Curve Fitting Algorithm Using Iterative Error Minimization for Sketch Beautification

Junyeong Yang and Hyeran Byun
Dept. of Computer Science, Yonsei University, Seoul, Korea, 120-749
{gundid,hrbyun}@cs.yonsei.ac.kr

Abstract

In previous sketch recognition systems, curve has been fitted by a bit heuristic method. In this paper, we solved the problem by finding the optimal parameter of quadratic Bezier curve and utilize the error minimization between an input curve and a fitting curve by using iterative error minimization. First, we interpolated the input curve to compute the distance because the input curve consists of a set of sparse points. Then, we define the objective function. To find the optimal parameter, we assume that the initial parameter is known. Then, we derive the gradient vector with respect to the current parameter, and the parameter is updated by the gradient vector. This two steps are repeated until the error is not reduced. From the experiment, the average approximation error of the proposed algorithm was 0.946433 about 1400 synthesized curves, and this result demonstrates that the given curve can be fitted very closely by using the proposed fitting algorithm.

1. Introduction

As the same general pattern recognition manner, to beautify a sketch or to recognize sketch, we first should extract the good feature to analyze a stroke. Then, points between each feature point pair are expressed by line or curve. Sezgin et al.[5] recognized the sketch by using curvature and speed feature of a stroke. The curve is fitted by cubic Bezier curve. In this system, two unknown control points is computed by using heuristic estimation. Yui[6] proposed a sketch recognition system by using the Mean-shift algorithm[3]. This system utilizes the mean shift algorithm for clustering on the direction feature space of the input stroke. In [6], to beautify sketch, it compare the input stroke with primitive shapes: line, arc, and circle. Then, it beautifies the sketch as one of primitive shapes. Calhoun et al.[2] proposes the multi-stroke symbol recognition system. This system only treats the line segment. Hence, to represent a curve, many control points are needed. Igarashi et al.[4] proposed the interactive beautification system. This system give a drawing feedback to the user. From the feedback, the user can easily draw a shape. However, the system has the drawing constraints. Arvo and Novins[1] proposed the online sketch recognition system. This system can recognize line, arc, and circle. During drawing, each primary shape is classified into three shapes by using the least square solution. This system should repeatedly compute the similarity between the drawing stroke and primary shapes at every drawing.

In this paper, we propose the fitting algorithm using iterative error minimization. We first derive the gradient vector with respect to the current parameter. Then, the parameter is updated from the gradient vector until the error is not reduced. In section 2, we describe the curve fitting algorithm in detail. In section 3, we demonstrate the experimental results. Finally, we present a conclusion in section 4.

2. Curve Fitting Algorithm using iterative error minimization

Bezier curve is the important tool to model smooth curves and is a parametric model. It is represented by the weighted linear combination of control points which controls the curvature of a curve. It is divided as three kinds of linear curve, quadratic curve, and cubic curve by the number of the control point. Quadratic Bezier curve is given as :

\[ \mathbf{B}(t) = w_0(t)\mathbf{P}_0 + w_1(t)\mathbf{P}_1 + w_2(t)\mathbf{P}_2 \]  

where \( w_0(t) = (1 - t)^2 \), \( w_1(t) = 2t(1 - t) \), and \( w_2(t) = t^2 \). A generated curve is the path trace by the equation (1), given points \( \mathbf{P}_0 \), \( \mathbf{P}_1 \), and \( \mathbf{P}_2 \). This equation is suitable with a representation of a parabolic curve.
A input curve or stroke $X$ is defined as a set of $k$ input points drawn on two dimensional plane with a input device such as a mouse and a pen where $X = \{x_1, \ldots, x_k\}$, $x_i \in \mathbb{R}^2$. When we define $P_0$ as $x_1$ and $P_2$ as $x_k$, we should solve $P_1$ and $t$. For displaying $X$, we link $x_i$ and $x_{i+1}$ as a line.

In this section, we propose the curve fitting algorithm. It is based on an iterative error minimization. We first propose a gradient vector, $\Delta P$, with respect to the current parameter, $P_1$. $P_1$ is updated by $P_1 + \Delta P$.

Hence, this method little depends on the initial $P_1$. Therefore, we propose the good initial estimate of $P_1$.

