.1 Rad
$\theta^{max} = -\theta^{min}$	0.1 Rad
$\psi^{max} = -\psi^{min}$	0.1 Rad
Inertia Weight (ω)	1
c_1	2
c_2	2

Table 3. Solution details.

Scenarios	Control Points	Swarm Size	Iterations	Execution Time
Scenario 1	4	50	250	5.090549 (s)
Scenario 2	5	100	400	6.572763 (s)
Scenario 3	5	100	450	7.464875 (s)

5.1. Scenario 1 (FL)

Figure 5 shows a number of realizations of possible trajectories generated by the algorithm. Among these trajectories one is chosen based on the constraints set in Equation (25) in order to isolate the one with least distance. The initial and final position can be chosen by the user. In this specific example the initial position of our quadrotor is $\Gamma_{initial} = (0, 0, 25, 0)$ in meters. The final position is $\Gamma_{final} = (20, 0, 25, 0)$. The fourth-order B-spline is considered for this scenario. Figure 6 shows the minimum length trajectory that is obtained from the PSO. Furthermore, the details about positions, actuators torques, control efforts, translational velocities and the acceleration obtained with the PSO are given in Figures 7–10. Figure 11 shows the average cost metric (Equation (27)) as a function of iterations. The convergence is achieved in about 250 iterations. PSO algorithm converges quickly to the optimal solution. The minimum length of the trajectory for this scenario obtained via simulation is $J = 20$ m and the total flight time is 20 s.

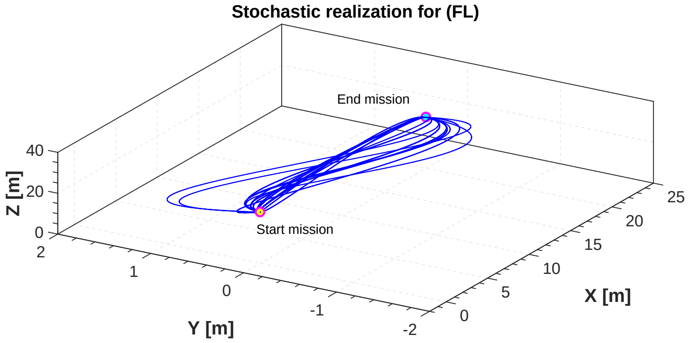

Figure 5. Stochastic realization for flight level (FL).

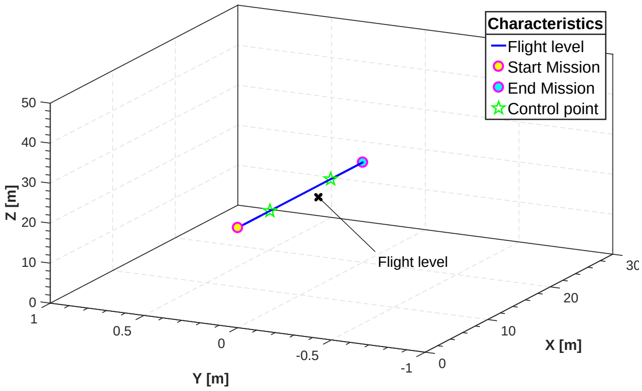

Figure 6. Three-dimensional flight level (FL) trajectory.

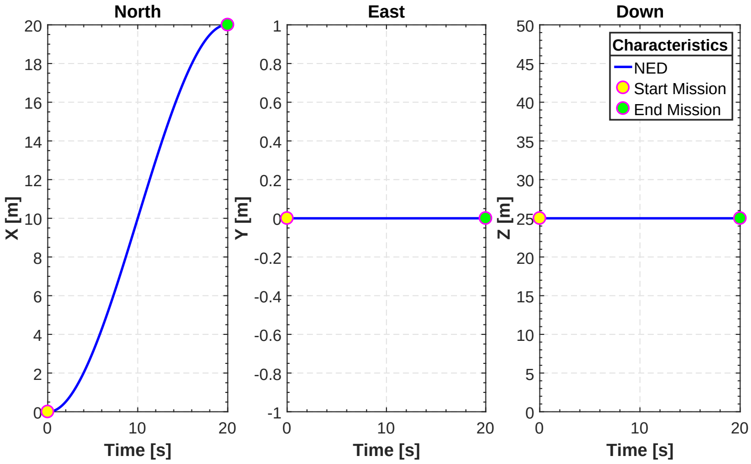

Figure 7. North east down (NED) for FL.

