The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Feasible RRT-based Path Planning Using Seventh Order Bézier Curves

Armando A. Neto Douglas G. Macharet 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 vehicles with holonomic constraints in environments with obstacles. Our approach is based on seventh order Bézier curves to connect vertexes of the tree, generating paths that do not violate the main kinematic constraints of the vehicle. The methodology also does not require complex kinematic and dynamic models of the vehicle. The smoothness of the acceleration profile of the entire path is directly guaranteed by controlling the curvature values at the extreme points of each Bézier that composes the tree. The proposed algorithm provides fast convergence to the final result with several other advantages, such as the reduction in the number of vertexes of the tree because the method enable connections between vertexes of the tree with unlimited range. In an environment with few obstacles, a very small quantity of vertexes (sometimes only two) is sufficient to take the robot between two points. The properties of the seventh order Bézier formulation are also used to avoid collisions with static obstacles in the environment.**

I. Introduction

The interest and research in Unmanned Aerial Vehicles (UAVs) has 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 tasks.

As the availability increases, so does the possibility of having multiple of such vehicles traversing a given volume of the air space. Therefore, there is a growing need to study and develop techniques for the generation of safe and feasible trajectories considering 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 such restriction imposed on trajectories generated for cars, since a typical car cannot move laterally.

In our approach, the more general case of UAV's movement is modeled as vehicles moving 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) [1] is described, where the possible link between new states is calculated based on the use of seventh order Bézier functions. We also show the advantages of using this approach in path planning for robots in environments with obstacles.

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

II. Related Works

Path-planning problem for autonomous vehicles is the subject of many investigations, and various works on this topic is reported in the literature. 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 trajectories for nonholonomic vehicles is in itself a great challenge, ignoring the possible presence of obstacles limits even further a broader use of such techniques.

Some approaches of single vehicle path planning in general environments can be found in literature [2], [3]. Voronoi diagram is a widely used technique to generate paths with such constraints [4], [5]. Rapidly-exploring Random Trees (RRTs) can also be used, especially for solving the path problem for nonholonomic vehicles. In [6] the authors present trajectory planning for both an automobile and a spacecraft. In the later example, even though an obstacle free environment is considered, the focus remains on the motion constraints that need to be satisfied for a safe entry of the spacecraft in Earth's atmosphere. Other works like [7] use this technique to generate nominal paths for miniature aerial vehicles. The authors present an algorithm for terrain avoidance for the air platform BYU/MAV, which , among other things, enables the vehicle to fly through canyons and urban environments.

There are also some works that only deal with the generation of safe paths for vehicles assuming an obstacle free environment. Among those is the work of [8], which presents one of the first methods based on the Pythagorean Hodograph (a special type of Bézier curve) in the path planning for UAVs. The author discusses numerous advantages of using such a curve in the modeling of paths for vehicles with holonomic constraints.

An important work that deals with the use of Bézier curves in the RRT algorithm framework can be found in [9]. In that work, the authors first calculate the RRT through the environment, and after that they use cubic Bézier curves to make the path feasible for a nonholonomic vehicle. This approach is simple, however it requires a great number of curves to compose a path in more complex environments (e.g. with several obstacles).

There are two main differences between this and our previous work [10]. First, this method focuses on a single UAV with kinematic constraints. Second, in this work we deal

with the problem of generating a motion plan with continuous curvature profile throughout the path.

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 and dynamic 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 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 mathematically defined as a parametric curve $\vec{r}(t)$ in the two-dimensional space, where t is the parameter that continuously varies in $\mathbb{R}$. Therefore, 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 positions (x, y) and one orientation (ψ) variable. The variable ψ (also called yaw) is an angle about the $\mathbf{Z}$ axis that describes the waypoint orientation parallel to the XY plane.

Both local and global constraints are considered. 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), a 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 that the vehicle can execute when moving 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 classical mechanics 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:

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

As far as dynamic is concerned, it is important to consider the way in which such constraints vary in time. The continuity of the curvature and velocity functions are another fundamental characteristic in planning paths for real vehicles. Discontinuities in the curvature function, for example, may induce infinite variations of lateral acceleration which, obviously, is unrealizable. The same reasoning is valid for the velocity profile. Finally, the curve produced by the path planning algorithm should be, at least, second order differentiable, according to 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 geometry, position and orientation of the obstacles. 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 because of its capability to generate trajectories which avoid static obstacles, but also because it is easily extensible for scenarios with 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, it is smooth in acceleration and the curve is entirely contained in an obstacle free region of the environment. In formal terms, we have three constraint conditions:

- $\vec{r}(t) \rightarrow |\kappa(t)| \leq \kappa_{max}$,
- $\vec{r}(t)$ is C^2 ,
- $\vec{r}(t) \in \mathbf{P}_{free}$,

where C^n is a n-th order differentiable function and $\mathbf{P}_{free}$ 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 will be a composition of seventh order Bézier functions that simultaneously fulfills the kinematic and dynamic constraints of the robot (which fulfills the first constraint item above). The second one is fulfilled by controlling the extreme curvature values of each curve in order to keep the union of Bézier curves continuous. Finally, the last item is guaranteed by the RRT principles described below.

B. Rapidly-Exploring Random Trees

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 generated paths given the kinematic and dynamic constraints of the robot. A known sampling based planning algorithm used to solve this problem is a technique called Rapidly-exploring Random Tree (RRT) [1]. The RRT algorithm has shown to provide good solutions, mainly due to the fact of its ability to quickly explore large areas and being able to consider both the environment and the vehicle's holonomic constraints during its generation.

Among the interesting features related to the RRTs are (i) that it is (probabilistically) complete under general conditions, since the tree 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 generating the 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 represents 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. 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 metric ρ (the Euclidean distance is a commonly used metric). The tree will then 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 it will actually be 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 any obstacle. If possible, a new state is generated and added to the tree ($\mathbf{P}_{new}$). This segment is computed by time integration of the differential model of the vehicle, which means that is necessary to know several parameters of the robot and to spend a lot of computational time to determine the path.

In this paper, instead of using a complex model, we compute the segments between vertexes of the RRT by means of curves that are realizable by the considered vehicles. These curves fulfill the three requirements described in the previous section. This approach decreases the computational complexity of the classical RRT method and eliminates the task of building a mathematic model of the robot. Also, instead of Euclidean distance, our method uses a metric ρ for determining the $\mathbf{P}_{near}$ that better fits the motion constraints of the robot.

C. 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 perform this step, as shown in [6]. 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. This step helps in achieving a faster conversion to the final result. In our work, the goal was chosen as a new position 20% of the time.

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

When dealing with the navigation of nonholonomic vehicles (as those considered here), it is also important to consider the vehicle´s 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 [12]. The metric (ρ) that we use is described below.

There are six different kinds of paths between two configurations $\mathbf{P}_i(x_i, y_i, \psi_i)$ and $\mathbf{P}_f(x_f, y_f, \psi_f)$ as determined by the Dubins' Path technique. Among those, four known as the *long case* (CSC) represent an approximation for the length of the Bézier curve, while the other two paths known as the *short case* (CCC) generate configurations that can compromise the convergence of the Bézier curve. Therefore, the following condition will be used:

$$\rho = \begin{cases} \min(L_i) \quad i = 1...4, & \text{if } d > d_{min}, \\ \infty & \text{elsewhere,} \end{cases} \tag{3}$$

where $d = D/\rho_{min}$ represents the Euclidean distance between the points normalized by minimum radius of curvature of the vehicle, and

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

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

$$\alpha = \psi_i - \tan^{-1}\left(\frac{y_f - y_i}{x_f - x_i}\right). \tag{5}$$

Each of the four possible distances can be calculated as follows [13]:

$$\begin{aligned} L_1 &= \sqrt{d^2 + 2 - 2\cos\alpha + 2d\sin\alpha} - \alpha, \\ L_2 &= \sqrt{d^2 + 2 - 2\cos\alpha - 2d\sin\alpha} + \alpha, \\ L_3 &= \sqrt{d^2 - 2 + 2\cos\alpha + 2d\sin\alpha} - \alpha - 2\mu - \gamma, \\ L_4 &= \sqrt{d^2 - 2 + 2\cos\alpha - 2d\sin\alpha} + \alpha + 2\nu - \gamma, \end{aligned} \tag{6}$$

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

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

This approach helps to avoid close vertexes from being connected and allows the union of distant vertexes, once it they not intersect with any obstacles. The consequence is that the algorithm converges rapidly in environments with a small number of obstacles.

Minimizing the function ρ gives an approximation to the nearest node $\mathbf{P}_{near}$ of the chosen random state. Now, according to the RRT algorithm, we must define the $\mathbf{P}_{new}$ vertex. In this 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 typical RRT calculation (determination of the u_{new} entries), by the use of an analytic function. The problem is that this may lead to critical points where obstacles may hinder the progress of the tree in the direction of the new point. In the event of such a collision, a random value can be added to ψ_{rand} in order to avoid the obstacle.

