J Intell Robot Syst (2010) 57:123–141
DOI 10.1007/s10846-009-9365-3

On the Generation of Trajectories for Multiple UAVs in Environments with Obstacles

Armando Alves Neto · Douglas G. Macharet · Mario F. M. Campos

Received: 1 February 2009 / Accepted: 1 August 2009 / Published online: 28 August 2009

Abstract This paper presents a methodology based on a variation of the Rapidly-exploring Random Trees (RRTs) that generates feasible trajectories for a team of autonomous aerial vehicles with holonomic constraints in environments with obstacles. Our approach uses Pythagorean Hodograph (PH) curves to connect vertices of the tree, which makes it possible to generate paths for which the main kinematic constraints of the vehicle are not violated. These paths are converted into trajectories based on feasible speed profiles of the robot. The smoothness of the acceleration profile of the vehicle is indirectly guaranteed between two vertices of the RRT tree. The proposed algorithm provides fast convergence to the final trajectory. We still utilize the properties of the RRT to avoid collisions with static, environment bound obstacles and dynamic obstacles, such as other vehicles in the multi-vehicle planning scenario. We show results for a set of small unmanned aerial vehicles in environments with different configurations.

Keywords Multiple UAV trajectory planning · Rapidly-exploring Random Trees · Pythagorean Hodograph curves · UAV swarm

This work was developed with the support of CNPq, CAPES and FAPEMIG.

A. Alves Neto (✉) · D. G. Macharet · M. F. M. Campos
Computer Vision and Robotic Laboratory (VeRlab), Computer Science Department, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil
e-mail: aaneto@dcc.ufmg.br

D. G. Macharet
e-mail: doug@dcc.ufmg.br

M. F. M. Campos
e-mail: mario@dcc.ufmg.br

1 Introduction

Several open issues still demand much research effort in order to endow a vehicle with full autonomy in a real environment. Among those, one well known problem is posed by one of mobile robotics basic questions related to navigation: "How do I get to a given goal?" Throughout the years, several efficient and relevant solutions have been proposed with varying degrees of success. Normally, given some knowledge of the environment, such as position of the obstacles and of others vehicles, it is possible to plan a path that allows the vehicle to safely achieve this goal.

A fundamental feature of a path planning algorithm is to ensure that the vehicle will be able to execute this path, which means that during the trajectory generation, the movements restrictions of the vehicle must be considered (i.e. nonholonomic constraints). For example, a typical car cannot move laterally, so the radius of curvature is one of the restrictions imposed on trajectories generated for cars.

It is important to remark, however, that trajectory planning and path planning are two different tasks. Path planning deals with the position of a continuous set of points that represents a collision free path through the environment. A trajectory is parameterized by time, somehow embedding the body dynamic constraints. Therefore, different vehicles can be at the same position, but in a different time.

Several strategies to obtain a path between two different positions can be found in the literature [1]. Among the simplest and most common are the visibility graph, cell decomposition and potential field path planners. However, one of the problems with those techniques is that they guarantee obstacle free path between two points with no consideration as to the vehicle's capacity to execute follow the generated path (e.g. negotiate a curve between obstacles).

In the robotics literature, most of path planners were designed for manipulators and ground vehicles moving in a two dimensional environment. Nowadays, the kinds of mobile robots abound, each of them possessing different locomotion particularities as well as different types of motion constraints. The choice of the type of vehicle to be used is determined by the mission it carry out and the environment thereof.

The interest and research in Unmanned Aerial Vehicles (UAVs) has been increasing growing, specially due to the decrease in cost, weight, size and performance of sensors and processors. Clearly UAVs have their niche of applications, which is cannot be occupied by other types of mobile robots, as such as they are capable of covering a broad set of relevant applications. They are able to navigate over large areas obviously faster than land vehicles, with a privileged view from above, which is one of their main uses in monitoring and surveillance. As the availability increases, so does the possibility of having multiple such vehicles traversing a given volume of the space. Therefore, there is growing need to study and develop techniques for the generation of safe and feasible trajectories considering the specific constraints of different types of aerial robots.

UAVs can be divided into at least three classes: rotary-wing aircrafts (e.g. helicopters and quadrotors), aerostatic aircrafts (such as airships and hot air balloons) and fixed-wing aircrafts (airplanes). The technique described in this text will be instantiated for fixed-wing UAV, however, without loss of generality, it can be applied to other types of vehicles. Fixed-wing aircrafts present constraints on their mobility such as minimum curvature, maximum angle of climb or dive, and minimum speed.

The possibility of using multiples UAVs brings additional advantages, including the reduction of time needed to complete a task, as well as improving the robustness of the overall system. Thus, the generation of a trajectory must also take into account all the vehicles in that air space. Having this as one of our research objectives, in this text we present an improved approach to the trajectory generation problem for multiples UAVs flying in an environment with obstacles. Our approach simplifies the three dimensional case to UAVs modeled as vehicles moving in two dimensions with non zero speed and limited turning rate.

An efficient adaptation of the well-known probabilistic technique used for the motion planning problem, called Rapidly-exploring Random Tree (RRT) is described, where the possible link between new states is calculated based on the use of a special type of Bézier curve, known as the Pythagorean Hodograph curve (PH). We also show the advantages of using this approach in the trajectory planning of multiple robots in different environments.

This paper is organized as follows. Next section presents previous works found in the literature which deal with the problem of motion planning. Section 3 is on our methodological approach. We first formalize the problem, and then we detail the classic RRT algorithm followed by a description of PH curves. Finally, our adaptation using these techniques is carefully developed and discussed. THe description and results of several experiments are presented in Section 4. Finally, Section 5 presents the conclusion and possible directions for future work.

2 Related Works

Motion-planning problem for autonomous vehicles is the subject of many investigations, and various works on this overall topic can be found in literature [1–3]. One possible taxonomy of the area can be based on the number of vehicles involved (one or several), and the presence or absence of obstacles in the environment. Even though the generation of feasible paths (or trajectories) for nonholonomic vehicles is in itself a great challenge, ignoring the possible presence of obstacles restricts a broader use of such techniques. Some approaches of single vehicle path planning in general environments can be found in literature [4, 5]. Voronoi diagram is a widely used technique to generate paths with such constraints [6, 7]. Rapidly-exploring Random Trees (RRTs) can also be used in this case, especially for solving the case of nonholonomic vehicles. Cheng et al. [8] presents trajectory planning for both an automobiles and a spacecraft. In the later example, even though an obstacle free environment is concerned, the focus remains on the motion constraints that need to be satisfied for a safe entry of the spacecraft in the Earth's atmosphere. Other works like [9, 10], use this technique to generate nominal paths for miniature air vehicles. The authors present an algorithm for terrain avoidance for the air platform BYU/MAV, which allows, among other things, flying through canyons and urban environments.