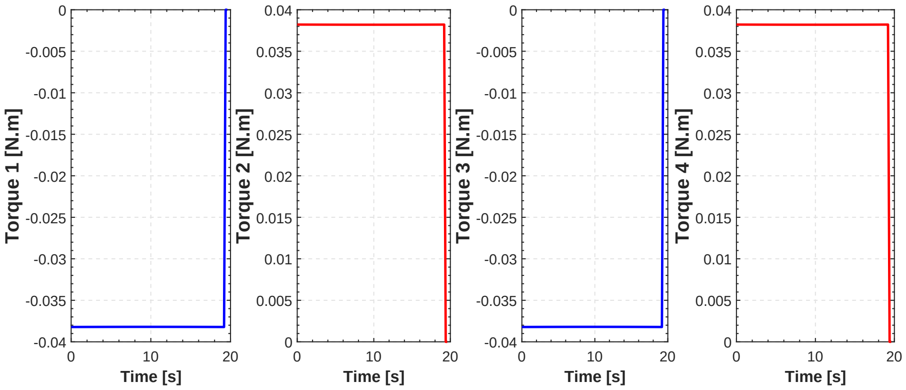

Figure 8. Evolution of actuators torques for FL.

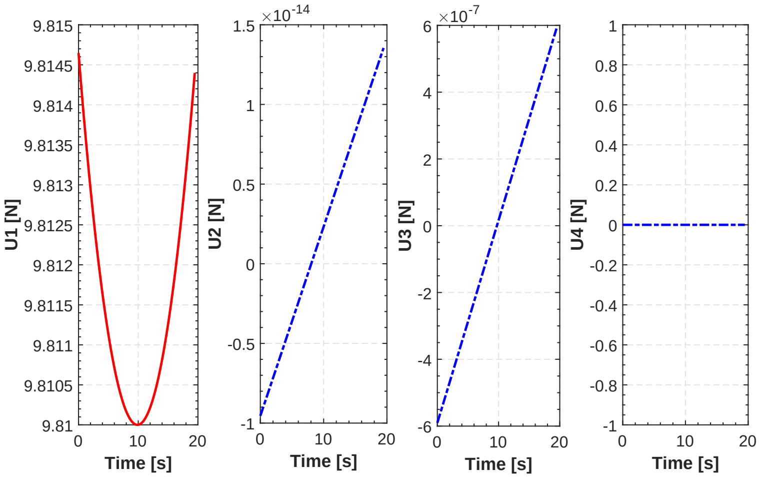

Figure 9. Control efforts for FL.

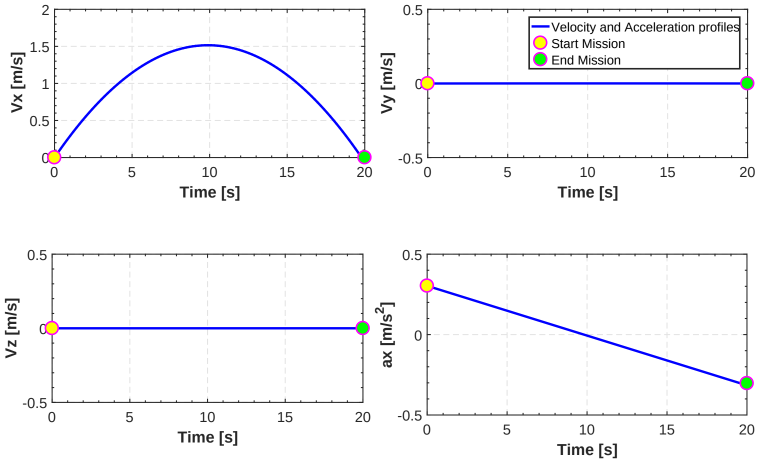

Figure 10. Translation of velocities and acceleration in FL.

