Quintic Splines for FTC

Ryan Brott

1 Introduction

In this paper, we explore the use of quintic splines for more sophisticated robot pathing in FTC. Traditionally, FTC robot autonomous motion consists of linear movements (including holonomic drive strafing) and point turns. Although these simple path primitives generally suffice, they are not time-optimal under reasonable kinematic constraints, especially for nonholonomic (e.g., tank/differential) drivetrains. That is, the fastest route between two poses (Cartesian position and heading combined) generally in these situations is not two point turns and a line.[1] To address this, we propose quintic splines to achieve quick, smooth motion on the field.

In the first part of the paper, we will describe the problem in depth and give motivation for splines. Next, we explore some of the mathematics behind quintic splines including interpolation and arc length parametrization.

2 Problem

In FTC, it's standard to have a sequence a poses that you want the robot to follow in autonomous (although pre-planned motions can also be utilized in TeleOp). For example, in the Relic Recovery game, a common movement task may be moving from one pose on the balancing stone to another in front of the cryptobox. This is traditionally handled with a series of straight lines and turns that are executed with 1D position PID controllers and potentially motor-based velocity PID (e.g., `RUN_USING_ENCODER`).

For many game tasks and teams, this is a perfectly viable approach. However, in scenarios where speed is desirable (such as the effectively unlimited scoring potential in Relic Recovery autonomous), conventional methods tend to become less effective. Higher speeds usually lead to greater wheel accelerations and slippage, hurting odometry. Additionally, the feedback controllers may have more chaotic transient behavior or decreased steady-state performance.

In this case, motion profiling can be used to achieve higher speeds without sacrificing accuracy by observing the robot's physical constraints (e.g., maximum velocity, acceleration).[2]. However, this is only part of the solution; in fact, motion profiling amplifies the

[1] A keen reader will realize that this is not necessarily the case for holonomic drives. All pose-to-pose movements can be executed with a combination of strafing and rotating. Nevertheless, splines can still be of use. For one, traveling along the tangent can reduce odometrical errors accrued from translating on the lateral axis. Additionally, splines can help navigate around obstacles in situations where a piecewise linear path would normally be required.

[2] For a good introduction to motion profiling, see the canonical talk by mentors from FRC teams 254 and 971: `https://www.youtube.com/watch?v=8319J1BEHwM`

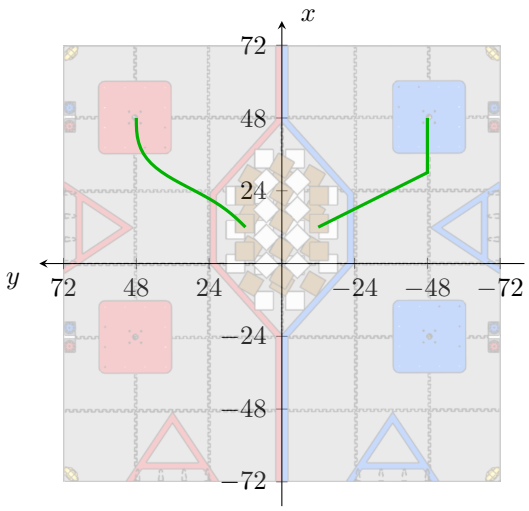

Figure 1: Example coordinate system for describing positions on the Relic Recovery field. This right-handed frame can be extended to three dimensions with the positive z-axis protruding from the origin. The path on the blue side is a typical autonomous route seen in FTC (don't forget there's a turn between the line segments — three profiles in total). The path on the red side is a quintic spline version of roughly the same move. The spline path is 1.6 times faster than the conventional path with reasonable drive constraints.

discontinuities between the elements of a typical pose-to-pose path. Each transition between rotating and translating demands a full deceleration-acceleration cycle.

To eliminate this unnecessary pause, we will combine the rotation and translation into a single smooth curve. Although many curves suffice, splines are typically employed for this purpose. For simplicity, this paper only considers polynomial splines[3]. Quintic splines in particular were selected as a proper balance between continuity and curvature although the methods described can be easily extended to polynomials of arbitrary degree.

3 Preliminaries

3.1 Coordinate System

Although the methods presented here do not depend on a specific coordinate system, it's essential to establish a consistent reference frame for describing robot movement. Figure 1 shows an example coordinate system for the Relic Recovery field. Most importantly, it is right-handed, and its origin is in the center of the field. Though many teams avoid an explicit coordinate-based representation of their robot motion, it is an essential tool for unambiguously describing more complex paths.

[3] `http://www.ryanjuckett.com/programming/biarc-interpolation/` is a great introduction to biarc splines.