2.1. Derivation of Gradient Vector

We first define the error between an input curve and a fitting curve. Assume that $P_0$ is a start point, $x_1$, and $P_2$ is a end point, $x_2$. Now, we should solve two parameters of quadratic Bezier curve : $t$ and $P_1$. Here, the equation (1) is redefined as :

$$B(P_1, t_i) = w_0(t_i)\hat{P}_0 + w_1(t_i)P_1 + w_2(t_i)\hat{P}_2 \quad (2)$$

The equation (2) is the function about $P_1$ and $t$. We can fit a synthesized curve to an input curve by adjusting $t$ and $P_1$ of quadratic Bezier curve. When $P_1$ is fixed, we can compute all points of quadratic Bezier curve by adjusting $t_i$. However, input points are a bit sparse. About each input point pair, we generate points by using a line equation which passes through two points. We define an interpolated curve as a set of input points and newly generated points. First, about predefined $\{t_1, \ldots, t_k\}$ and the initial $P_1$, we compute points on a fitting curve by the equation (2). We compute a normal direction of a line linked between $\hat{P}_0$ and $\hat{P}_2$, and compute an intersection point between an interpolated curve and a line which is elongated along normal direction from a fitting point. Here, about each $t_i$, we define an intersection point as $I_i$. Figure 1 shows the conceptual diagram of the proposed distance between an input stroke and a fitting curve. The proposed objective function is given as :

$$E(P_1) = \sum_{i=1}^{k} \left[ B(P_1, t_i) - I_i \right]^2 \quad (3)$$

When we assume that a initial $P_1$ is known, we can solve this problem by iterative parameter updating. The redefined objective function is expressed as :

$$E(P_1 + \Delta P) = \sum_{i=1}^{k} \left[ B(P_1 + \Delta P, t_i) - I_i \right]^2 \quad (4)$$

where $\Delta P = (\Delta x, \Delta y)^T \in \mathbb{R}^2$. We differentiate the equation (4) with respect to $\Delta P$, and then we set it as zero.

$$\sum_{i=1}^{n} w_i(t_i) \left[ B(P_1 + \Delta P, t_i) - I_i \right] = 0 \quad (5)$$

If we compute an expansion of above equation and solve it with respect to $\Delta P$, we can obtain a following result :

$$\Delta P = \sum_{i=1}^{n} U(t_i) / \sum_{i=1}^{n} w_i(t_i)^2 \quad (6)$$

where $U(t_i) = \hat{P}_2 / \hat{P}_0 - w_0(t_i)w_1(t_i)\hat{P}_0 - w_1(t_i)^2P_1 - w_1(t_i)w_2(t_i)\hat{P}_2$. $\Delta P$ is a gradient vector which minimize the equation (4) with an initial $P_1$ and an initial $P_1$ is updated by $P_1 \leftarrow P_1 + \Delta P$. This procedure is repeated until $E(\cdot)$ is not reduced. Figure 2 shows the result of curve fitting algorithm without estimating the good initial $P_1$. As iteration increases, our algorithm gradually fits the synthesized curve to the input curve. However, this algorithm depends on the initial $P_1$ because the bad initial $P_1$ increases the number of iterations. To overcome this problem, we propose the method to estimate a good initial $P_1$ in next section.

2.2. The Initial $P_1$ Estimation

If we can estimate the proper initial $P_1$, we can reduce the number of fitting iterations. To overcome this problem, we use the maximum of the normal direction distance from the line which passes through $x_1$ and $x_k$.

The projection basis is represented as :

$$\phi = \frac{\hat{P}_2 - \hat{P}_0}{\left\| \hat{P}_2 - \hat{P}_0 \right\|} \quad (7)$$

Here, each point $x_i$ is projected onto $\phi$ by a following equation :

$$x_i^{(p)} = \hat{P}_0 + c_i \phi \quad (8)$$
where $c_i = \phi^T(x_i - \hat{P}_0)$. Now, the maximum distance is given as:

$$i_{max} = \max_i \left[ \|x_i - x_i^{(p)}\| \right]$$

(9)

$i_{max}$ corresponds to an index of an input point having the maximum distance. Define $\hat{t}$ as $t$ having the maximum of the normal direction distance on the ideal Bezier curve. We estimate $\hat{t}$ as:

$$\hat{t} = \begin{cases} 
0 & \text{if } 0 \leq c_{i_{max}} < s \\
0.5 & \text{if } s \leq c_{i_{max}} < 2s \\
0.55 & \text{otherwise}
\end{cases}$$

(10)

where $s = \|P_2 - P_0\|/3$. $\hat{t}$ estimate is based on the property of quadratic Bezier curve. The function value of quadratic Bezier curve is densely put on left or right side according to $t$. Each value of the equation (10) is obtained from experiment and the location of high peak of Quadratic Bezier curve is put on left or right side according to $\hat{t}$. Note that $\hat{t}$ is not a accurate value. Although $\hat{t}$ estimate is always not a best initial estimate, this method is very helpful to reduce the iteration of the fitting phase.