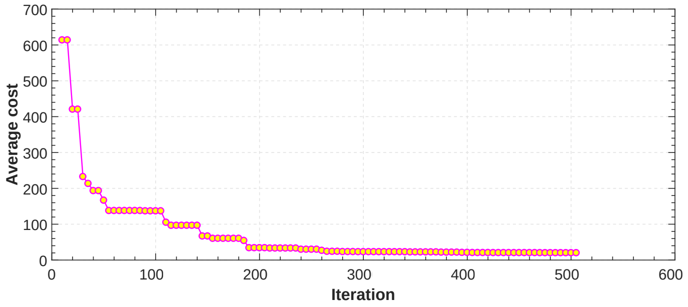

Figure 11. Average cost function vs. iteration for FL.

5.2. Scenario 2 (TML)

In Figures 12–16, we report an example of trajectory belonging to the Scenario 2 (TML). In particular, the starting point is $\Gamma_{initial} = (0, 0, 0, 0)$ and the landing point is $\Gamma_{landing} = (8, 6, 0, 0)$. The maximum altitude during the flight time is 10 m. Also, Figure 17 shows the minimum length trajectory ($J = 33.9608$ m) which is obtained after 100 iterations of the PSO algorithm. Figure 15 represents the inertial frame for the velocity components, which are less than 1.5 m/s according to our constraints definition in scenario two and the total flight time is 23 s.

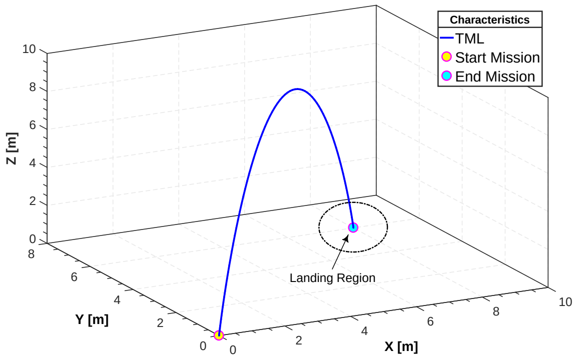

Figure 12. Three-dimensional take-off, mission and landing (TML) trajectory.

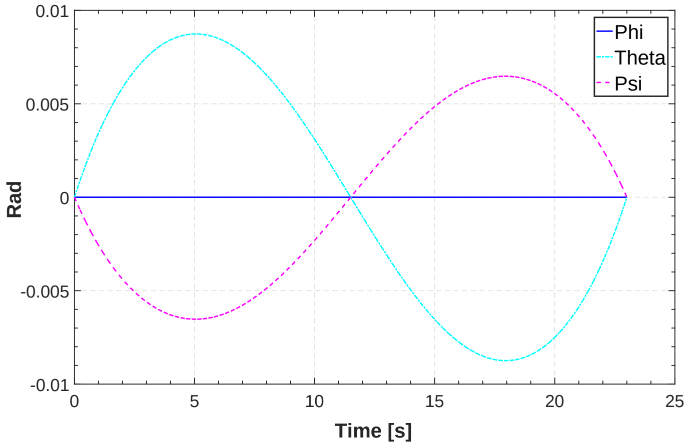

Figure 13. Roll, pitch and yaw angle for TML.