As we have already mentioned, the use of multiple vehicles can provide a better and more efficient execution of a given task, but at a possible higher cost and increase in the complexity of the controller responsible for the generation of such motions. Therefore, some works only deal with the generation of safe trajectories for all vehicles assuming an obstacle free environment. It is important to note that vehicles

on a team may be regarded as obstacles, making cooperation almost an inevitable approach for the problem [11, 12].

The more general case of motion planners that can handle multiple vehicles in the presence of obstacles can also be found in the literature. A common technique used to deal with a large number of robots is to team them up, simplifying the planner's task by considering all the robots in a group as a single robot [13, 14]. The main problem with this approach is when different robots in the team need to execute different tasks. One of the most popular techniques for dealing with those cases is based on potential fields [15]. The planning stage, in this case consists basically in computing the attraction force to the goal letting the obstacle avoidance be completely reactive. A major problem with this approach is the possible occurrence of local minima, and some works have described ways to overcome this problem [16]. Finally, two other methods are the use an adaptation of the classic roadmap path planning, making it act dynamically, considering each robot as a moving obstacle [17], and those based on geometric solutions [18].

3 Methodology

3.1 Problem Statement

Our technique assumes the existence of an environment with obstacles whose position and geometry are known. Also, others limitations for the robot navigation are imposed by its own kinematic constraints. Two configurations, $\mathbf{P}_{init}$ and $\mathbf{P}_{goal}$ respectively, describe the initial and final states and which define the position and the orientation of the robot in the extreme points of the path. These states can represent any pair of waypoints in a set, which in turn, is defined by the mission planning module at a higher level in the robot's architecture.

A trajectory may be defined, mathematically, as a parametric curve $\vec{r}(t)$ in the two-dimensional space, where t is the time parameter that continuously varies in $\mathbb{R}$. In this manner, the trajectory planning problem for a single robot can formally be described as:

$$\begin{aligned} \mathbf{P}_{init}(x_{init}, y_{init}, \psi_{init}) &= \vec{r}(t_{init}), \\ \mathbf{P}_{goal}(x_{goal}, y_{goal}, \psi_{goal}) &= \vec{r}(t_{goal}), \end{aligned} \tag{1}$$

where t_{init} and t_{goal} are the initial and final time values, respectively, for the curve parameter t.

Each waypoint is described by two position (x, y) and one orientation (ψ) variable. The variable ψ (also called yaw) is an angle that describes the waypoint orientation parallel to the XY plane in relation to the $\mathbf{X}$ axis of the space.

We consider both local and global constraints. The local constraints are related to the kinematic and dynamic behavior of the robot in the configuration space. The global constraints represent the composition of the obstacles in the environment. The RRT is a technique which generates paths that respect both constraints.

In order to be considered a feasible trajectory for the robot (in an environment with or without obstacles), the curve $\vec{r}(t)$ must simultaneously fulfill kinematic and dynamic constraints and their maximum numerical values. The main kinematic constraint considered in this work is the maximum curvature κ_{max}, that corresponds

to the minimum curvature radius of the vehicle in space. It is possible to completely define a curve in $\mathbb{R}^2$ only by means of its curvature function [19].

As far as the physics is concerned, the curvature may be defined as a quantity that is directly proportional to the lateral acceleration of the robot in the space. The value of κ_{max} is inversely proportional to the minimum curvature radius (ρ_{min}) of the curve that the vehicle is able to execute, which is also proportional to the vehicle's maximum lateral acceleration. The curvature function of a curve in the n-dimensional space is given by the following equation:

$$\kappa(t) = \frac{|\dot{\vec{r}}(t) \times \ddot{\vec{r}}(t)|}{|\dot{\vec{r}}(t)|^3}. \tag{2}$$

We still consider the minimum velocity constraint, which represents the minimum translation velocity for the robot on a given trajectory. In case of fixed-wing airplanes, for example, this is an important requirement once those vehicles are not capable to hover. This will also be considered when we discuss the motion planning problem for multiple robots.

With regard to the dynamic, it is important to consider the form by which such constraints vary in the time. The continuity of the curvature and velocity functions is another fundamental characteristic in the trajectory planning for real vehicles. Discontinuities in the curvature function, for example, can induce infinite variations of lateral acceleration, which of course, are not realizable. The same reasoning is valid for the velocity profile. Finally, the curve produced by the trajectory planning algorithm is continuously derivable, should be second order differentiable, according to the Eq. 2.

As far as global constraints are concerned, there are two types of environments. The most common type is the static environment, which is specified only by the geometry, position and orientation of the obstacles in it. In the dynamic environment, the obstacles (which may be other agents, such as robots) move through space, and their trajectories may or may not be known. As we will show later this text, we will use the RRT algorithm to generate trajectories that avoid both static and dynamic obstacles.

Finally, a trajectory $\vec{r}(t)$ is valid for a vehicle if, and only if, the magnitude of the curvature function is smaller than the vehicle's absolute maximum curvature value, and if the curve is entirely contained in a region of the environment free of obstacles, as described below:

$$\vec{r}(t) \rightarrow (|\kappa(t)| < \kappa_{max}) \ \wedge \ \left(\vec{r}(t) \in \mathbf{P}_{free}\right), \tag{3}$$

where $\mathbf{P}_{free}$ represent the set of all states that satisfy the global constraints.

To calculate this trajectory, we first established paths between states produced by the RRT algorithm. These paths are produced by means of the Pythagorean Hodograph curve technique that simultaneously fulfills the kinematic and dynamic constraints of the robot. The composition of these paths will be considered a trajectory when the velocity profile of the robot in each of the PH curves is defined. This will be accomplished by the calculation of the parametric speed of each curve that composes the RRT tree. With this, we will be able to establish positions and orientations for the robot in time, which is essential to avoid collision.

3.2 Rapidly-Exploring Random Tree

There are some key factors of a path generating method that need to be considered, such as its efficiency, its ability to deal with obstacles in the environment, and the feasibility of the produced trajectory given a robots kinematic and dynamic restrictions. A known sampling based planning algorithms used to solve this problem is a technique called Rapidly-exploring Random Tree (RRT) [20], which has been shown to be a good alternative, mainly due to the fact of its ability to quickly explore large areas, and to consider both environmental and the vehicle's nonholonomic constraints during its generation. Figure 1 shows the evolution of the RRT algorithm in a environment, which is faster than others techniques.

Some interesting features related to the RRTs are the fact that it is (probabilistically) complete under general conditions, since it always remains connected regardless of the amount of edges and that it is a relatively simple algorithm when compared to other techniques.

Algorithm 1 presents the basic steps for the generation of an RRT.