3.2 Vectors and Parametric Curves

To fully understand the following mathematics, it helps to grasp a little vector calculus (don't be scared — only basic knowledge of single-variable calculus is required). Recall that a Euclidean vector $\mathbf{v}$ in n dimensions ($\mathbf{v} \in \mathbb{R}^n$) uniquely codifies a single point in the space (commonly represented as an arrow from the origin to the point). This vector can be decomposed into n components:

$$\mathbf{v} = (x_1, x_2, \ldots, x_n) = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \ldots + x_n\mathbf{e}_n$$

The length of a vector is given by its norm:

$$|\mathbf{v}| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

Aside from vector addition and scalar multiplication, two additional operations are commonly defined between Euclidean vectors. First, the dot or inner product is defined as the scalar sum of the element-wise products of two vectors:

$$\mathbf{v} \cdot \mathbf{w} = v_1w_1 + v_2w_2 + ... + v_nw_n$$

Note that the norm can be alternatively expressed as $|\mathbf{v}| = \sqrt{\mathbf{v} \cdot \mathbf{v}}$. Second, the cross product is defined for three-dimensional vectors as the pseudo-determinant of this quasi-matrix:

$$\mathbf{v} \times \mathbf{w} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

Two-dimensional curves can be represented as graphs of one-dimensional functions ($y = f(x)$). However, this puts unnecessary constraints on the set of possible curves (e.g., no two points on the curve can share the same x-value, no self-intersections), especially in higher dimensions. Instead, we will represent curves as a series of vectors that together trace out the shape. This is a parametric curve $\mathbf{r}(t)$ where t is a real number ($\mathbf{r} : \mathbb{R} \to \mathbb{R}^n$). Intuitively, a $\mathbf{r}(t)$ can be thought of as the path traveled over time by a fly that is located at $\mathbf{r}(t_0)$ at the instant $t = t_0$.

Parametric curves can be represented as a vector of single-variable functions:

$$\mathbf{v}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$$

As one might expect, $\mathbf{r}(t)$ can be differentiated component-by-component:

$$\mathbf{v}'(t) = (x_1'(t), x_2'(t), \ldots, x_n'(t))$$

Derivatives of dot products and cross products can also be computed:

$$\frac{d}{dt}\Big[\ \mathbf{r}_1(t) \cdot \mathbf{r}_2(t)\ \Big] = \mathbf{r}_1(t) \cdot \mathbf{r}_2'(t) + \mathbf{r}_1'(t) \cdot \mathbf{r}_2(t)$$
$$\frac{d}{dt}\Big[\ \mathbf{r}_1(t) \times \mathbf{r}_2(t)\ \Big] = \mathbf{r}_1(t) \times \mathbf{r}_2'(t) + \mathbf{r}_1'(t) \times \mathbf{r}_2(t)$$

A parametric curve is said to be C^n if its nth-order derivatives are defined and continuous everywhere on its domain (n here is irrespective of the Euclidean space). For the rest of the paper, only two-dimensional parametric curves are considered ($\mathbf{r}(t) = (x(t), y(t))$).

4 Interpolation

Quintic splines consist of a series of segments assembled together into a single piecewise curve. Each of the segments is a parametric curve with a quintic polynomial for each component. The ith segment of a two-dimensional quintic spline of n segments can be represented as the following:

$$\begin{cases} x^{(i)}(t) = a_x^{(i)}t^5 + b_x^{(i)}t^4 + c_x^{(i)}t^3 + d_x^{(i)}t^2 + e_x^{(i)}t + f_x^{(i)} \\ y^{(i)}(t) = a_y^{(i)}t^5 + b_y^{(i)}t^4 + c_y^{(i)}t^3 + d_y^{(i)}t^2 + e_y^{(i)}t + f_y^{(i)} \end{cases} \qquad \text{where} \quad 0 \le t \le 1$$

Now the goal of interpolation is to "fit" these polynomials between a series of $n+1$ points (commonly referred to as knots) labeled (x_i, y_i). To accomplish this, we can impose the conditions $x^{(i)}(0) = x_i$ and $x^{(i)}(1) = x_{i+1}$ (and correspondingly for $y^{(i)}(t)$). Additionally, to ensure greater continuity, we also match the first and second derivatives at each knot point[4]:

$$\left.\frac{dx^{(i)}}{dt}\right|_{t=0} = x_i'$$
$$\left.\frac{dx^{(i)}}{dt}\right|_{t=1} = x_{i+1}'$$
$$\left.\frac{d^2x^{(i)}}{dt^2}\right|_{t=0} = x_i''$$
$$\left.\frac{d^2x^{(i)}}{dt^2}\right|_{t=1} = x_{i+1}''$$

(and analogously for y)

Collectively, these constraints guarantee that the overall spline will be (by construction) C^2. They also fully define the polynomial coefficients for each spline segment. To actually compute the coefficients, we simply need to solve the linear system with following matrix representation:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 5 & 4 & 3 & 2 & 1 & 0 \\ 20 & 12 & 6 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \\ e_x \\ f_x \end{bmatrix} = \begin{bmatrix} x_i \\ x_i' \\ x_i'' \\ x_{i+1} \\ x_{i+1}' \\ x_{i+1}'' \end{bmatrix}$$

This equation can then be solved to yield the coefficients with your favorite matrix library (or put in row echelon form to yield a set of back-substitutable equations).