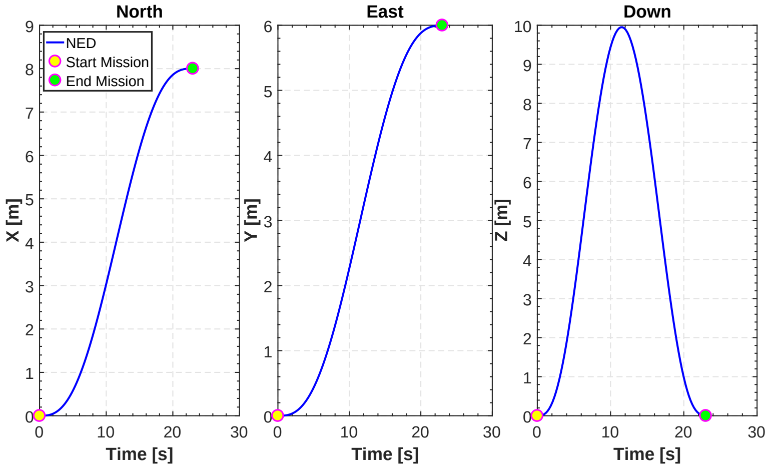

Figure 14. North east down NED for TML.

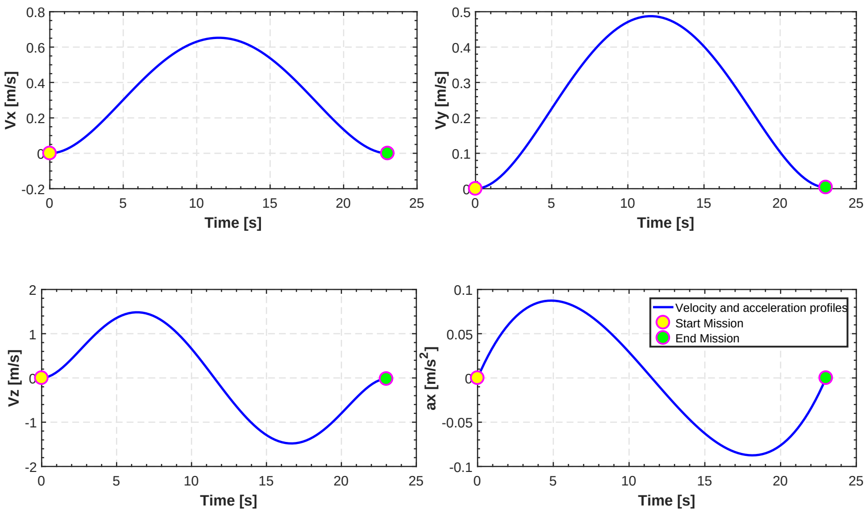

Figure 15. Translation of velocities and acceleration for TML.

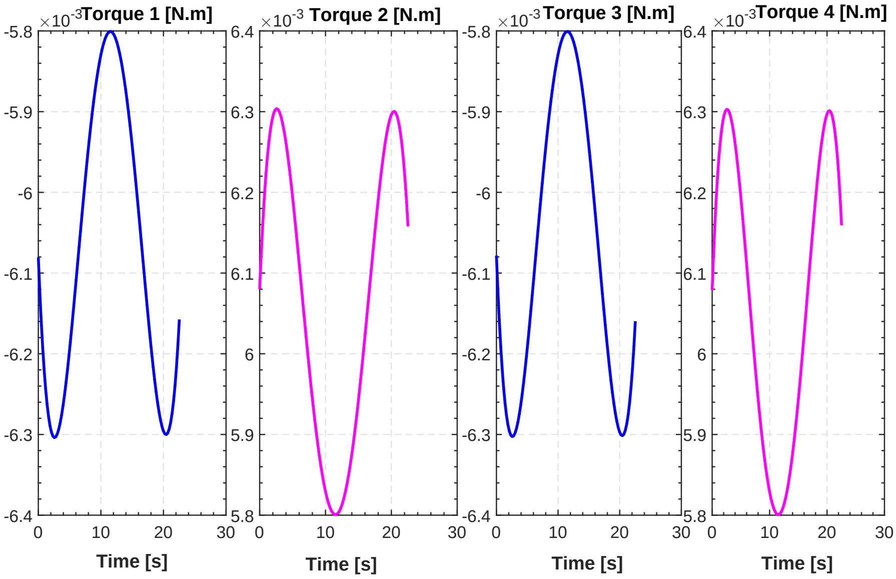

Figure 16. Evolution of actuators torques for TML.