Algorithm 1 GENERATE_RRT($\mathbf{P}_{init}$, K)

1: $\mathcal{T}$.init($\mathbf{P}_{init}$)
2: **for** $k = 1$ to K **do**
3: $\quad \mathbf{P}_{rand} \leftarrow$ RANDOM_STATE()
4: $\quad \mathbf{P}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{P}_{rand}$, $\mathcal{T}$)
5: $\quad u_{new} \leftarrow$ SELECT_INPUT($\mathbf{P}_{rand}$, $\mathbf{P}_{near}$)
6: $\quad \mathbf{P}_{new} \leftarrow$ NEW_STATE($\mathbf{P}_{near}$, u_{new})
7: $\quad \mathcal{T}$.add_vertex($\mathbf{P}_{new}$)
8: $\quad \mathcal{T}$.add_edge($\mathbf{P}_{near}$, $\mathbf{P}_{new}$, u_{new})
9: **end for**
10: **return** $\mathcal{T}$

First of all, the starting point to the execution of the algorithm needs to be chosen. The starting point will represent a robot state ($\mathbf{P}_{init} \in \mathbf{P}_{free}$), and the maximum number of iterations (K) that the algorithm must perform before it stops must also set. Then, a random position on the map ($\mathbf{P}_{rand}$) is generated, and this position will be used to guide the growth of the tree. A search is executed through all nodes already in the tree to find the one that is the closest to the new random position

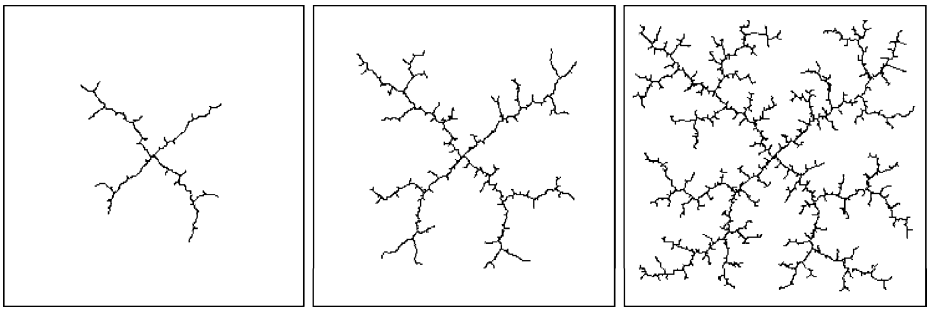

Fig. 1 Rapidly-exploring random tree algorithm evolution [20]

($\mathbf{P}_{near}$) according to some defined metric ρ (the Euclidean distance is a commonly used metric). The tree will expand from this node. A segment connecting $\mathbf{P}_{near}$ to $\mathbf{P}_{rand}$ must be inserted, however only a certain and fixed part of this segment will be actually used to expand the tree. A verification is made to check if it is possible to expand the tree, which means to verify if the segment doesn't intersect with any obstacles. If possible, a new state is generated and added to the tree ($\mathbf{P}_{new}$). In this case, instead of Euclidean distance, our method uses a metric ρ for determining the $\mathbf{P}_{near}$ that best suits the motion constraints of the robot. Then, we use the key ideas of a Pythagorean Hodograph curve to determine the value of $\mathbf{P}_{new}$, and describe a path between this and the nearest state.

3.3 Pythagorean Hodograph Curves

Pythagorean Hodograph (PH) curves are a special kind of parametric curves that present many computational advantages over polynomial parametric curves in general. They were introduced in [21], where, for the first time PH curves of fifth order for the two-dimensional case were presented . A Hermite Interpolation algorithm was proposed in [22], where the author demonstrate that there exist four possible solutions for the curve in $\mathbb{R}^2$. The chosen solution is the one that minimizes the cost function [23] (bending energy function) based on the integral of the magnitude of the curvature function.

A PH curve is represented, in general, as $\vec{p}(\tau) = [x(\tau),\ y(\tau)]$ which are the derivatives in relation to its parameter (hodograph components). They exhibit a special algebraic property, which is that they satisfy the Pythagorean condition

$$\dot{x}(\tau)^2 + \dot{y}(\tau)^2 = h(\tau)^2, \tag{4}$$

for some polynomial $h(\tau)$, where the variable τ is the curve parameter. This characteristic provides some properties for the PH curve that can be considered advantageous in the path planning problem: (a) uniform distribution of points along the curve, which contributes to the smoothness of the path; (b) elimination of numerical squaring in computing the path-length, which allows its determination by an exact form; (c) the parametric speed $\dot{s}(\tau)$ is a polynomial function of its parameter which means that the velocity profile of the curve is continuous; and finally (d) the curvatures and offset curves are in exact rational form, which enables PH curves to admit real-time interpolator algorithms for computer numerical control.

The length of the path, s, can be exactly calculated as

$$s = \int_{\tau_i}^{\tau_f} \sqrt{\dot{x}(\tau)^2 + \dot{y}(\tau)^2}\, d\tau = \int_0^1 |h(\tau)|\, d\tau, \tag{5}$$

and the PH curves are still shaped as fifth order Bézier curves

$$\vec{p}(\tau) = \sum_{k=0}^{5} \mathbf{p}_k \binom{5}{k} (1-\tau)^{5-k} \tau^k; \quad \tau \in [0, 1], \tag{6}$$

where $\mathbf{p}_k = [x_k,\ y_k]$ is the k-th control point of the Bézier curve. The Fig. 2 presents a example of PH curve of fifth order modeled as a Bézier function.

The path planning problem is then reduced to finding a solution to the Hermite Interpolation problem described in [21]. One important advantage of using this

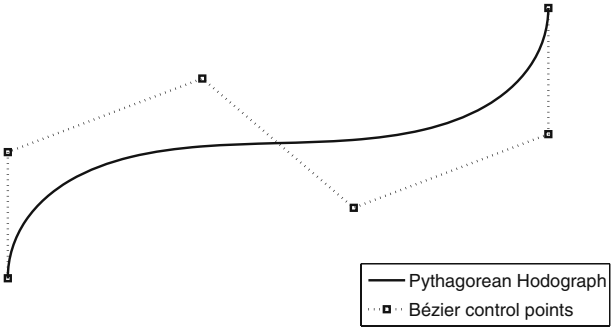

Fig. 2 Pythagorean Hodograph curve of fifth order and its Bézier control points for the two-dimensional case

model is that the resulting curve is infinitely continuous, so that the curvature function is always smooth.

Another advantage is related to the offset curve of a PH, $\vec{p}_o(\tau)$. This can be used to define a safety navigation region or still a sensor uncertainty for the position and orientation of the vehicle along the path. The offset curve can be computed as

$$\vec{p}_o(\tau) = \vec{p}(\tau) \pm d\mathbf{N}(\tau), \tag{7}$$

where $\mathbf{N}(\tau)$ represents the unit normal vector of the robot acceleration,

$$\mathbf{N}(\tau) = \frac{\dot{\mathbf{T}}(\tau)}{h(\tau)\,\kappa(\tau)},$$

that is a function of the variation of the unit tangent vector $\mathbf{T}(\tau)$ of the path, and d is the distance of the offset. As the offset curve self-intersects when the path is too convex or too concave, d must be chosen as a value less than the local radius of curvature to avoid this problem.