[4] This is generally the best choice for splines in the context of path planning although other schemes are occasionally employed. For instance, one can instead force the third and fourth derivatives to equal zero.

5 Reparametrization

5.1 Arc Length

Now that we've interpolated the spline segments, we have to join together all of the spline segments into a single piecewise function of a unified variable. To accomplish this, we're going to reparametrize $\mathbf{r}(t)$ from $t \in [0, 1]$ to the arc length parameter s. Although it sounds complex, s is just the true displacement along the path, and it traces out the curve with unit speed ($|\mathbf{r}'(s)| = \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2} = 1$). This procedure is completely generic, allowing you to combine arbitrary parametric curve with (basically) the same knot derivatives.

To shift from t to s, we first have to define $t(s)$. This is difficult to represent analytically although the inverse function $s(t)$ is relatively simple:

$$s(t) = \int_0^t |\mathbf{r}'(\tau)| d\tau = \int_0^t \sqrt{\left(\frac{dx}{d\tau}\right)^2 + \left(\frac{dy}{d\tau}\right)^2} d\tau$$

In practice, $t(s)$ can be obtained by numerically evaluating the above integral and stopping when the integral sum reaches s.

By computing $t(s)$, we can determine $\mathbf{r}(s)$ by composition: $\mathbf{r}(s) = \mathbf{r}(t(s))$. Next, differentiating yields the new parametrized derivative:

$$\begin{aligned}
\mathbf{r}'(s) &= \mathbf{r}'(t)\ t'(s) \\
&= \frac{\mathbf{r}'(t)}{s'(t)} \\
&= \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|}
\end{aligned}$$

This makes sense intuitively as $\frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|}$ is just a normalized version of $\mathbf{r}'(t)$ that satisfies the condition $\mathbf{r}'(s) = 1$. We similarly obtain the new second derivative by differentiating again[5]:

$$\begin{aligned}
\mathbf{r}''(s) &= \frac{\mathbf{r}''(t)}{|\mathbf{r}'(t)|^2} - \frac{\mathbf{r}'(t)\ \mathbf{r}''(t) \cdot \mathbf{r}'(t)}{|\mathbf{r}'(t)|^4} \\
&= \frac{\mathbf{r}''(t)\ \mathbf{r}'(t) \cdot \mathbf{r}'(t) - \mathbf{r}'(t)\ \mathbf{r}''(t) \cdot \mathbf{r}'(t)}{|\mathbf{r}'(t)|^4} \\
&= \frac{\mathbf{r}'(t) \times (\mathbf{r}''(t) \times \mathbf{r}'(t))}{|\mathbf{r}'(t)|^4}
\end{aligned}$$

5.2 Trajectory

Now, all of the spline segments and various other parametric curves are combined into a single curve $\mathbf{r}(s)$. However, this still needs to be combined with the motion profile to yield the robot's kinematic state over time.

Let $s(t)$ be the generated motion profile (note: this t actually refers to time; it's different

[5]This derivation relies on the BAC-CAB identity: $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a} \cdot \mathbf{c}) - \mathbf{c}(\mathbf{a} \cdot \mathbf{b})$.

from the t used earlier). Now the velocity and acceleration of the robot can be determined:

$$\begin{aligned}
\mathbf{v}(t) &= \frac{d}{dt}\Big[\ \mathbf{r}(s(t))\ \Big] \\
&= \mathbf{r}'(s(t))s'(t) \\
\mathbf{a}(t) &= \frac{d}{dt}\Big[\ \mathbf{r}'(s(t))s'(t)\ \Big] \\
&= \mathbf{r}''(s(t))[s'(t)]^2 + \mathbf{r}'(s(t))s''(t)
\end{aligned}$$

6 Heading

The discussion in the previous sections has been limited to the translational components of the path. This, of course, must be accompanied by some sort of angular motion. For holonomic drives, the heading can be controlled completely independently. In this case, heading can be treated as a third independent path component that can be specified by an arbitrary parametric curve.

However, for nonholonomic drives, the heading is constrained to the direction of travel. For parametric curves, this direction is given by the vector $\mathbf{r}'(t)$:

$$\theta(t) = \arctan\frac{y'(t)}{x'(t)}$$

In practice, $\theta(t)$ is computed with the two-argument version of arctan, eliminating issues arising from signs and $x'(t) = 0$. Note that C^0 heading requires C^1 $x(t)$ and $y(t)$. This is the primary reason quintic splines were selected as they guarantee C^2 translation and C^1 rotation.

7 Conclusion

This paper discussed the motivation for splines in FTC and some mathematics for generating basic quintic splines. The author hopes this will help further the proliferation of advanced motion planning techniques in FTC.