Procedurally Generated Handwriting

Sam Pollard
Western Washington University

CS 580 – Fall 2015 – Geoff Matthews
December 2, 2015

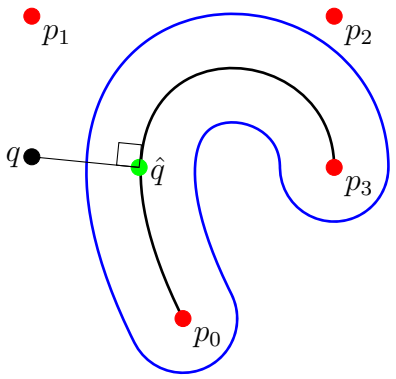

Figure 1: A sample bezier curve with its surrounding border

1 Introduction

To model a handwritten language we assume the hand draws curves with thickness ϵ which may change based on physical attributes of the curve. For a fountain pen, the thickness of the stroke depends on the minimum and maximum widths of the nib, the angle of the nib to the paper, and tangent of the stroke with the angle of the nib. For brushes, the amount of ink on the bristles and the force with which the brush is pressed also have an effect. There may be other factors so in general we define a function T which returns a value in $[0, \infty)$. We denote ϵ the output of this function. The T used for in Fig. 1 is a constant function.

To model each stroke (or part of a stroke), we define a Bézier curve $B(t)$ for $t \in [0, 1]$ over four control points p_0, p_1, p_2, $p_3 \in \mathbb{R}^2$. For our purposes, the control points are in $[0, 1]^2$. To clarify notation, $B(t) = \begin{bmatrix} B_u(t) & B_v(t) \end{bmatrix}^T$

To determine whether this point lies inside or outside the region, we must first express $B(t)$ implicitly. We define

$$B(t) = (1-t)^3 p_0 + 3(1-t)^2 t p_1 + 3(1-t)t^2 p_2 + t^3 p_3$$

where $p_i = (u_i, v_i)$, the coordinates of each control point.

Next, we determine if a given point $q = (u, v)$ lies inside of the stroke. That is, there exists some $\hat{q} = B(\hat{t})$ such that $||q - \hat{q}||_2 < T(\hat{t})$. One way to accomplish this is to find the nearest point to q on B, denoted $\hat{q} = (\hat{u}, \hat{v})$. Since B is a parametrized curve there exists a $\hat{t}$ such that $B(\hat{t}) = \hat{q}$. Then, q is contained in the stroke if and only if $||q - \hat{q}||_2 < \epsilon$ for some $\hat{t} \in [0, 1]$. Notice that both the $\hat{q}$ and $\hat{t}$ may not be unique. This is unimportant since every $\hat{q}$ is the same distance from q. We find this $\hat{q}$ by observing the vector $\hat{q} - q$ is orthogonal to $\mathrm{d}B/\,\mathrm{d}t$. Notice there may exist multiple $\hat{t} \in [0, 1]$ such that $B(\hat{t}) = \hat{q}$. Of these, we choose the $\hat{t}$ such that $||q - \hat{q}||_2$ is minimized. So we solve for t in

$$\begin{aligned} 0 &= \hat{q} \cdot \frac{\mathrm{d}B}{\mathrm{d}t} \\ &= B_u(t)\frac{\mathrm{d}B_u}{\mathrm{d}t}(t) + B_v(t)\frac{\mathrm{d}B_v}{\mathrm{d}t}(t) \end{aligned} \tag{1}$$

which amounts to finding the real roots of a fifth-order polynomial. For the exact formula of this polynomial see the attached Python code.

2 The `Handwriting` Texture

We define a texture as a function which takes as input a (u, v) coordinate and returns either another texture (which can in turn be evaluated at (u, v)) or an RGB color value. In the case of handwriting, we return one texture if (u, v) is inside the curve and a second texture otherwise. The T function described in Section 1 is known as the brush in the `Handwriting` class. This implementation defines T as a function only of the Bézier curve's parameter t.

A given `Handwriting` instance has a list of characters defined by zero or more Bézier curves. Each character has all of its Bézier curves (also called strokes) contained in $[0, 1]^2$. For each (u, v) texture coordinate, we determine if $\hat{q}$ is contained in any of the strokes in the nearby characters.

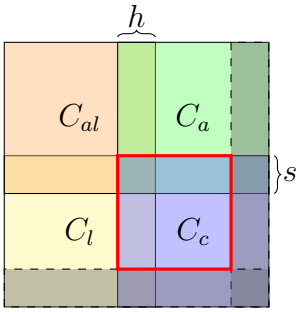

Figure 2: A point inside of the red box may be inside any of C_{al}, C_c, C_a, or C_l.