In the next section we show how to compute the Pythagorean Hodograph curve between two arbitrary states. This principle will be used when we generate a RRT tree whose vertices are connected by this Bézier curve. The total trajectory for a single robot will be a composition of PH curves, which can be stated as

$$\vec{r}(t) = \left[\vec{p}_1(t),\ \vec{p}_2(t), ...,\ \vec{p}_V(t)\right], \tag{8}$$

where V is the number of vertex in the tree that belongs to the correct path, and $\vec{p}_i(t)$ is represented as

$$\vec{p}_i(t) = \frac{s_i}{v_{max}} \left[\frac{\vec{p}_i(\tau)}{|\dot{s}_i(\tau)|}\right],$$

where s_i is the path-length of the PH given by Eq. 5, $\dot{s}_i(\tau)$ is the parametric speed and v_{max} is the maximum speed of the UAV.

3.4 Realizable Path Calculation

In this section we discuss how to generate paths that are attainable by a robot in a plane using the principles of PH curves. Those paths comply with the curvature constraints imposed by a given robot. Thus, assuming the model described by Eq. 6,

the path planning problem reduces to determining the six control points of the Bézier curve.

Four of these points can be calculated by the following set of equations (derived from [22]), which depends only the initial and final configuration of the robot:

$$\begin{aligned}
\mathbf{p}_0 &= [x_i,\ y_i],\\
\mathbf{p}_1 &= \mathbf{p}_0 + \frac{k_0}{5}[\cos(\psi_i),\ \sin(\psi_i)],\\
\mathbf{p}_4 &= \mathbf{p}_5 - \frac{k_5}{5}[\cos(\psi_f),\ \sin(\psi_f)],\\
\mathbf{p}_5 &= [x_f,\ y_f],
\end{aligned} \tag{9}$$

where k_0 and k_5 are gain factors that have unit value for PH with no constraints.

There are two problems to be resolved at this point. First, we must be able to calculate the remaining points, $\mathbf{p}_2$ and $\mathbf{p}_3$ of the Bézier curve. The determination of these points is closely related with the Pythagorean condition (Eq. 4). Second, we must choose the best values of k_0 and k_5 in order to comply with the curvature constraint of the robot.

To find the remaining points, we should resolve the following equation, derived form the Hermite Interpolation system presented in [22]:

$$\begin{aligned}
\mathbf{p}_2 &= \mathbf{p}_1 + \frac{1}{5}\begin{bmatrix} u_0u_1 - v_0v_1 \\ u_0v_1 + u_1v_0 \end{bmatrix}^T,\\
\mathbf{p}_3 &= \mathbf{p}_2 + \frac{1}{15}\begin{bmatrix} 2u_1^2 - 2v_1^2 + u_0u_2 - v_0v_2 \\ 4u_1v_1 + u_0v_2 + u_2v_0 \end{bmatrix}^T.
\end{aligned} \tag{10}$$

The parameters $[u_0, u_1, u_2]$ and $[v_0, v_1, v_2]$ represent coefficients of the polynomial function $u(t)$ and $v(t)$ respectively, used in the fifth order PH project. They can be calculated as

$$\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \sqrt{\frac{5}{2}}\begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_0\| + \Delta x_0} \\ \operatorname{sign}(\Delta y_0)\sqrt{\|\Delta\mathbf{p}_0\| - \Delta x_0} \end{bmatrix}, \tag{11}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \pm\sqrt{\frac{5}{2}}\begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_4\| + \Delta x_4} \\ \operatorname{sign}(\Delta y_4)\sqrt{\|\Delta\mathbf{p}_4\| - \Delta x_4} \end{bmatrix}, \tag{12}$$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = -\frac{3}{4}\begin{bmatrix} u_0 + u_2 \\ v_0 + v_2 \end{bmatrix} \pm \sqrt{\frac{1}{2}}\begin{bmatrix} \sqrt{c + a} \\ \operatorname{sign}(b)\sqrt{c - a} \end{bmatrix}, \tag{13}$$

where

$$\begin{aligned}
\Delta x_k &= x_{k+1} - x_k,\\
\Delta y_k &= y_{k+1} - y_k,
\end{aligned}$$

and

$$\Delta\mathbf{p}_k = [\Delta x_k,\ \Delta y_k].$$

The a, b and c parameters can be determinated by following:

$$a = \frac{9}{16}(u_0^2 - v_0^2 + u_2^2 - v_2^2) + \frac{5}{8}(u_0 u_2 - v_0 v_2) + \frac{15}{2}(x_4 - x_1),$$
$$b = \frac{9}{8}(u_0 v_0 + u_2 v_2) + \frac{5}{8}(u_0 v_2 + v_0 u_2) + \frac{15}{2}(y_4 - y_1),$$
$$c = \sqrt{a^2 + b^2}.$$

As stated before, there are four solutions for the PH calculation between any initial and final waypoints, $\mathbf{P}_i$ and $\mathbf{P}_f$, as it can be readily seen in the ambiguity of Eqs. 12 and 13. They represent an underdetermined system, for which the best solution of the combinatorial arrangement is the one that minimizes the cost function of the path, or the bending elastic energy function [23],

$$\mathcal{E} = \int_0^1 \kappa(\tau)^2 \, |\dot{\vec{p}}(\tau)| \, d\tau. \tag{14}$$

Once the Bézier points are computed, we must guarantee that the PH does note violate the kinematic constraints of the vehicle. For this, we must determine the values of k_0 and k_5 in Eq. 9. As there is no closed form solution, these values are increased iteratively, until that $\vec{p}(\tau)$ becomes realizable.

At this point we just monitor the maximum curvature and the minimum parametric speed of the PH. The global constraints will be obtained by a variation of the RRT algorithm described in the next section.

3.5 Single Trajectory Planning

Initially, the case of planning a trajectory for a single robot in an environment with obstacles will be considered. As stated earlier, the kinematic and dynamic constraints of the vehicle must be satisfied in order that the robot be able to execute the generated trajectory.

The first step in calculating the RRT consists of initializing the tree, which is accomplished by placing a node with the specifications of the initial state of the robot $\mathbf{P}_{init}$. This state not only describes the starting point of the tree but it also includes the initial orientation of the vehicle. The time of arrival at this node, and the initial speed of the UAVs, are considered to be null for the first node.

