Spline Trajectory Algorithm Development: Bézier Curve Control Point Generation for UAVs

Lauren R. Howell[1] and B. Danette Allen[2]
NASA Langley Research Center
Hampton, VA 23681

A greater need for sophisticated autonomous piloting systems has risen in direct correlation with the ubiquity of Unmanned Aerial Vehicle (UAV) technology. Whether surveying unknown or unexplored areas of the world, collecting scientific data from regions in which humans are typically incapable of entering, locating lost or wanted persons, or delivering emergency supplies, an unmanned vehicle moving in close proximity to people and other vehicles, should fly smoothly and predictably. The mathematical application of spline interpolation can play an important role in autopilots' on-board trajectory planning. Spline interpolation allows for the connection of Three-Dimensional Euclidean Space coordinates through a continuous set of smooth curves. This paper explores the motivation, application, and methodology used to compute the spline control points, which shape the curves in such a way that the autopilot trajectory is able to meet vehicle-dynamics limitations. The spline algorithms developed used to generate these curves supply autopilots with the information necessary to compute vehicle paths through a set of coordinate waypoints.

Nomenclature

B	=	B-spline curve
n	=	number of control points
P	=	control point values
t	=	internal knots
C	=	the Bezier curve

[1] Intern, NASA LaRC Autonomy Incubator, Univ of Alabama, lrhowell1@crimson.ua.edu, AIAA Student Member.
[2] Head, NASA LaRC Autonomy Incubator, MS 492, danette.allen@nasa.gov, AIAA Senior Member.

I. Introduction

A GREATER need for sophisticated autonomous piloting systems has risen in direct correlation with the progression of technology. Mission-driven, unpiloted, autonomous aerial vehicles must have a way of planning the path of the vehicle during real-time flight to promote robust functionality. The motivation for real-time path planning is derived from its possible mission applications. For example, unmanned aerial systems (UAS) could be capable of surveying unknown or unexplored areas of the world, collecting scientific data from regions in which humans are typically cannot or should not access e.g., rain forests, disaster areas), locating lost or wanted persons, or delivering emergency supplies to reote lcations or faster than currently possible. Mission requirements that consistently push vehicle platform limits in a rapidly developing field should be equipped with the tools that equally push the limits of improvement and optimization.

Besides the user expectancy to see a drone flying smooth paths, there are many benefits to flying curved trajectories. A common problem with many waypoint navigation systems is target overshooting. Often, these piloting systems will connect waypoints with a series of straight-line segments. Due to the dynamic constraints on the aerial vehicle, the UAV will often over-shoot the target waypoint in an attempt to change direction to fly to the next specified waypoint. This causes the vehicle to deviate from its specified path, thus creating a measurable difference between the specified path and the actual path flown. This is neither efficient nor safe, especially in the case of multivehicle coordinated flight, where, if a drone deviates from its projected flight path, it could enter the flight path of a neighboring drone. By instead connecting waypoints with curves that adhere to the dynamics of the particular vehicle, a trajectory that minimizes filght track deviation can be invoked. This secures the location that of the vehicle during real-time flight and ensures a safer flight environment.

The research on this topic that existed previously was utilized as a starting point and was extended for optimal results in specific flight application. Several project studies were conducted to better understand the problem of spline construction while the mathematic foundation was laid through textbooks and several university papers. Through the careful selection of control points for Bèzier splines, curve shapes can be manipulated in such a way that waypoints can be connected by a smooth path.

This paper will go through the defining, implementing, and testing of the algorithms used to select the location of the control points, and explore how this research applies to flight in Euclidean space.

II. Research

The elements which must first be understood to compose a spline are the Bèzier Curve equation, B-splines, and the general notion of a spline itself. A spline is a set of piecewise-polynomial functions of degree n that connect curves in space. The different levels of continuity at the curves' joints ensure different "smoothness" properties of the overall path.

