Investigation on mathematical method used to construct curves in computer systems

Daekun Kim

April 28, 2018

Word count: 1447

Contents

1 Introduction

In the modern era, computer is one of the tools that are having the most influence in our lives. It is commonly used to design different products, such as cars and airplanes, or in the movie industry, such as producing scenes with computer graphics and animations. In these fields, ability to form and display smooth, realistic curvatures is essential. However, computers are digital, and they can only process discrete information. This nature of computers makes it seem to be only able to render discrete and pixelated shapes that cannot imitate natural curves. Yet, they seem to be able to easily generate many kinds of curved lines and surfaces, as shown in many computer-generated animations and products produced by computer-controlled machineries. This provoked my interest in how mathematics may play a role in achieving such seemingly impossible task.

This paper aims to investigate the mathematics behind how a computer system generates smooth curves. One of the most prominent methods used to achieve this is the *Bézier curve*. It is used in many popular photo-editing and animating softwares, such as the Adobe Flash®, Blender™, Autodesk Maya®, and more (*Bezier curves*, 2017; *Bezier Pen Tool - Flash Professional CS5*, 2017). This paper aims to clarify the journey that it took to come into being and be widespread to be utilized in numerous fields. Specifically, this paper describes the mathematical development of the Bézier curve, which starts from a rudimentary linear interpolation, repeated linear interpolation, the De Castaljau Algorithm, and finally the Bézier curve itself. Then, the properties of the Bézier curve that made it to be one of the most popular tool in curvature rendering in computer systems will be investigated.

2 Linear Interpolation

Straight lines are one of the easiest shapes one can imagine and draw, because of their simplicity to understand and manipulate. Such holds true even for computers. Graphically, drawing straight lines is the most rudimentary job a computer can perform. However, because it is so easy to render, it forms the fundamentals of how computers can render sophisticated curves and surfaces onto a screen.

Lines can be thought of as the shortest path that connects two points. A *linear interpolation*, then, is calculating which position on the line one would arrive on if one was to travel certain distance or time. Mathematically, given two points, (x_0, y_0) and (x_1, y_1) with $x_0 < x_1$, a linear interpolation is construction of a new set of y from x values within $[x_0, x_1]$. The formula of linear interpolation can be given as a relationship of the slopes of the lines connecting the starting and end point with that of line connecting the interpolated point with the starting point, or

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{1}$$

Eq. (1) is stating that, if a person was to travel over a line, the ratio of the distances at which one travelled vertically over the horizontal distance travelled must be the same no matter the position he or she lies on the line.

Example 2.1. Fig. 2.1 shows an example of linear interpolation between $(1, 1)$ and $(3, 2)$.

The equation of linear interpolation in Fig. 2.1 can be defined as following:

$$\frac{y - 1}{x - 1} = \frac{2 - 1}{3 - 1}, 1 \leq x \leq 2 \tag{2}$$

The line shown in Fig 2.1 is called the **interpolant**, which can be thought of, literally, as a shortest path between two points. Solving Eq. (2) for x gives us the following:

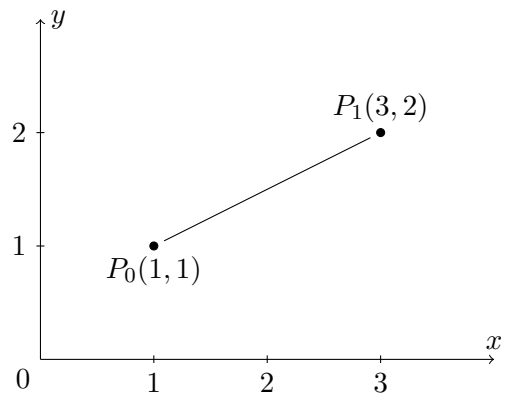

Figure 2.1: Linear interpolation between two points

$$y = \frac{1}{2}(x-1) + 1,$$
$$y = \frac{1}{2}x + \frac{1}{2} \tag{3}$$

In fact, this is now in slope-intercept form of an equation of a linear function, $y = mx + b$! This signifies that linear interpolation is nothing more than a linear function with a restricted domain between two points.