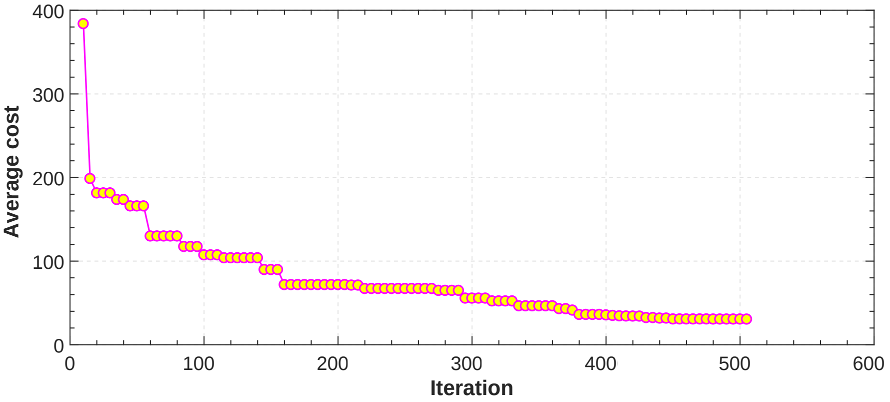

Figure 17. Average cost function vs. iteration for TML.

5.3. Scenario 3 (CMS)

In Figure 18, the search space is populated with three generic spherical objects that act as obstacles for the quadrotor and thus hinder their path. These obstacles can be either positioned arbitrarily by the user or randomly generated by the algorithm. In this specific case they were generated by the user. For CMS, a stochastic realization as a set of trajectories is shown in Figure 18. The start point is $\Gamma_{initial} = (0, 0, 0, 0)$ and the end point is $\Gamma_{final} = (8, 6, 0, 0)$. Simulation results for the cost function and the actuator torques are shown in Figures 19 and 20. Also, the minimum path after 150 iterations and the total flight time are $J = 23.495$, and 23 s, respectively. Furthermore, the minimum length path obtained with PSO and time histories of the attitude angle of the quadrotor are shown in Figures 21 and 22, respectively.

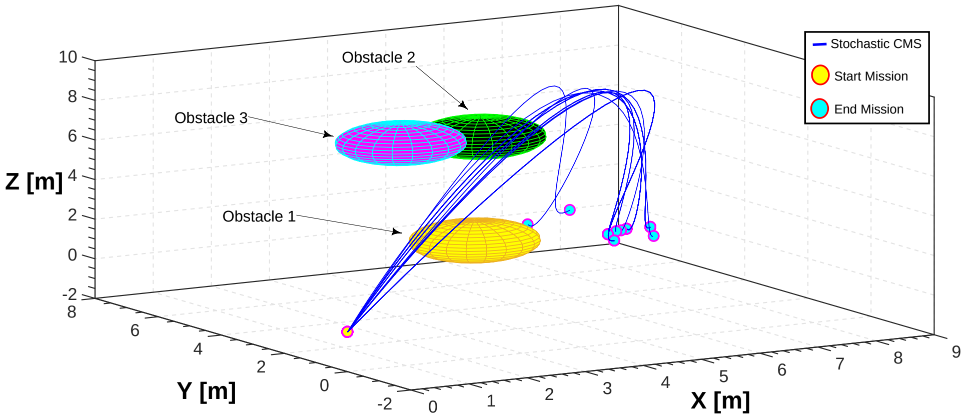

Figure 18. Stochastic complex maneuvering in 3D space (CMS).