As we use Bézier curves to connect $\mathbf{P}_{near}$ and $\mathbf{P}_{new}$ vertexes, we can represent u_{new} as a vector composed by the eight control points of the seventh order Bézier curves, as follows

$$u_{new} = \mathbf{p}_k, \quad k = [0...7].$$

The last step is the creation of new state $\mathbf{P}_{new}$, which will be used to expand the tree. We consider this as the problem of computing a Bézier curve which allows vehicle navigation with limited and continuous curvature values. This curve can be shaped as a seventh order function:

$$\vec{p}(\tau) = \sum_{k=0}^{7} \mathbf{p}_k \binom{7}{k} (1-\tau)^{7-k}\tau^k; \quad \tau \in [0,1], \qquad (7)$$

where $\mathbf{p}_k = [x_k,\, y_k]$ is the k-th control point and τ is the parameter of the Bézier curve.

D. Smooth Path Calculation

In this section we present a method to calculate Bézier curves that fulfills the three constraint conditions mentioned before. The last one is less problematic, because, once calculated the Bézier we only need to verify collisions with the obstacles, and in positive case, eliminate the curve. The other two needs more sophisticated solutions, as follows.

We begin with an interesting characteristic of Bézier curves. They are infinity differentiable functions, which means that one curve has continuous curvature profiles. However, if we connect two or more curves in order to form a path $\vec{r}(t)$, we will have discontinuity accelerations in the connecting points, violating the second condition. Then, we must calculate Bézier curves which curvature values at it extreme points are equivalent, in order to keep the continuity.

According to [14], mathematically, the initial curvature of a Bézier function can be determined by means of Equation 8.

$$\kappa(t)|_{t=0} = \frac{(n-1)}{n}\,\frac{|\mathbf{p}_2 - \mathbf{p}_1|}{|\mathbf{p}_1 - \mathbf{p}_0|}\sin\sigma, \qquad (8)$$

where n is the Bézier order, and σ represents the angle between the vectors $\overrightarrow{\mathbf{p}_0\mathbf{p}_1}$ and $\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$. We can easily guarantee a null curvature value at the initial point by making these, collinear vectors. The same idea can be extended to the final curvature value, where are considered the vectors $\overrightarrow{\mathbf{p}_5\mathbf{p}_6}$ and $\overrightarrow{\mathbf{p}_6\mathbf{p}_7}$. Then, we always will have zero curvature value at the extreme points of each Bézier, making its composition with others curves ($\vec{r}$) a C^2 function, as required.

The main reason which led us to choose using a seventh order function is now clear. We must keep the three initial and three final control points aligned, so we need at least other two points to create a function with a point of inflexion, in order to make a more flexible path.

With this idea, six of the eight control points can be calculated by the following set of equations, which depends only of the initial and final configuration at each edge of the tree:

$$\begin{aligned}
\mathbf{p}_0 &= [x_i,\, y_i],\\
\mathbf{p}_1 &= \mathbf{p}_0 + k_0[\cos\psi_i,\, \sin\psi_i],\\
\mathbf{p}_2 &= \mathbf{p}_1 + k_0[\cos\psi_i,\, \sin\psi_i],\\
\mathbf{p}_5 &= \mathbf{p}_6 - k_7[\cos\psi_f,\, \sin\psi_f],\\
\mathbf{p}_6 &= \mathbf{p}_7 - k_7[\cos\psi_f,\, \sin\psi_f],\\
\mathbf{p}_7 &= [x_f,\, y_f],
\end{aligned} \qquad (9)$$

where k_0 and k_7 are gain factors that determined the vectors modulus.

There are two problems to be solved at this point. First, we must be able to calculate the remaining points, $\mathbf{p}_3$ and $\mathbf{p}_4$ of the Bézier curve. Second, we must choose the best values of k_0 and k_7 in order to comply with the curvature constraint represented by the third constraint condition.

The first one can be done by considering a particular set of Bézier curves, called Pythagorean Hodograph (PH) curves. The PH curves are a special kind of parametric functions that present many computational advantages over polynomial parametric curves in general. They were introduced in [15], where for the first time, PH curves of fifth order for the two-dimensional case were presented. An Hermite Interpolation algorithm was proposed in [16], where the author demonstrated that there exist four possible solutions for the curve in $\mathbb{R}^2$. The chosen solution is the one that minimizes the cost function [17] (bending energy function) based on the integral of the magnitude of the curvature function.

