The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

On the Generation of Feasible Paths for Aerial Robots in Environments with Obstacles

Douglas G. Macharet Armando A. Neto Mario F. M. Campos

Abstract— **This paper presents a methodology based on a variation of the Rapidly-exploring Random Trees (RRTs) that generates feasible trajectories for 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. 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 obstacles of a environment. We show results for a small unmanned aerial vehicles in environments with different configurations.**

I. INTRODUCTION

Several strategies to obtain a path between two different positions can be found in the literature. Among the simplest and most common of such strategies are the visibility graph, cell decomposition and potential field path planners. However, one of the problems with them is that they guarantee obstacle free path between two points without considering the vehicle's capacity to follow the generated path (e.g. negotiate a curve between obstacles).

The interest and research in Unmanned Aerial Vehicles (UAVs) have been increasingly growing, specially due to the decrease in cost, weight, size and performance of actuators, sensors and processors. Clearly UAVs have their niche of applications, which cannot be occupied by other types of mobile robots, as far as the capacity of covering a broad set of relevant applications is concerned. 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 advantages 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 a 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. One 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, the radius of curvature is one of the restrictions imposed on trajectories generated for cars, since a typical car cannot move laterally.

The authors are with the Computer Vision and Robotics Laboratory (Verlab), Computer Science Department, Federal University of Minas Gerais, Brazil. e-mails: {doug,aaneto,mario}@dcc.ufmg.br

UAVs can be divided into at least three categories: 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 in a 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.

In our approach, the more general case of UAV's moving in the three dimensional is efficiently modeled as vehicles moving in two dimensions with non zero speed and limited turning rate. A novel 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 path planning of robots in environments with obstacles.

II. 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], [2], [3]. One possible taxonomy of the area can be based on the number of vehicles involved, 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 even further 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. [8] presents trajectory planning for both an automobiles and a 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], 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.

There are still some works that only deal with the generation of safe paths for vehicles assuming an obstacle free environment. Among them is the work by [10], which presents one of the first methods about the use of the Pythagorean Hodograph curves in the path planning problem. The author discusses numerous advantages (which will be listed later in this text) of use such a curve in the modeling of paths for vehicles with holonomic constraints.

III. Methodology

A. Problem Statement

Our technique assumes the existence of an environment with obstacles whose position and geometry are known. Also, other 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 poses (or states) which also 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 path may be defined, mathematically, as a parametric curve $\vec{r}(t)$ in the two-dimensional space, where t is the parameter that continuously varies in $\mathbb{R}$. In this manner, the path planning problem for a single robot can be formally 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 parameter 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 path 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 [11].

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 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:

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

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

As far as global constraints are concerned, one may define two types of environments, the most common one being the static environment, which is specified only by the geometry, position and orientation of the obstacles therein. In a dynamic environment, the obstacles (which may also be other agents, such as robots) move through space, and their trajectories may or may not be previously known. As we will show later, we lay hold of the RRT algorithm to generate trajectories which avoid both static and dynamic obstacles.

Finally, a path $\vec{r}(t)$ 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:

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

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

To calculate this path, we first established a curve between states produced by the RRT algorithm. These curves are produced by means of the Pythagorean Hodograph curve technique that simultaneously fulfills the kinematic and dynamic constraints of the robot.

B. 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 paths 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) [12], 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.