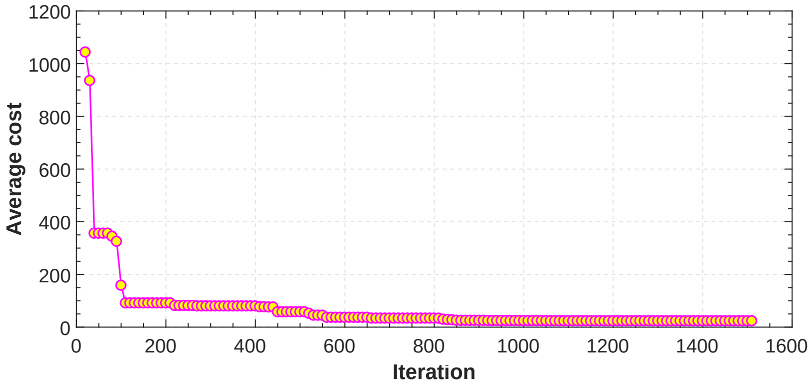

Figure 19. Average cost function vs. iteration for CMS.

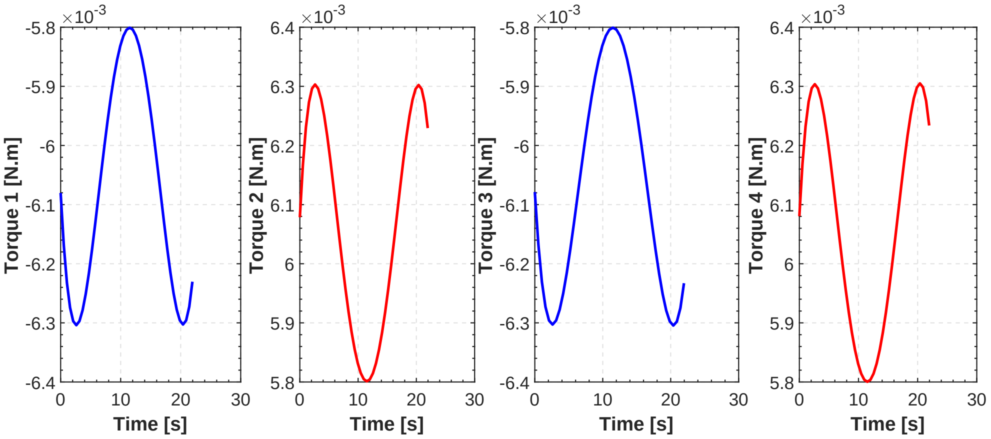

Figure 20. Evolution of actuator torques for CMS.

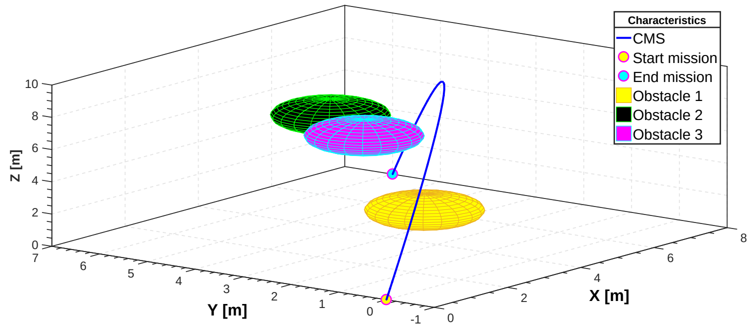

Figure 21. Complex maneuvering in 3D space (CMS).

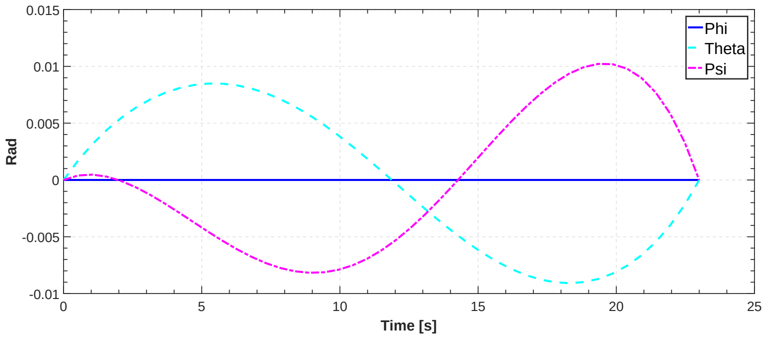

Figure 22. Roll, pitch and yaw angle for CMS.