Then, a new random state $\mathbf{P}_{rand}$ is chosen for the expansion of the tree. There are many ways to achieve this step, as shown in [8]. According to the authors, a good approach is to perform a biased random choice of this state so that the goal $\mathbf{P}_{goal}$ is used some of the time. That helps in a more rapid conversion to the final result. In this work, the goal was chosen as a new position 50% of the time, with a completely random state in the rest of the time.

After choosing the random state, the next step is to identify which node already inserted in the tree is the closest to the generated one. Such proximity is measured by a metric ρ, determined for each specific issue dealt with by the RRT. The most common way to measure the proximity between two points in space is through the Euclidean distance D. However, this calculation does not take into account the orientation at these points in space.

When dealing with the navigation of nonholonomic vehicles (as those considered here), it is important to also consider the orientation, as the real distance between

two states can vary greatly between two positions. For this reason, a nonholonomic distance calculation is used, which is the same applied in the determination of the Dubins' Path [3]. The metric, ρ, that we use is described below.

It is known that there are six different kinds of paths between two configurations $\mathbf{P}_i$ and $\mathbf{P}_f$ as calculated by the Dubins' method. Among those, four represent an approximation for the length of the PH curve, while the other two paths generate configurations that can compromise the convergence of the PH curve. Therefore, the following condition will be used:

$$\rho = \begin{cases} \min(L_i) \quad i = 1...4, & \text{if } d > d_{min}, \\ \infty & \text{elsewhere}, \end{cases} \tag{15}$$

where $d = D/\rho_{min}$ represents the Euclidean distance between the points, and

$$d_{min} = \sqrt{4 - (|\cos(\alpha)| + 1)^2} + |\sin(\alpha)|, \tag{16}$$

is a parameter used as a minimum distance between the two states. In this specific case, it is considered that the angle of arrival at the final state is always zero, and

$$\alpha = \psi_i - \tan^{-1}\left(\frac{y_f - y_i}{x_f - x_i}\right). \tag{17}$$

Each of the four possible distances can be calculated as follows [24]:

$$\begin{aligned} L_1 &= \sqrt{d^2 + 2 - 2\cos(\alpha) + 2d\sin(\alpha)} - \alpha, \\ L_2 &= \sqrt{d^2 + 2 - 2\cos(\alpha) - 2d\sin(\alpha)} + \alpha, \\ L_3 &= \sqrt{d^2 - 2 + 2\cos(\alpha) + 2d\sin(\alpha)} - \alpha - 2\mu - \gamma, \\ L_4 &= \sqrt{d^2 - 2 + 2\cos(\alpha) - 2d\sin(\alpha)} + \alpha + 2\nu - \gamma, \end{aligned} \tag{18}$$

where

$$\mu = \tan^{-1}\left(\frac{-2}{d^2 - 2 + 2\cos(\alpha) + 2d\sin(\alpha)}\right),$$

$$\nu = \tan^{-1}\left(\frac{2}{d^2 - 2 + 2\cos(\alpha) - 2d\sin(\alpha)}\right),$$

and

$$\gamma = \tan^{-1}\left(\frac{\cos(\alpha) + 1}{d - \sin(\alpha)}\right).$$

Minimizing the function ρ gives an approximation to the nearest node $\mathbf{P}_{near}$ of the chosen random state. The last step is the creation of new state $\mathbf{P}_{new}$, which will be used to expand the tree. This will use the principles of the PH curve, as follows:

$$\mathbf{P}_{new} = \text{PH\_CURVE}(\mathbf{P}_{near}, \mathbf{P}_{rand}, \rho_{min}).$$

In that stage, the orientation ψ_{rand} of the random point is chosen such that it sets the direction of arrival in $\mathbf{P}_{new}$ to zero,

$$\psi_{rand} = \tan^{-1}\left(\frac{y_{rand} - y_{near}}{x_{rand} - x_{near}}\right).$$

This is an arbitrary choice that aims at reducing the number of iterations of a regular RRT calculation (determination of the u_{new} entries), by the use of an analytic function. The problem is that the generated PH curve may lead to a critical point where obstacles may hinder the progress of the tree in the direction of the new point. In the event of a collision like this, a random value can be added to ψ_{rand} in order to avoid the obstacles.

The entry vector is given by the control points of the Bézier curves, as follows:

$$u_{new} = \mathbf{p}_k, \quad k = [0...5].$$

Finally, the trajectory curve can be defined by Eq. 8. Figure 3 presents the evolution of a RRT tree with the Pythagorean Hodograph interpolation between its vertices for an UAV with $\rho_{min} = 10$ m.

We can see that the use of an analytical function as edges for the tree provides a fast convergence to the final result, since there is no need to limit the reach of $\mathbf{P}_{new}$. Compared with the traditional RRT algorithm presented in the Fig. 1, we find a much smaller number of vertices.

3.6 Trajectory Replanning

After the calculation of the first curve $\vec{r}(t)$ for the case of a single UAV, is performed the replanning stage of this trajectory. The main objective is to minimize the total length of the path made by the vehicle, estimated by the random technique of RRT.

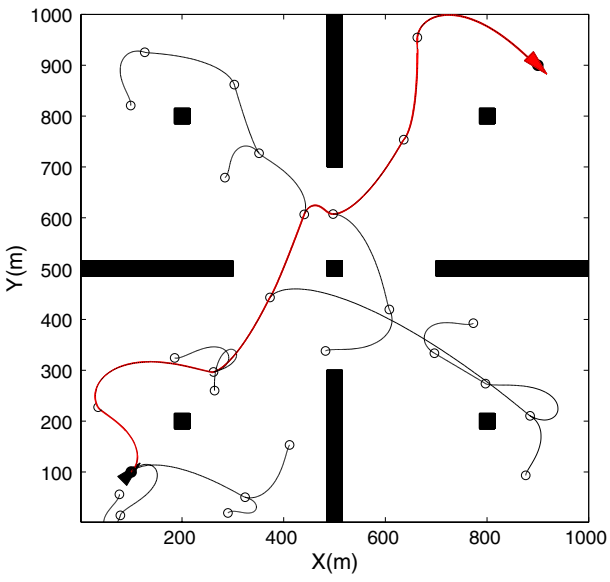

Fig. 3 Example of a RRT with Pythagorean Hodograph interpolation. The *black lines* represent the RRT tree, with the *circles* representing its vertices. *Red lines* correspond to the final trajectory of the UAV

A simple way to achieve this replanning is the elimination of the vertexes of the main tree that culminate in path length prohibitively unnecessary. The Algorithm 2 presents the main steps used in the replanning task from that principle. The method receives as input a tree $\mathcal{T}$ (with the trajectory description) created in the previous step.