Among the interesting features related to the RRTs are i) that it is (probabilistically) complete under general conditions, since it always remains connected regardless of the amount of edges, and ii)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:      $\mathbf{P}_{rand} \leftarrow$ RANDOM_STATE()
4:      $\mathbf{P}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{P}_{rand}$, $\mathcal{T}$)
5:      $u_{new} \leftarrow$ SELECT_INPUT($\mathbf{P}_{rand}$, $\mathbf{P}_{near}$)
6:      $\mathbf{P}_{new} \leftarrow$ NEW_STATE($\mathbf{P}_{near}$, $u_{new}$)
7:      $\mathcal{T}$.add_vertex($\mathbf{P}_{new}$)
8:      $\mathcal{T}$.add_edge($\mathbf{P}_{near}$, $\mathbf{P}_{new}$, $u_{new}$)
9:  end for
10: return $\mathcal{T}$
```

First of all, the starting point of the execution of the algorithm needs to be chosen. The starting point represent a robot pose ($\mathbf{P}_{init} \in \mathbf{P}_{free}$), 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 ($\mathbf{P}_{near}$) 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.

C. 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 [13], where, for the first time PH curves of fifth order for the two-dimensional case were presented. A Hermite Interpolation algorithm was proposed in [14], 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 [15] (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 PH curve parameter. This characteristic provides some properties for the PH curve that can be considered advantageous in the path planning problem: (i) uniform distribution of points along the curve, which contributes to the smoothness of the path; (ii) elimination of numerical squaring in computing the path-length, which allows its determination by an exact form; (iii) 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 (iv) 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 path planning problem is then reduced to finding a solution to the Hermite Interpolation problem described in [13]. One important advantage of using this 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 poses. This principle will be used when we generate a RRT tree whose vertices are connected by this Bézier curve. The total path for a single robot will be a composition of PH curves, which can be stated as

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

where V is the number of vertex in the tree that belongs to the correct path.

D. Realizable Path Calculation

In this section we discuss the generation of 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 Equation 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 [14]), 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 solved 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 (Equation 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 solve the following equation, derived form the Hermite Interpolation system presented in [14]:

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

The parameters $[u_0,\, u_1,\, u_2]$ and $[v_0,\, v_1,\, v_2]$ represent coefficients of the polynomial function $u(\tau)$ and $v(\tau)$ respectively, used in the fifth order PH design. 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:

$$
\begin{aligned}
a &= \frac{9}{16}(u_0^2 - v_0^2 + u_2^2 - v_2^2) + \frac{5}{8}(u_0u_2 - v_0v_2) + \frac{15}{2}(x_4 - x_4),\\
b &= \frac{9}{8}(u_0v_0 + u_2v_2) + \frac{5}{8}(u_0v_2 + v_0u_2) + \frac{15}{2}(y_4 - y_1),\\
c &= \sqrt{a^2 + b^2}.
\end{aligned}
$$

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 equations 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 [15],

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

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 Equation 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.

E. Tree Generation

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 pose of the robot $\mathbf{P}_{init}$. This pose not only describes the starting point of the tree but it also includes the initial orientation of the vehicle.

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 poses. In this specific case, it is considered that the angle of arrival at the final pose 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 [16]:

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

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 curve can be defined by Equation 8.

IV. Experiments

Our technique was used to plan paths for a simulated small unmanned aerial vehicle. The robot was modeled as fixed-wing aircraft based on a UAV named AqVS (Figure 1), developed at Universidade Federal de Minas Gerais/Brazil. It 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 [17]. The AqVS presents the following characteristics:

- ρ_{min}: about 30 meters,
- maximum cruising speed: approximately 50 km/h,
- uncertainty of localization: 12 meters.

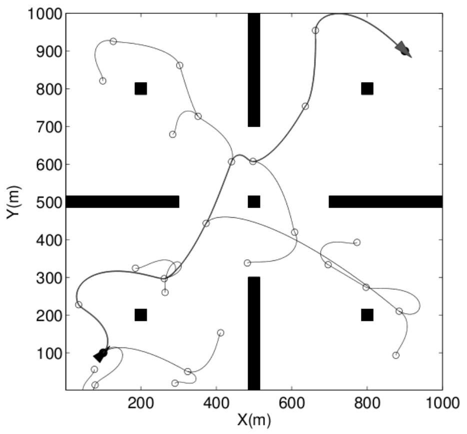
The above values were determined using data from actual flights, considering a speed of approximately 50 km/h. The first value is directly considered in the path calculation, while the third value is used to compute the offset PH curves as the parameter d for determination of the safety-flight area. This vehicle has shown to be a good choice for testing our methodology because of the large uncertainty of its localization.

Fig. 1. AqVS, a UAV from the Universidade Federal de Minas Gerais-Brazil.

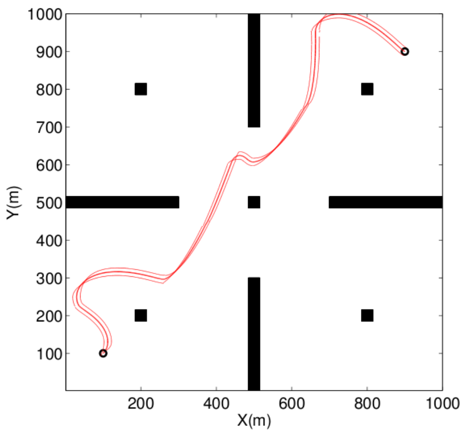
Figure 2 presents the evolution of a RRT tree with the Pythagorean Hodograph interpolation between its vertices for an AqVS/UAV.

We can see in Figure 2(a) 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, we found a much smaller number of vertices.

Figure 2(b) still shows the offset path to each PH curve that composes the path, which was calculated by means of Equation 7. This represent a safety-flight area for robot navigation because any uncertainty on the robot localization is likely to be contained inside $\mathbf{P}_{free}(t)$.



(a)



(b)

Fig. 2. Example of a RRT with Pythagorean Hodograph interpolation. (a) Path calculation with the black lines representing the RRT tree, and the circles representing its vertices. (b) Final path, in red line, with the PHs and its offset curves.

V. Conclusion and Future Works

We have described a novel technique that allows the planning of paths for robots in environments with a variable 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 and the external disturbances (e.g. wind) 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 at 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 (or dive) angles. Initial results have shown that it is possible to use an extension of the PH curve into three dimensions, while maintaining most of the advantages described in this paper.

VI. Acknowledgment

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] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[3] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[4] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Robust constrained receding horizon control for trajectory planning," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2005.

[5] M. Wzorek and P. Doherty, "Reconfigurable path planning for an autonomous unmanned aerial vehicle," *Hybrid Information Technology, 2006. ICHIT '06. International Conference on*, vol. 2, pp. 242–249, Nov. 2006.

[6] S. A. Bortoffl, "Path planning for uavs," in *Proceedings of the American Control Conference*, 2000.

[7] A. Dogan, "Probabilistic path planning for uavs," in *Proceedings of 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations*, 2003.

[8] P. Cheng, Z. Shen, and S. LaValle, "RRT-based trajectory design for autonomous automobiles and spacecraft," 2001.

[9] S. Griffiths, J. Saunders, A. Curtis, B. Barber, T. McLain, and R. Beard, "Maximizing Miniature Aerial Vehicles," *Robotics and Automation Magazine, IEEE*, vol. 13, no. 3, pp. 34–43, Sept. 2006.

[10] M. Shanmugavel, A. Tsourdos, R. Zbikowski, B. A. White, C. A. Rabbath, and N. Léchevin, "A Solution to Simultaneous Arrival of Multiple UAVs using Pythagorean Hodograph Curves," in *Proceedings of the IEEE American Control Conference (ACC)*, Minneapolis, Minnesota, USA, June 2006, pp. 2813–2818.

[11] E. Kreyszig, *Differential Geometry*. New York: Dover Publications, June 1991, vol. 1.

[12] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep., 1998.

[13] R. T. Farouki and T. Sakkalis, "Pythagorean Hodographs," *IBM Journal of Research and Development*, vol. 34, no. 5, 1990.

[14] R. T. Farouki and C. A. Neff, "Hermite Interpolation by Pythagorean Hodograph Quintics," *Mathematics of Computation*, vol. 64, pp. 1589–1609, 1995.

[15] R. T. Farouki, "The Elastic Bending Energy of Pythagorean Hodograph Curves," *Comput. Aided Geom. Design*, vol. 13, pp. 227–241, 1996.

[16] A. M. Shkel and V. Lumelsky, "Classification of the Dubins Set," in *Robotics and Autonomous Systems*, vol. 34, September 2001, pp. 179–202.

[17] P. Iscold, "Development of a Small Unmanned Aerial Vehicle for Aerial Reconaiscence," in *International Congress of Mobility Engineering*, São Paulo, Brazil, 2007.