3 Parametric Equations

The previous section showed that linear interpolation is essentially plotting a point on a line using a linear function. However, such is not possible with our original definition of linear interpolation when the line cannot be defined using a function. For instance, consider a straight vertical line as shown in Fig 3.1.

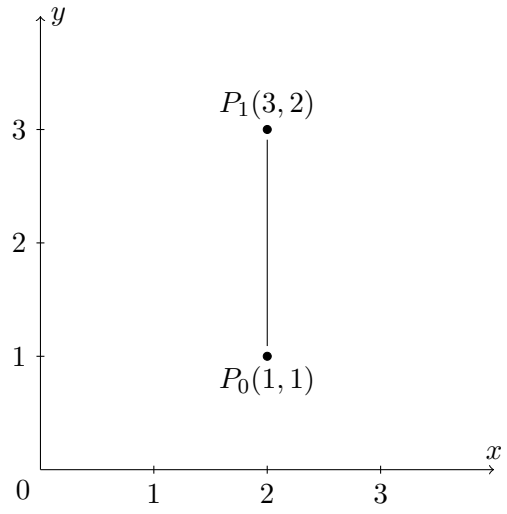

Figure 3.1: A vertical linear interpolation

In order for a line to be defined by a function, it must pass the Vertical Line Test, which states that there can only be one and only one y value for each x value. The line shown in Fig 3.1 obviously does not pass this condition, since there are infinitely many y values for $x = 2$.

However, by redefining linear interpolation with a set of parametric equations, one can easily overcome this apparent problem. Instead of defining an interpolation in respect to x, we can do so with another variable t within $[0, 1]$, and define different equations for the values of y and x with no explicitly direct relationship between them. This new variable t can be thought of as the time, or the ratio of how much of the path one have travelled from the initial point.

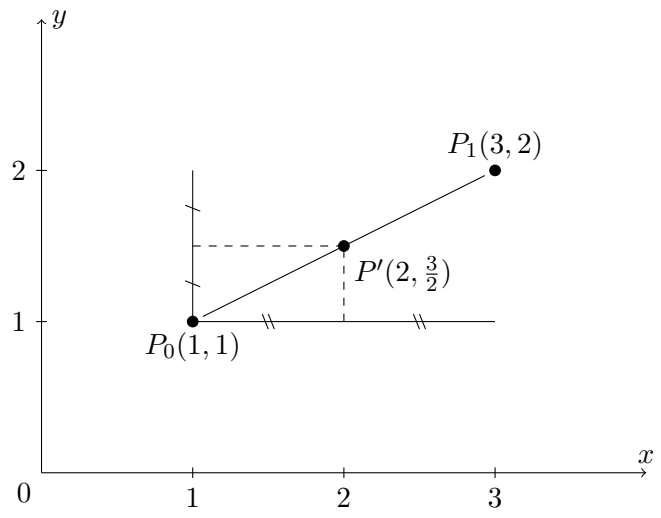

Figure 3.2: Parametric linear interpolation for $t = 0.5$

Example 3.1. Fig. 3.2 shows a linear interpolation at $t = 0.5$. At $t = 0.5$, the resulting point at halfway between P_0 and P_1 in both directions of the x- and y-axes. This can be thought of as a driver having travelled 50% of the path.

Example 3.2. In the same way, at $t = 0.25$, the resulting points are one-quarter of the way between the initial and the terminal points in both axes, which is like travelling 25% of the path since the starting point.

As shown in the two examples above, the t value is analogous to the ratio in which the person has travelled from the initial point that a GPS device would let one know.