Algorithm 2 REPLANNING_TRAJECTORY($\mathcal{T}$)

```
1: T_R.init(T)
2: i ← 1
3: while i < V do
4:     for j = V to i do
5:         p(τ) ← PH_CURVE(P_i, P_j, ρ_min)
6:         if ¬ COLLISION_DETECTION( p(τ) ) then
7:             u_new ← SELECT_INPUT(P_i, P_j)
8:             T_R.add_vertex(P_j)
9:             T_R.add_edge(P_i, P_j, u_new)
10:            i ← j
11:        end if
12:    end for
13: end while
14: return T_R
```

As stated before, V represents the total number of vertexes that compose the trajectory described in $\mathcal{T}$, while $\mathcal{T}_R$ corresponds to the new redesigned trajectory.

For each vertex $\mathbf{P}_i$ that describes the trajectory (corresponding initially to $\mathbf{P}_{init}$ of the tree), the algorithm tries to establish a direct connection with the vertex closest to the final goal ($\mathbf{P}_{goal}$). This connection is again built upon the calculation of a Pythagorean Hodograph curve, still seeking respect the limits of the UAV navigation constraints. For this, there should be no collision of the curve with the obstacles of the environment, as found in the algorithm.

Figure 4 shows the result of trajectory replanning applied to the example of the trajectory on the previous section. As we can see, the new trajectory generated presents a total length less than the first result, showing that the replanning has a great advantage, especially because of its low cost. Another advantage is to reduce the number of vertexes of the tree, making it faster for the planning of multiple UAVs, where each robot is now regarded as a dynamic obstacle for the others.

3.7 Multiple Trajectory Planning

After the generation of a trajectory for a single vehicle, the expansion for multiple vehicles can be immediately obtained.

As discussed before, trajectories are parameterized by time, which assumes that after a trajectory is calculated it is possible to know in which position the vehicle will be at a specific instant in time. This means one can assume each plane as an obstacle that is at different positions at different instants in time. Then, the next step

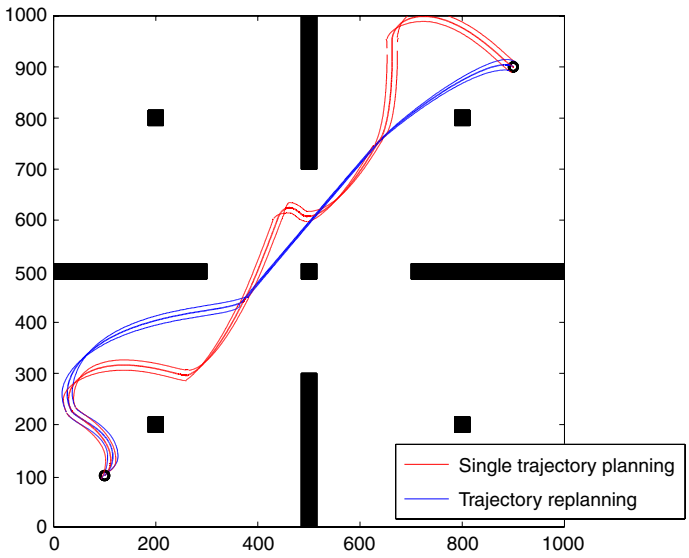

Fig. 4 Trajectory replanning algorithm result

is to parameterize $\mathbf{P}_{free}$ by time, thus representing different free positions at different moments. The new configuration space is then given by the equation below:

$$\mathbf{P}^{k}_{free}(t) = \mathbf{P}_{free} - \sum_{j=1}^{k-1} \vec{r}_j(t). \tag{19}$$

For a certain robot k, during the step of generating each new state in the expansion of the tree, it will be calculated which are the free positions at an instant in time. In order to do that, the planner has to subtract from the free space, based only on the static obstacles ($\mathbf{P}_{free}$), all the positions that will be occupied by other robots at that instant in time t, where $\vec{r}_j(t)$ represents the position j of the robot in its trajectory at instant t.

Thus, the paths are computed sequentially, where the first UAV considers only the static obstacles already present in the environment and each UAV considers the obstacles and trajectories computed for the other UAVs.

4 Experiments

Our technique was used to plan trajectories for a set of simulated small unmanned aerial vehicles. Those robots were modeled as fixed-wings aircrafts based on a UAV named AqVS (Fig. 5), developed at Universidade Federal de Minas Gerais/Brazil.

Fig. 5 AqVS, an UAV from the Universidade Federal de Minas Gerais-Brazil [25]

This is a small hand launched hybrid electric motor sail plane, equipped with GPS receptor, barometric altimeter, infrared inclinometer, airspeed sensor and CCD camera, and controlled by a set of PID stabilizators running on a Palm® PDA for autonomous navigation [25]. The AqVS presents the following characteristics:

- ρ_{min}: about 50 m,
- maximum cruising speed: approximately 50 km/h,
- uncertainty of localization: 12 m.

The above values were determined using data from actual flights, considering a speed of approximately 50 km/h. This vehicle has shown to be a good choice for testing our methodology because of its large uncertainty of localization.

Three experiments were realized with using our approach. In the first experiment, a group of four AqVS/UAVs were used. The objective of each one of those robots was to exchange its position with another, flying through the environment without collision with obstacles or others UAVs. Its initial and goal configurations were chosen as:

- UAV_1, $\mathbf{P}_{init}\left(100, 100, \frac{\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(900, 900, \frac{-3\pi}{4}\right)$,
- UAV_2, $\mathbf{P}_{init}\left(900, 900, \frac{-3\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(100, 100, \frac{\pi}{4}\right)$,
- UAV_3, $\mathbf{P}_{init}\left(100, 900, \frac{-\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(900, 100, \frac{3\pi}{4}\right)$,
- UAV_4, $\mathbf{P}_{init}\left(900, 100, \frac{3\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(100, 900, \frac{-\pi}{4}\right)$.

Figure 6 shows the result of the trajectory planning for all UAVs. In this experiment, we built a simple environment with few obstacles, just to assess the methodology. Although the paths self-intersect in space, the trajectories generated do not intersect for any given time.

One can still observe that the offset path to each PH curve that composes the path, which was calculated by means of Eq. 7. This represent a safety-flight area to the robot navigation because any uncertainty on the robot localization still is contained in $\mathbf{P}_{free}(t)$.