There are several different types of splines, which suit different purposes. The type of spline that is used is selected based on its fulfillment of control parameters such as the boundary and joint (or knot) conditions, or degrees of freedom[3]. These conditions are crucial and serve as the basis of the criteria that must be met. The utilization of such a mathematical tool is trajectory planning (as seen being implemented in many projects such as "Task-Space Trajectories via Cubic Spline Optimization[4]" and *Planar Spline Trajectory Following for Autonomous Helicopter*[5]). The formula for the basis spline, or B-spline, is[3]

$$B(t) = \sum_{i=0}^{n} P_i N_{i,p}(t) \quad (1)$$

where

B(t) is the B-spline curve
n is the variable which denotes the number of control points
P is the variable that denotes the control point values
t is the parameter variable that represents the internal knots

and

$$N_{i,0}(t) = \begin{cases} 1 \; if \; t_i < t < t_{i+1} \\ 0 \; otherwise \end{cases}$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} + t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t)$$

In general, the Bèzier Curve is a polynomial equation of degree *n*, completely defined by a set of control points and acts as a function of its parameters. The formula used to interpolate curves between the waypoints is the Bèzier Curve. The Bèzier Curve is a special case of the B-spline that contains no internal knots, and has the formula[3]

$$C(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \quad (2)$$

where:

C(t) is the curve
n is variable that denotes the number of control points, $\binom{n}{i} = \frac{n!}{i!(n-i)!}$
P is the variable that denotes the control point values
t is the parameter variable whose values range $0 \leq t \leq 1$

and

$$b_{i,n}(t) = \frac{n!}{i!\,(n-i)!} (1-t)^{n-i} t^i$$

is defined as the Bernstein Basis[3].

So therefore the equation can then be rewritten as

$$C(t) = \sum_{i=0}^{n} b_{i,n}(t)\, P_i \quad (3)$$

All the curves must be joined with at least C^0 continuity to guarantee that the last point of one curve be the first point of the next curve. For assurance of smoothness, additional constraints on the curve were added to ensure C^1 continuity (equality of the tangents at the junction of the curves) and C^2 continuity (equality of curvature at the junction of the curves). Ordinarily the Bèzier Curve loses local control for C^2 continuity because the control points' locations are dependent upon each other. For application, the control points must first be selected to define the shape of the curve. The B-spline has both local control and C^2 continuity but does not retain the interpolation ability of the Bèzier.

The Bèzier Cubic-Spline was chosen because of its infinite continuity properties. The control points had to be selected to retain C^1 continuity so the velocity would be continuous at the junction of each curve, which serves as the application of equality in curves' tangent line slopes. The Bèzier Curve was thusly selected at the trajectory base with specific recognition of the importance of the selection of the position of control points.

Some modifications to conventional mathematical methods had to be made to meet the specific need of controlling and specifying velocity, not only position. Two UCLA Department of Mathematics papers[1-2] were extended to meet the needs of this mission by introducing an algorithm that served as the basis for the final algorithm used to calculate the control points for the goal trajectory.

The problem faced is actually the reverse of how people typically handle this situation. Rather than knowing the control points and interpolating the curve between them, it is rather the waypoints of the trajectory that are known and the control points that need to be found. To is a two-step process. In the first step, the more general B-spline control points are found. Once these points are known, the B-spline control points are used to then find the four needed control points for the cubic Bèzier-spline[1-2].

Given a set of target waypoints $P_0 \ldots P_n$, the B-spline control points, $B_0 \ldots B_n$, must first be calculated as an intermediate step which will then set up for the second step to find the Bèzier control points.

An observation is that $P_0=B_0$ and $P_n=B_n$. A second observation is that each point, P, is a set of three numbers that represents a position in space. The rest of the B-spline points can be calculated by the following set of linear equations:

Let S represent the series of values of P and B represent the series of values of the B-Spline control points, then

$$S_i = \frac{1}{6}B_{i-1} + \frac{2}{3}B_i + \frac{1}{6}B_{i+1} \quad (4)$$

for all $i = 0 \ldots n$.

This set of linear equations can be re-written and solved by treating P as the knowns and B as the unknowns which yields (shown for $n=5$)[1]:

$$\begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} = \begin{bmatrix} (6S_1 - S_0) \\ 6S_2 \\ 6S_3 \\ (6S_4 - S_5) \end{bmatrix} \quad (5)$$

To extend this work so that it was applicable to the project, an additional degree of freedom had to be added by the consideration of velocity, as well as two additional B-spline control points. Therefore, for n amount of given waypoints, the matrix will be (n+2) x (n+2) with n+2 B-spline control points. This changes the matrix equation to be:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ B_7 \end{bmatrix} = \begin{bmatrix} S_0 \\ \boldsymbol{V}i \\ 6S_1 \\ 6S_2 \\ 6S_3 \\ \boldsymbol{V}f \\ S_4 \end{bmatrix} \quad (6)$$

Where $\boldsymbol{V}i$ and $\boldsymbol{V}f$ are the initial and final velocity vectors.