5.4. PSO vs. A, RRT* and GA*

As already mentioned before, to illustrate the effectiveness of the proposed method, we consider three different methods named the RRT* algorithm, the A* algorithm and the GA to obtain the minimum path in 2D that obeys several constraints. The search space is also surrounded by four obstacles. As shown in Table 4 and Figure 23, the results for the proposed method show that it has a faster convergence time and it finds a path with smaller length and collision avoidance. The optimum length for our method is 14.2095 m. Also, it is noticeable that the execution-time for the A* algorithm is 0.02315 s. The execution time does not consider the part for the construction of the B-spline curves.

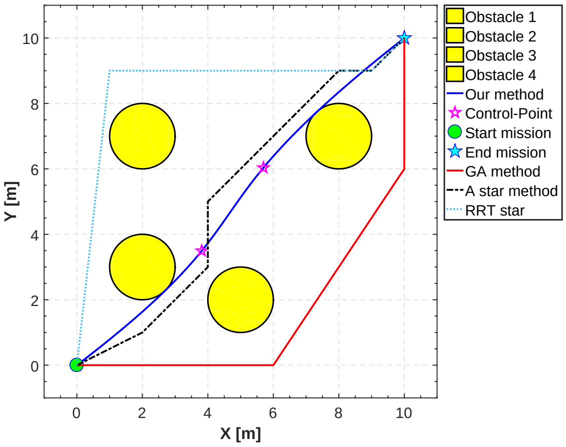

Figure 23. Optimal path by rapidly-exploring random tree (RRT*), standard (A*) algorithm, genetic algorithm (GA) and the PSO.

Table 4. Comparision Results.

Parameters	RRT*	A*	GA	PSO + B-Spline
Length (m)	18.1265	16.5563	17.2111	14.2095
Execution time (s)	17.343599	0.02315	9.230695	1.544244

6. Conclusions

In this paper, a new methodology has been proposed for stochastic modeling and generating trajectories followed by a UAV. The approach is based, firstly, on modeling a certain flight scenario. Then, the evolution of the trajectory is obtained using a B-spline with a certain order and setting some control points. A PSO multi-objective optimization algorithm is used to not only tune the parameters of the B-spline function, but also to guarantee that the trajectories are dynamically feasible. The trajectories can also model an obstacle avoidance situation. This has been realized by defining a violation function to avoid collision during the flight time with obstacles at certain positions. A numerical simulator has been implemented to emulate the reference scenarios. The simulator can be configured to allow variability and representativity of the flight scenario. Several trajectory examples have been shown to verify the trajectory smoothness and fulfillment of practical constraints introduced by the UAV dynamics.