As such, a linear interpolation between (x_0, y_0) and (x_1, y_1) can be parametrically defined as following:

$$x(t) = x_0 + (x_1 - x_0)t \tag{4}$$

$$y(t) = y_0 + (y_1 - y_0)t \tag{5}$$

Note that at $t = 0$, the interpolated point is at (x_0, y_0), the initial point, and at $t = 0$, it is at (x_1, y_1), the terminal point. This can be considered as being at the initial point

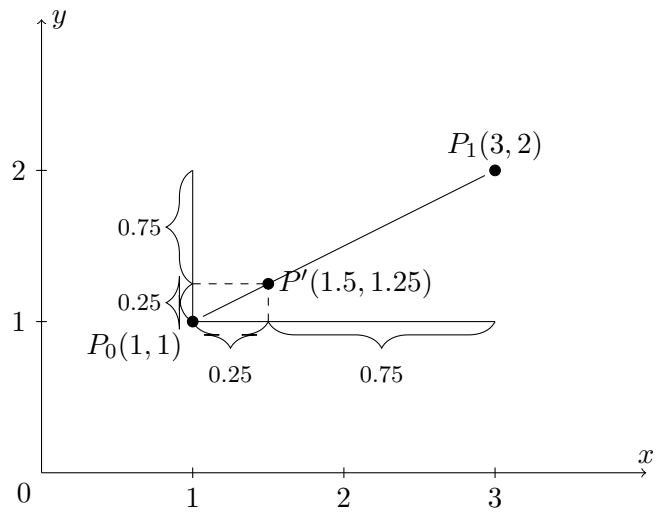

Figure 3.3: Parametric linear interpolation for $t = 0.25$

when one has travelled 0% of the path, and as being at the terminal point when one has travelled 100% of the path.

Parametric equations (4) and (5), however, looks to be very similar. In fact, it can be combined to form a 2-dimensional (2D) vector equation. Let P_0 and P_1 be position vectors, where $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$. Then,

$$P(t) = P_0 + (P_1 - P_0)t, \tag{6}$$

where $P(t)$ is a function that linearly interpolates a point between P_0 and P_1 for $t \in [0, 1]$. Analogously, this is a vector addition between the initial location and the path a driver has travelled so far at time t. Eq. (6) can be rearranged as following:

$$P(t) = P_0(1 - t) + P_1 t, \tag{7}$$

Eq. (7) further emphasizes the analogous nature of t being the "ratio" (Prautzsch, Boehm, & Paluszny, 2013) of the resulting point P with respect to the initial and final points. More specifically, the ratio $t : (1 - t)$ equals to the ratio of the distance from the current point P with the initial point P_0 and the final point P_1, respectively, as shown in Fig. 3.4.

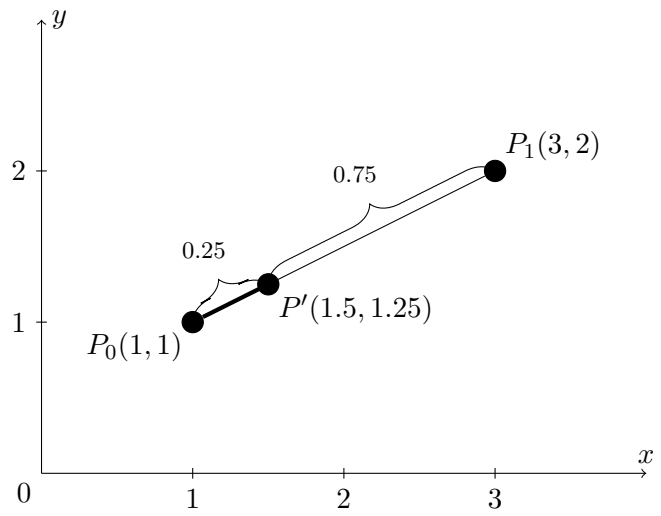

Figure 3.4: Ratio of $(1-t)$ and t for $t = 0.25$

References

Bezier curves. (2017, December). Retrieved from https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya-Modeling/files/GUID-5A399D05-9271-4517-8370-0533DC468277-htm.html.

Bezier Pen Tool - Flash Professional CS5. (2017, April). Retrieved from https://helpx.adobe.com/flash/kb/bezier-pen-tool-flash-professional.html.

bubba (https://math.stackexchange.com/users/31744/bubba). (2015, December). *Bezier curve coefficients intuition.* Mathematics Stack Exchange. Retrieved from https://math.stackexchange.com/q/1589350.

Prautzsch, H., Boehm, W., & Paluszny, M. (2013). *Bézier and B-spline techniques.* Springer Science & Business Media.

(bubba (https://math.stackexchange.com/users/31744/bubba), 2015)