The PH provides several properties that can be considered advantageous in the path planning problem. The most relevant for this paper are: (i) uniform distribution of points along the curve, which contributes to the smoothness of the path; (ii) parametric speed (first order derivative) that provides a continuous velocity profile; and finally (iii) a capability to admit real-time interpolator algorithms for computer numerical control.

To find the remaining points, we should solve the following equation, modified from the Hermite Interpolation system

presented in [16]:

$$\mathbf{p}_3 = \mathbf{p}_2 + \tfrac{1}{5}\begin{bmatrix} u_0u_1 - v_0v_1 \\ u_0v_1 + u_1v_0 \end{bmatrix}^T,$$
$$\mathbf{p}_4 = \mathbf{p}_3 + \tfrac{1}{15}\begin{bmatrix} 2u_1^2 - 2v_1^2 + u_0u_2 - v_0v_2 \\ 4u_1v_1 + u_0v_2 + u_2v_0 \end{bmatrix}^T. \quad (10)$$

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, and are used in the traditional PH technique. They can be calculated as

$$\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \sqrt{\tfrac{5}{2}}\begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_1\| + \Delta x_1} \\ \mathrm{sgn}(\Delta y_1)\sqrt{\|\Delta\mathbf{p}_1\| - \Delta x_1} \end{bmatrix}, \quad (11)$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \pm\sqrt{\tfrac{5}{2}}\begin{bmatrix} \sqrt{\|\Delta\mathbf{p}_5\| + \Delta x_5} \\ \mathrm{sgn}(\Delta y_5)\sqrt{\|\Delta\mathbf{p}_5\| - \Delta x_5} \end{bmatrix}, \quad (12)$$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = -\tfrac{3}{4}\begin{bmatrix} u_0 + u_2 \\ v_0 + v_2 \end{bmatrix} \pm \sqrt{\tfrac{1}{2}}\begin{bmatrix} \sqrt{c + a} \\ \mathrm{sgn}(b)\sqrt{c - a} \end{bmatrix}, \quad (13)$$

where

$$\Delta x_k = x_{k+1} - x_k,$$
$$\Delta y_k = y_{k+1} - y_k,$$

and

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

The a, b and c parameters can be determined as:

$$a = \tfrac{9}{16}(u_0^2 - v_0^2 + u_2^2 - v_2^2) + \tfrac{5}{8}(u_0u_2 - v_0v_2) + \tfrac{15}{2}(x_5 - x_2),$$
$$b = \tfrac{9}{8}(u_0v_0 + u_2v_2) + \tfrac{5}{8}(u_0v_2 + v_0u_2) + \tfrac{15}{2}(y_5 - y_2),$$
$$c = \sqrt{a^2 + b^2}.$$

There are four solutions for the curve 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. Instead of minimizing the bending elastic energy function, as seen in [17], we choose the solution with the smallest maximum curvature. This allows curves with great curvature variance, which is more desirable for environments with obstacles.

Once the Bézier points are computed, we must guarantee that the generated curve does note violate the kinematic constraints of the vehicle, according to the first constraint condition. For this, we must determine the values of k_0 and k_7 in Equation 9. As there is no closed form solution, these values are increased iteratively, until $\vec{p}(\tau)$ becomes realizable. At this point, we just monitor the maximum curvature value of the curve.

IV. Experiments

Our technique was used to plan paths for a simulated small unmanned aerial vehicle. The robot was modeled as a 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 [10].

The AqVS presents the following characteristics: ρ_{min} about 30 meters, maximum cruising speed of approximately 50 km/h, and uncertainty of localization of 12 meters. The above values were determined using data from actual flights, considering a flight speed of approximately 50 km/h.

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

Figure 2 presents the evolution of a RRT composed by seventh order Bézier curves between its vertexes for the AqVS/UAV. As we can see, a few number of these vertexes is required to reach the goal configuration in this particular environment. Figure 3 shows the curvature profile of the path in the tree. It is possible to note that all union points between curves present null curvature values, which makes the acceleration profile of the vehicle to be continuous (but not differentiable).