To make the writing look more natural, we wish to have characters overlap. This may be specified with h and s. If $h, s \in [0, 0.5)$ then there will never be more than four characters present

in any given point. Consider Fig. 2. Given that each colored box is $[0,1]^2$ and we overlay each character so that the overlap horizontally is h and the overlap vertically is s, for each character C we must also check the characters to the left, above, and diagonally above and to the left of C. We do not need to check the other surrounding characters because these will be checked if (u,v) is in the first $[1-h]\times[1-s]$ box of the next character. For the center character C_c this box is outlined with thick, red lines in Fig. 2.

Then, given some coordinate (u,v) we transform the coordinate so we can determine which character box the point is contained in as well as where inside the box. This is achieved with the functions

$$g_u(u) = u + h\left\lfloor \frac{u}{1-h} \right\rfloor \tag{2}$$

$$g_v(v) = u + s\left\lfloor \frac{v}{1-s} \right\rfloor. \tag{3}$$

$$f_u(u) = u + h\left\lfloor \frac{u-h}{1-h} \right\rfloor \tag{4}$$

$$f_v(v) = v + s\left\lfloor \frac{v-s}{1-s} \right\rfloor \tag{5}$$

Equations (2) and (3) transform from a (u,v) coordinate to the position in the overlapped characters, where the ingetral parts of $(g_u(u), g_v(v))$ determine the character and the decimal parts determine the coordinate in that character. However, this overlapping lattice may have multiple characters for a given point. Thus for each point u, v we must check $(g_u(u), g_v(v))$, $(g_u(u), f_v(v))$, $(f_u(u), g_v(v))$, and $(f_u(u), f_v(v))$. These determine the coordinates inside of C_c, C_a, C_l, and C_{al} from Fig. 2, respectively. The plots of (2) and (4) are shown in Fig. 3. Notice this is only along one axis and there is overlap of these functions on the domain $n + [h, 1-h]$ for $n \in \{0, 1, 2, \dots\}$.

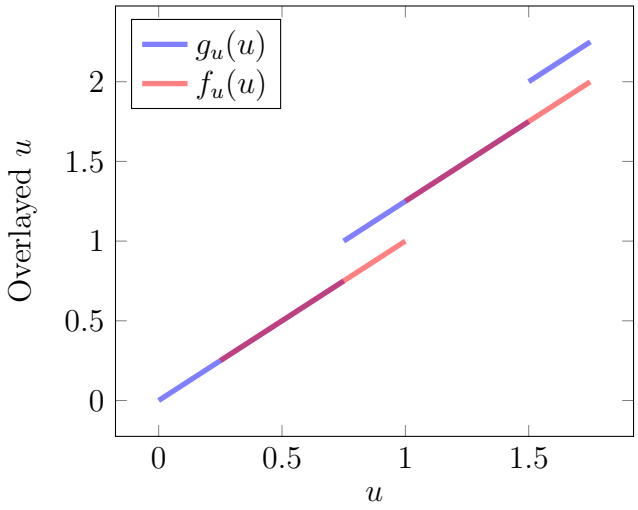

Figure 3: Plots of equations (2) and (4) with $h = 0.25$.

3 Lessons Learned

This approach is rather complicated and limited. Using Equations (2)–(5), characters must all have the same spacing between them. A more general approach would be to use traditional computer graphics methods and with each character associate a 3×3 transformation matrix. In general, this makes determining which characters to check at every point more difficult but a reasonable assumption with handwriting is the characters are roughly evenly-spaced. Thus, one could split up the image into a rectangular grid and each time a transformation is set, keep track of which boxes that character fits inside.

4 Future Work

The first approach to add generality to the `Handwriting` class would be to add the transformations described previously.

This method relies on finding solutions to the polynomial (1). This gives arbitrary precision when sampling, but requires solving a fifth order polynomial at each pixel. This takes about 600 seconds to render a 512×512 image with an average of 4 Bézier curves per character. However, since we have the implicit equation for B we may sample B at arbitrary points to "draw" the curve much more efficiently; it is inexpensive to compute $B(t)$ and a normal to B. These provide enough information along with the value of T to fill in the area around a given curve. Furthermore, this gives the benefit of providing the drawing of the curve as it would actually be drawn by hand. One could specify some increasing sequence of points $\{t_i\}_{i=1}^{N}$ where $t_i \in [0, 1]$, and the line between the two edges could be drawn for each $B(t_i)$.