The addition of the vectors as degrees of freedom allows the direction of motion of the end of one curve to be the initial direction of the beginning of the next curve. This, in turn, produces the smoothness of the entire trajectory such that the vehicle is better able to stay on course. Once the B-spline control points are found, the Bèzier control points can be calculated by the following:

Let C represent the series of values of Bèzier control points, then

$$\begin{aligned} C_{i1} &= S_{i-1} \\ C_{i2} &= \frac{2}{3}B_{i-1} + \frac{1}{2}B_i \\ C_{i3} &= \frac{1}{3}B_{i-1} + \frac{2}{2}B_i \\ C_{i4} &= S_i \end{aligned} \quad (7)$$

which is repeated for every set of waypoints, including the two interpolated additional waypoints. Once the Bèzier control points have been found, these can then be transferred into the Bèzier Curve equation.

This equation is then complemented by several other algorithms that apply velocity profile and time segmentation for each position to comply with the dynamics of the vehicle. This information is then compiled and finalized into the trajectory.

III. Implementation

With the algorithm solved, the next step was to integrate this math into software that would output commands to control the flight of an Unmanned Aerial Vehicle (UAV). The algorithm was converted into the computer-based C++ program so that these computations could be made rapidly and. This allows for the input of any number of waypoints

to generate a large quantity of trajectory curves. Once the program was successfully built and compiled, it was integrated into the software that controlled UAV flight.

The user is able to input information into the program in the form of sets of coordinates in space which represent the waypoints through which the user desires the UAV to travel. The program will read in the waypoints and subsequently calculate one curve for every two waypoints. For every curve, the program calculates four control points to garuntee complete continuity and smooth transition between each curve. The sets of curves are then discretized and output to a file which can be read in as a trajectory through the external piloting system.

The applications of such a calculation process can even expand the way that the user delivers commands for flight. For example, using motion sensor technology, a user could define a path in three-dimensional space using gestures or voice commands that can be sent as a file into the program as waypoints. The program will then be able to go through the previously discussed steps to successfully generate the desired trajectory while maintaining safety and accuracy. Because the algorithm is able to take in such large quantities of raw data and return reliable, three-dimensional curve solutions, the application of this process of defining flyable paths in space opens the door to new and innovative types of autopiloting.

IV. Testing

To test the algorithm now embedded in the C++ program, different approaches were used for verification of the algorithm's accuracy.

A. Graphing

The discretized points of the Bèzier Curve were plotted to verify that the curve was continuous, smooth, and passed through all of the target waypoints. Once this was confirmed, measures were taken to prove the accuracy of the algorithm. Sample waypoints from a known trajectory were taken and fed into the developed algorithm. The plots of the known trajectory were then compared to the trajectory output by the algorithm, as shown in Figure 1.

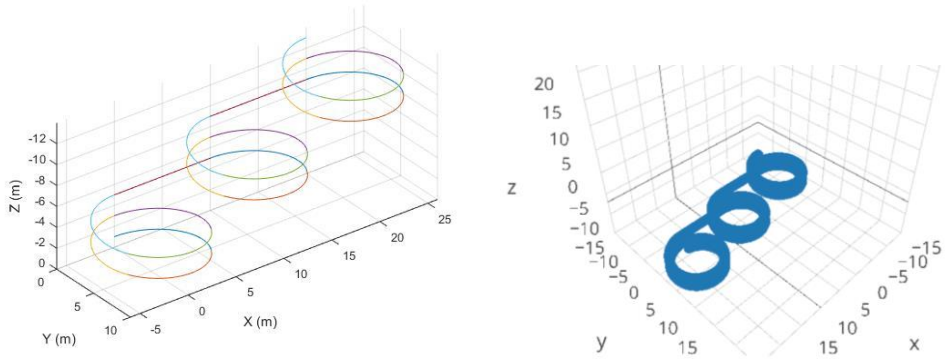

Figure 1: Comparison of commanded trajectories

B. Flight

Finally, to confirm the dynamic capabilities of the algorithm, waypoints for a known trajectory were chosen and sent through the algorithm, which returned the control points for the Bèzier curves to be flown. Small Crazy-Fly quad-copters were selected to fly these curves, along with larger AR Parrot Drone quad-copters. The use of different sized vehicles demonstrates the versatility of the software and the universal application of the algorithm developed. With the success of the flight test of the trajectory shown below, the accuracy of the algorithm was further confirmed. Methods of flight testing included both direct file input of selected coordinate waypoints as well as user generated path using gestures as discussed above.