Another experiment was conducted, now considering a more complex environment, with about one hundred static obstacles and four UAVs. Most of the

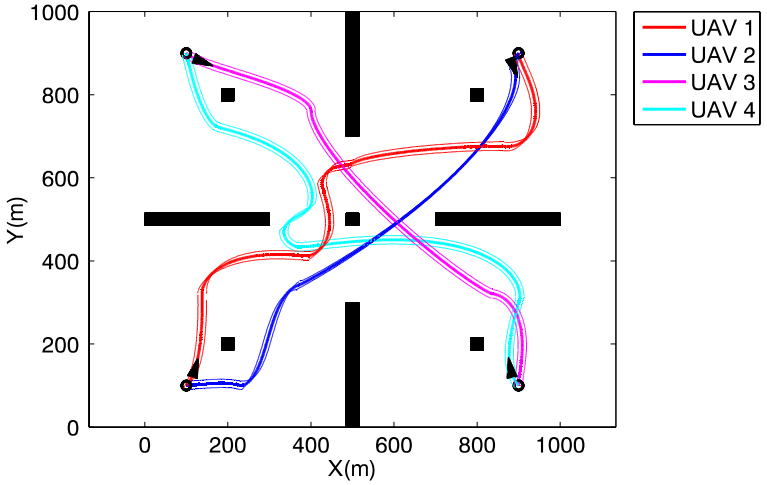

Fig. 6 Trajectory planning for a set of four AqVS/UAVs in a simple environment

localization and geometry of those obstacles were chosen randomly, as shown in Fig. 7. In this case, the following UAV configurations were used:

- UAV_1, $\mathbf{P}_{init}(100, 520, 0)$ and $\mathbf{P}_{goal}(900, 520, \pi)$,
- UAV_2, $\mathbf{P}_{init}(900, 480, \pi)$ and $\mathbf{P}_{goal}(100, 480, 0)$,
- UAV_3, $\mathbf{P}_{init}(100, 480, 0)$ and $\mathbf{P}_{goal}(900, 480, \pi)$,
- UAV_4, $\mathbf{P}_{init}(900, 520, \pi)$ and $\mathbf{P}_{goal}(100, 520, 0)$.

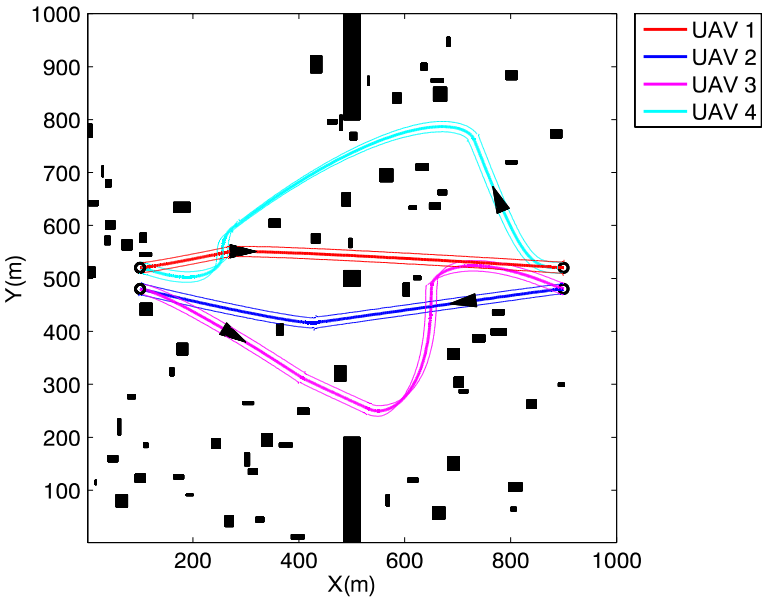

Fig. 7 Trajectory planning for a set of four AqVS/UAVs in a more complex environment

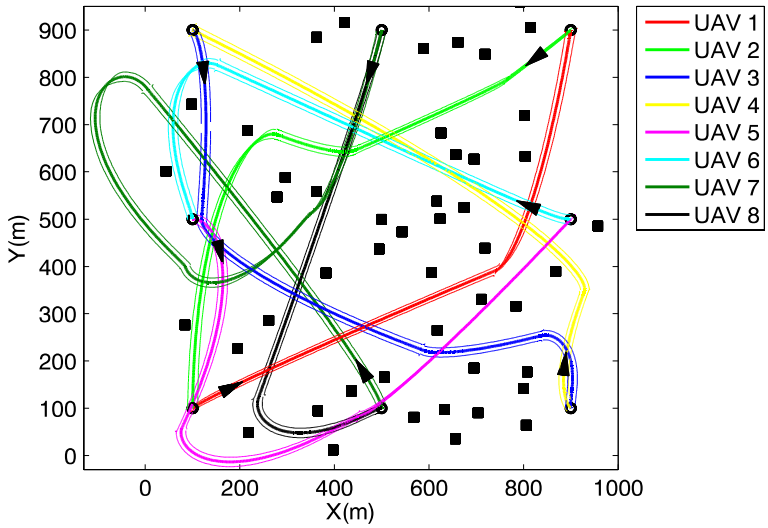

Fig. 8 Trajectory planning for a set of eight AqVS/UAVs in an environment with random obstacles

In the last experiment, we exercised our methodology with a set of eight UAVs and fifty obstacles disposed randomly in space. The configurations to the set of robots were chosen as:

- UAV_1, $\mathbf{P}_{init}\left(100, 100, \frac{\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(900, 900, \frac{-3\pi}{4}\right)$,
- UAV_2, $\mathbf{P}_{init}\left(900, 900, \frac{-3\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(100, 100, \frac{\pi}{4}\right)$,

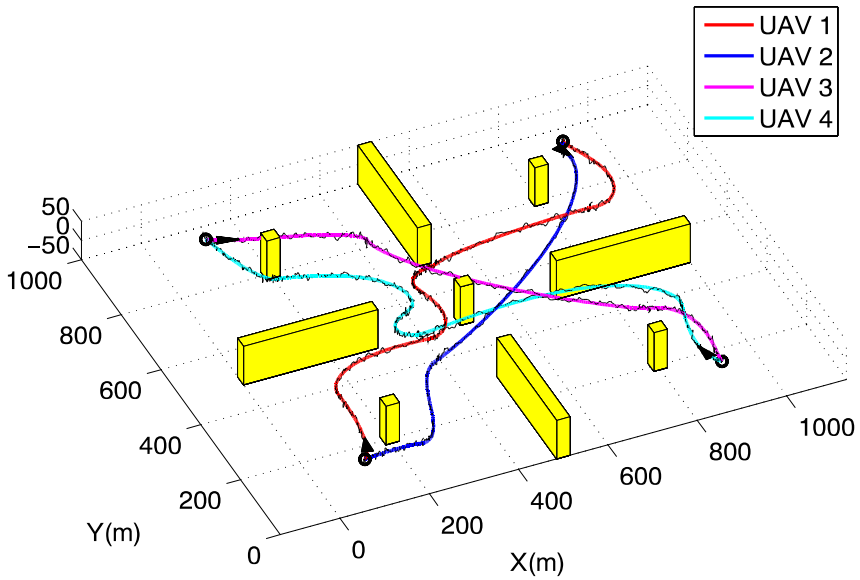

Fig. 9 AqVS/UAV trajectory following for the planning in the first test

- UAV_3, $\mathbf{P}_{init}\left(100, 900, \frac{-\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(900, 100, \frac{3\pi}{4}\right)$,
- UAV_4, $\mathbf{P}_{init}\left(900, 100, \frac{3\pi}{4}\right)$ and $\mathbf{P}_{goal}\left(100, 900, \frac{-\pi}{4}\right)$,
- UAV_5, $\mathbf{P}_{init}\,(100, 500, 0)$ and $\mathbf{P}_{goal}\,(900, 500, \pi)$,
- UAV_6, $\mathbf{P}_{init}\,(900, 500, \pi)$ and $\mathbf{P}_{goal}\,(100, 500, 0)$,
- UAV_7, $\mathbf{P}_{init}\left(500, 100, \frac{\pi}{2}\right)$ and $\mathbf{P}_{goal}\left(500, 900, \frac{-\pi}{2}\right)$,
- UAV_8, $\mathbf{P}_{init}\left(500, 900, \frac{-\pi}{2}\right)$ and $\mathbf{P}_{goal}\left(500, 100, \frac{\pi}{2}\right)$.