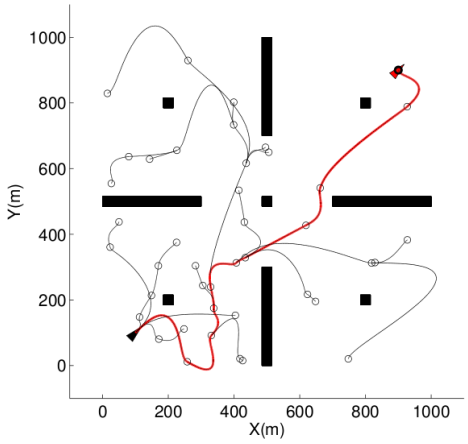

Fig. 2. Rapidly-exploring random trees using seventh order Bézier curves. The red line represents the final path at the tree.

We can see in Figure 2 that the use of analytical functions 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 vertexes.

Tables I(a) and I(b) present a comparison between these two techniques. In the first table we use a simple dynamic linear model (which is used at the RRT integration step) with the same velocity and curvature constraints to generate the tree. In order to build these two tables, we consider three different environments with 5, 20 and 100 obstacles (randomly distributed), and we execute 50 experiments in every one of them, taking the number of vertices used to expand the tree until it reaches the goal. The mean and standard deviation of vertices number of each experiment were organized into the next tables.

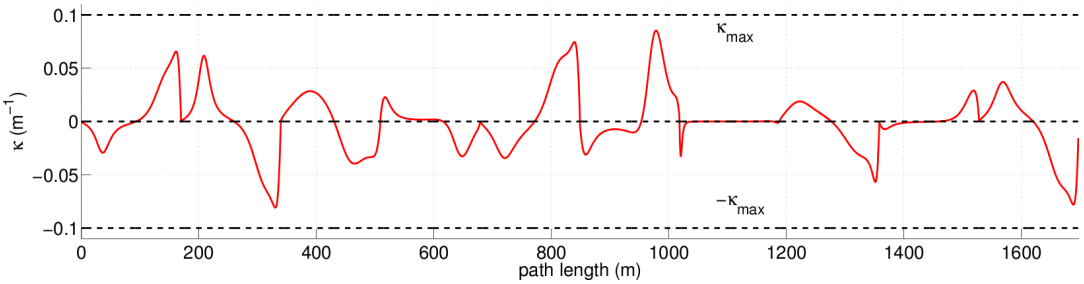

Fig. 3. Continuous curvature profile of the path presented at Figure 2.

TABLE I
TRADITIONAL RRT VERSUS RRT WITH 7^{th} ORDER BÉZIER CURVES.

Obstacles	Mean	St. dev.
5	204.9	108.9
20	243.6	122.4
100	272.2	152.7

(a) Traditional RRT

Obstacles	Mean	St. dev.
5	6.2	6.4
20	10.5	7.3
100	12.8	9.0

(b) RRT with 7^{th} order Bézier

As discussed previously, the number of vertexes necessary to define a path between simple (with few obstacles) and complex environments (with several obstacles) is significantly smaller when compared with the traditional technique, as described in [6].

V. CONCLUSION AND FUTURE WORKS

We have described a technique that allows the planning of paths for robots in environments with a variable number of obstacles. While the RRT technique 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 of 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 design 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 number of vertexes of the tree because the use of seventh order Bézier curves that enable connections between vertexes of the tree with unlimited range. In an environment with few obstacles, a very small quantity of vertexes (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 to the three-dimensional space, considering other kinematic constraints, such as the maximum torsion and maximum climb (or dive) angles. Initial results have shown that it is possible to use an extension of the Bézier curves in three dimensions, while maintaining most of the advantages described in this paper.

VI. ACKNOWLEDGMENT

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

REFERENCES

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

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

[3] 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.

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

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

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

[7] 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.

[8] 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 *Proc. of the IEEE American Control Conference (ACC)*, Minneapolis, Minnesota, USA, June 2006, pp. 2813–2818.

[9] K. Yang and S. Sukkarieh, "Real-time continuous curvature path planning of uavs in cluttered environments," in *Proc. of the 5th International Symposium on Mechatronics and its Applications(ISMA08)*, Amman, Jordan, May 2008.

[10] A. Alves Neto, D. G. Macharet, and M. F. M. Campos, "On the generation of trajectories for multiple uavs in environments with obstacles," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 4, pp. 123–141, 2010.

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

[12] 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.

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

[14] T. W. Sederberg, *Computer Aided Geometric Design*. Brigham Young University, April 2007, ch. 2, pp. 17–36.

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

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

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