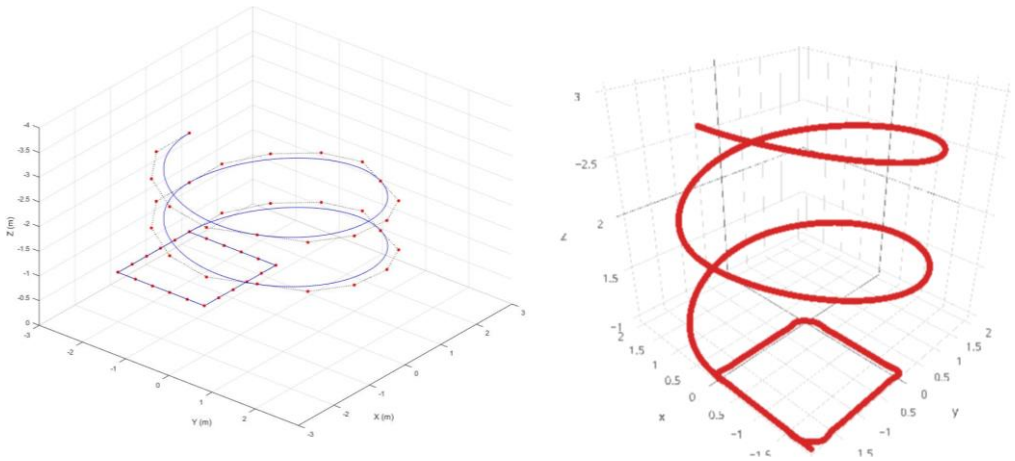

Figure 2: Comparison of as-flown trajectories

V. Conclusion

The algorithm developed to create the control points for Bezier curves aids in trajectory planning for UAV's. Through different methods of testing, the system has been shown to produce valuable, working data. The final product of the algorithm described in this paper is a series of control points, which are coupled with an additional system to complete a trajectory through a given set of waypoints.

An addition that could be made to the work thus far is the ability to more robustly analyze that data produced when compared to a trajectory that has already been flown with the motivation of generating reproducible paths for use in future missions..

Acknowledgments

I would like to acknowledge the assistance of Danette Allen, my mentor, without whom none of this would have been possible. I would also like to acknowledge my program coordinator, Elizabeth Ward, for all the work that she did in connecting me and sending me in the right direction. In addition I would like to thank my fellow interns, Javier Puig Navarro, Billal Mehdi, Gilbert Montague, and Meghan Chandarana for their assistance in my academic pursuit and algorithm development and working support of the Bezier Control Point algorithm testing. Finally I would like to acknowledge Loc Tran, Jim Neilan and all of my co-workers at the NASA Langley Research Center Autonomy Incubator for their assistance and support ofmy work.

References

[1]Baker, A. Kirby, "Math 149 S99 PP Nonuniform Splines", *UCLA Department of Mathematics* [online database] URL: http://www.math.ucla.edu/~baker/pdf/pp_nonuniform.pdf [Cited May-July 2015].

[2]Baker, A. Kirby, "Math 149 W02 DD Cubic Spline Curves", *UCLA Department of Mathematics* [online database] URL: http://www.math.ucla.edu/~baker/149.1.02w/handouts/dd_splines.pdf [Cited May-July 2015].

[3]Farouki, T. Rida, *Pythagorean-Hodograph Curves: Alegbra And Geometry Inseparable*, Springer-Verlag, Berlin Heidelberg 2008.

[4]J. Zico Kolter and Andrew Y. Ng, "Task-Space Trajectories Via Cubic Spline Optimization", *ACM Digital Library* [online database] URL: http://dl.acm.org/citation.cfm?id=1703833 [Cited May-July 2015].

[5]Kale Harbick, James F. Montgomery and Gaurav S. Sukhatme, *Planar Spline Trajectory Following for an Autonomous Helicopter,* Robotic Embedded Systems Laboratory, Center for Robotics and Embedded Systems Department of Computer Science, University of Southern California, Los Angeles, CA.