Figure 8 shows the final result of the trajectory planning for all UAVs. Once more, there are no collisions.

The planned trajectories were tested with the dynamic model of the AqVS model implemented in Matlab. Figure 9 shows the final result for the first planning.

5 Conclusion and Future Works

We have described a novel technique that allows the planning of trajectories for multiple vehicles in a very complex, with a large number of obstacles. While the technique of RRT already allows the generation of smooth paths for nonholonomic vehicles, in some cases, the kinematic and dynamic models used in the integration step of the algorithm can become very complex. There is a great need for a reliable model of the vehicle, otherwise there is a chance that the generated path may not be achieved by a real robot. This is the case a real aircraft, where some of its aerodynamic coefficients are very difficult to model.

The use of analytical curves, such as Bézier curves, allows for greater flexibility of this model with a low computational cost. The PH curves, in particular allows for the calculation of offset curves and the parametric speed functions directly. The project of these curves takes into account very simple kinematics and dynamics constraints, which implies the simplification of the model of the vehicle around a few points of operation.

An important advantage of our method is the reduction in the computational time to convert the RRT algorithm, because the use of PH curves enable connections between vertices of the tree with unlimited range. In an environment with few obstacles, a very small quantity of vertices (sometimes only two) is sufficient to take the robot from $\mathbf{P}_{init}$ to $\mathbf{P}_{goal}$.

As future work, we plan to expand the use of the technique for the three-dimensional space, considering other kinematic constraints, as the maximum torsion and maximum climb (dive) angle. It is possible to use an expansion of the PH curve to three dimensions, while maintaining the same benefits described in this paper.

Another possibility is to evaluate other metrics for ρ in the RRT algorithm, since the presented here cannot be applied in the three-dimensional case. Perhaps a metric based on the actual length of the PH curve might be a better solution, despite their higher computational cost.

Acknowledgements The authors gratefully thank prof. Paulo Iscold for the flight model of the AqVS/UAV. This work was developed with the support of CNPq, CAPES and FAPEMIG.

References

1. Siegwart, R., Nourbakhsh, I.R.: Introduction to Autonomous Mobile Robots. MIT, Cambridge (2004)
2. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
3. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am. J. Math. **79**, 497–516 (1957)
4. Kuwata, Y., Richards, A., Schouwenaars, T., How, J.P.: Robust constrained receding horizon control for trajectory planning. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2005)
5. Wzorek, M., Doherty, P.: Reconfigurable path planning for an autonomous unmanned aerial vehicle. Hybrid Information Technology, 2006. ICHIT '06. International Conference on vol. 2, pp. 242–249 (2006)
6. Bortoffl, S.A.: Path planning for UAVs. In: Proceedings of the American Control Conference (2000)
7. Dogan, A.: Probabilistic path planning for UAVs. In: Proceedings of 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations (2003)
8. Cheng, P., Shen, Z., LaValle, S.: RRT-based trajectory design for autonomous automobiles and spacecraft. Arch. Control Sci. **11**(3–4), 167–194 (2001)
9. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Maximizing miniature aerial vehicles. Robot. Autom. Mag. IEEE **13**(3), 34–43 (2006). doi:10.1109/MRA.2006.1678137
10. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Advances in unmanned aerial vehicles: state of the art and road to autonomy, chap. Obstacle and Terrain Avoidance for Miniature Aerial Vehicles, pp. 213–244. Springer, Tampa (2007)
11. Schouwenaars, T., How, J., Feron, E.: Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2004)
12. Shanmugavel, M., Tsourdos, A., Zbikowski, R., White, B.A., Rabbath, C.A., Léchevin, N.: A solution to simultaneous arrival of multiple UAVs using Pythagorean Hodograph curves. In: Proceedings of the IEEE American Control Conference (ACC), pp. 2813–2818. Minneapolis (2006)
13. Li, T.Y., Chou, H.C.: Motion planning for a crowd of robots. Proc. IEEE Int. Conf. Robot. Autom. **3**, 4215–4221 (2003)
14. Belta, C., Kumar, V.: Abstraction and control for groups of robots. IEEE Trans. Robot. **20**(5), 865–875 (2004)
15. Warren, C.: Multiple robot path coordination using artificial potential fields. Robotics and Automation, 1990. Proceedings, 1990 IEEE International Conference on vol. 1, pp. 500–505 (1990)
16. Marcolino, L.S., Chaimowicz, L.: No robot left behind: coordination to overcome local minima in swarm navigation. In: Proceedings of the 2008 IEEE International Conference on Robotics and Automation (2008)
17. Gayle, R., Sud, A., Lin, M., Manocha, D.: Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. IEEE/RSJ International Conference on Intelligent Robots and Systems pp. 3777–3783 (2007)
18. Leroy, S., Laumond, J.P.: Multiple path coordination for mobile robots: a geometric algorithm. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI, pp. 1118–1123 (1999)
19. Kreyszig, E.: Differential Geometry, vol. 1. Dover, New York (1991)
20. Lavalle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Tech. Rep., Computer Science Dept., Iowa State University (1998)
21. Farouki, R.T., Sakkalis, T.: Pythagorean Hodographs. IBM J. Res. Develop. **34**(5), 736–752 (1990)
22. Farouki, R.T., Neff, C.A.: Hermite interpolation by Pythagorean Hodograph quintics. Math. Comput. **64**, 1589–1609 (1995)
23. Farouki, R.T.: The elastic bending energy of Pythagorean Hodograph curves. Comput. Aided Geom. Des. **13**, 227–241 (1996)
24. Shkel, A.M., Lumelsky, V.: Classification of the Dubins set. In: Robotics and Autonomous Systems, vol. 34, pp. 179–202 (2001)
25. Iscold, P.: Development of a small unmanned aerial vehicle for aerial reconaiscence. In: International Congress of Mobility Engineering. São Paulo, Brazil (2007)