Now, $\hat{t}$ can be obtained from the equation (10), and we can estimate $P_1$ because we know $P_0$, $P_2$, $\hat{t}$, and $x_{i_{max}}$.

$$w_0(\hat{t})\hat{P}_0 + w_1(\hat{t})P_1 + w_2(\hat{t})\hat{P}_2 = x_{i_{max}}$$

(11)

we solve the equation (11) about $P_1$ and we can obtain the following solution:

$$P_1 = (x_{i_{max}} - w_0(\hat{t})\hat{P}_0 - w_2(\hat{t})\hat{P}_2)/w_1(\hat{t})$$

(12)

Figure 3 shows examples of the result of the proposed initial $P_1$ estimate by using the equation (12). As you see Figure 3, our initial algorithm is very good. From the initial $P_1$ estimate, five iterations was enough to obtain a good fitting result in our experiment.

We first compute the initial $P_1$ by the equation (10). Then, we compute the gradient vector by the equation (6), and $P_1$ is updated by using $P + \Delta P$ in the number of predefined iterations until the error is not reduced. This algorithm is summarized in Table 1.

3. Experiment

We generated the curve by using the equation (1) for the experiment. We first defined $P_0$ as $(50, 200)^T$ and $P_2$ as $(300, 200)^T$. We defined $y$-coordinate of the $P_1$ as 100, and varied $x$-coordinate of the $P_1$ from 1 to 350. We changed $t_i$ from 0 to 1 with 0.05 and generated 20 points. Then, we inserted noise into generated curves. We inserted the noise about each subset with four different ranges: $0 \sim 1$ pixel, $0 \sim 2$ pixel, $0 \sim 3$ pixel, and $0 \sim 4$ pixel. We did this work about each synthesized curve 20 times. Therefore, our experimental database consists of four subsets with different noise level. Finally, each subset consists of 350 synthesized curves.
Table 1. The proposed fitting algorithm using iterative error minimization.

A curve $X = [x_1, \ldots, x_k], x_i \in \mathbb{R}^2$ is given.

**Initialization:**
1. $\hat{P}_0 \leftarrow x_1$.
2. $\hat{P}_2 \leftarrow x_k$.
3. Compute $t_{\text{max}}$ by using the equation (9).
4. Compute $\ell$ estimate by using the equation (10).
5. Compute the initial $P_1$ by using the equation (12).

**Iteration:**
6. Compute the old error, $E_o$, by using the equation (3).
7. Compute the gradient vector by using the equation (6).
8. $P_1 \leftarrow P_1 + \Delta P_1$.
9. Compute the new error, $E_n$, by using the equation (3).
10. If $E_n < E_o$, go to step 6.
11. Otherwise, terminate.

About each subset, we averaged the Euclidean distance between two curves which are a generated curve and a curve obtained from the proposed fitting algorithm.

Figure 4 shows examples of the fitting result and Figure 5 shows the experimental result of the proposed fitting algorithm. From the experiment, the average approximation error was 0.946433 about 1400 synthesized curves.

![Figure 4. Example of the fitting result. A line is the synthesized curve and a dot line is the fitting result.](image)

4. Conclusion

In this paper, we proposed the simple and powerful curve fitting algorithms by using iterative error minimization. To define the error between the input curve and the synthesized curve, we interpolated the input curve. Then, we derived the gradient vector, $\Delta P$, to minimize the sum of perpendicular distance between two curves. Then, we computed the optimal parameter from updating the initial $P_1$ by a gradient vector, $\Delta P$. Our experimental result showed that the fitting performance of the curve fitting algorithm is very good.

**Acknowledgements**

This work was supported by grant No. R01-2007-000-11683-0 from the Basic Research Program of the Korea Science & Engineering Foundation

**References**