Author Contributions: Babak Salamat developed the UAV trajectory optimization procedure under feasibility constraints, obtained the numerical results and prepared the paper. Andrea M. Tonello conceived the overall UAV random trajectory generation concept, supervised the development of the model, results and paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Austin, R. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*, 1st ed.; Wiley: Chichester, UK, 2010.
2. Majka, A. Trajectory Management of the Unmanned Aircraft System (UAS) in Emergency Situation. *Aerospace* **2015**, *2*, 222–234.
3. Suicmez, E.C.; Kutay, A.T. Optimal path tracking control of a quadrotor UAV. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 115–125.
4. Sprunk, C. *Planning Motion Trajectories for Mobile Robots Using Splines*; Student Project; University of Freiburg: Freiburg, Germany, 2008.
5. Montés, N.; Mora, M.C.; Tornero, J. Trajectory Generation based on Rational Bezier Curves as Clothoids. In Proceedings of the IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 505–510.
6. Babel, L. Three-dimensional Route Planning for Unmanned Aerial Vehicles in a Risk Environment. *J. Intell. Robot. Syst.* **2013**, *71*, 255–269.
7. Jayasinghe, J.A.S.; Athauda, A.M.B.G.D.A. Smooth trajectory generation algorithm for an unmanned aerial vehicle (UAV) under dynamic constraints: Using a quadratic Bezier curve for collision avoidance. In Proceedings of the 2016 Manufacturing Industrial Engineering Symposium (MIES), Colombo, Sri Lanka, 22 October 2016; pp. 1–6.
8. Tsai, Y.J.; Lee, C.S.; Lin, C.L.; Huang, C.H. Development of Flight Path Planning for Multirotor Aerial Vehicles. *Aerospace* **2015**, *2*, 171–188.
9. Cruz, A.B.; Montaño, J.C.; Mier, P.G. TG2M: Trajectory Generator and Guidance Module for the Aerial Vehicle Control Language AVCL. In Proceedings of the 40th International Symposium on Robotics, Barcelona, Spain, 10–14 March 2009.
10. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res. (IJRR)* **2011**, *30*, 846–894.
11. Barbehenn, M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Trans. Comput.* **1998**, *47*, 263.

12. Papaiz, A.; Tonello, A. Azimuth and Elevation Dynamic Tracking of UAVs via 3-Axial ULA and Particle Filtering. *Int. J. Aerosp. Eng.* **2016**, *2016*, 1–9.
13. Sanca, A.S.; Alsina, P.J.; Cerqueira, J.J.F. Dynamic Modelling of a Quadrotor Aerial Vehicle with Nonlinear Inputs. In Proceedings of the 2008 IEEE Latin American Robotic Symposium, Salvador, Bahia, Brazil, 29–30 October 2008 ; pp. 143–148.
14. Mohammadi, M.; Shahri, A.M. Modelling and decentralized adaptive tracking control of a quadrotor UAV. In Proceedings of 2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, 13–15 Febuary 2013.
15. Galvez, R.L.; Dadios, E.P.; Bandala, A.A. Path planning for quadrotor UAV using genetic algorithm. In Proceedings of the 2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Puerto Prinsesa, Palawan, Philippines, 12–16 November 2014; pp. 1–6.
16. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995 (MHS '95), Nagoya, Japan, 4–6 October 1995.
17. Kennedy, J. The particle swarm: Social adaptation of knowledge. In Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, IN, USA, 13–16 April 1997.
18. Kang, H.I.; Lee, B.; Kim, K. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm. In Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 19–20 December 2008; Volume 2, pp. 1002–1004.
19. Wang, Q.; Zhang, A.; Qi, L. Three-dimensional path planning for UAV based on improved PSO algorithm. In Proceedings of the 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, China, 31 May–2 June 2014; pp. 3981–3985.
20. Cook, M.V. *Flight Dynamics Principles*; Elsevier: Amsterdam, The Netherlands, 2007.
21. Tang, Y.R.; Li, Y. Dynamic modeling for high-performance controller design of a UAV quadrotor. In Proceedings of the 2015 IEEE International Conference on Information and Automation, Lijiang, China, 8–10 August 2015; pp. 3112–3117.
22. Nemra, A.; Aouf, N. Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering. *IEEE Sens. J.* **2010**, *10*, 789–798.
23. Jolly, A.C. *Trajectory Generation by Piecewise Spline Interpolation*; Techinical Report; U.S. Army Missile Command: Redstone Arsenal, AL, USA, 1976.
24. Prautzsch, H.; Boehm, W.; Paluszny, M. *Bézier and B-Spline Techniques*; Springer: Berlin/Heidelberg, Germany; New York, NY, USA, 2002.
25. Eshelby, M.E. *Aircraft Performance*; Elsevier: Oxford, UK, 2000.
26. Jamieson, J.; Biggs, J. Path Planning Using Concatenated Analytically-Defined Trajectories for Quadrotor UAVs. *Aerospace* **2015**, *2*, 155.
27. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.