1X-07

Curve Fitting with G1 Continuous Cubic Bézier Curves

Zhou Chiyu† Suguru Saito‡

†, ‡Tokyo Institute of Technology

E-mail :†chiyu@img.cs.titech.ac.jp, ‡suguru@c.titech.ac.jp

One merit of parametric curves is that users can modify the shape of curves by changing parameters. In our previous paper [1], we proposed a procedure to vectorize the contours of lines. However, it does not support a long line.

This paper proposes a new method that can fit a long curve on a raster image using several cubic Bézier curves, and the G1 continuity is maintained between adjacent curves.

1 Method

1.1 Preprocessing

Our line drawing vectorization method [1] is divided into three steps: critical points (junctions and intersections) detection, topology extraction and curve fitting for each path in the extracted topology. This paper introduces the fitting method for the third step.

The fitting method is divided into two steps. The separation of one long smooth curve into several Bézier Curve segments according to curvature, and the fitting of each segment with one cubic Bézier curve while maintaining G1 continuity.

1.2 Curve separating

Our method is inspired by [2] that separates a curve at points that have local maximum curvature. As shown in Fig.1, the curvature at the point can be approximately calculated by [3], which uses a chord-curve area surrounded by a fixed-length chord (blue lines) L and the curve. Then it groups the points whose curvature is larger than a threshold into several clusters (red parts) A_i and pick the extrema points as breakpoints (green points) T_i that have the largest curvature in each cluster. Consequently, a curve is separated into several segments according to T_i.

1.3 Fitting

When measuring the distance between a fitted curve and original data points, a common method is sampling on the fitted curve and calculating the sum of the shortest distances between each data point and one point in sampling points. However, it does not perform well when the sampling points are not enough. We adapted [4] that uses the height of the adjacent triangles as the distances.

To make the result smoother, G1 continuity that two segments share the same tangent direction at their adjacent point is constrained. In the case of the cubic Bézier curves that defined by four control points $\mathbf{P^{(i)}} = \left\{P_0^{(i)}, P_1^{(i)}, P_2^{(i)}, P_3^{(i)}\right\}$, as shown in Fig.2, it is described as Eq.1.

$$\begin{aligned} P_3^{(i)} &= P_0^{(i+1)} \\ P_3^{(i)} &= (1-\lambda)P_2^{(i)} + \lambda P_1^{(i+1)} \end{aligned} \quad (1)$$

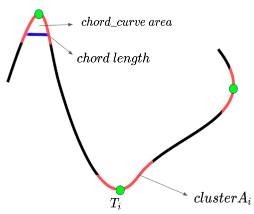

Fig.1: Curve separating

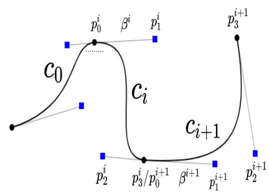

Fig.2: A G1 continuity cubic Bézier curve

In pre-experiments, we adapted many optimization methods such as harmony search and simulated annealing. However, in the case of cubic Bézier curves, we observed that if given an appropriate initial solution, the energy function could have very few local optima, using gradient descent would converge much faster.

The breakpoints are fixed as endpoints of Bézier curves, we used gradient descent to approximate the coordinates of the other two control points. To have a better global performance, we optimize two segments at the same time. For the first two segments, their initial solutions are midpoints of endpoints as the red points shown in Fig.3. After optimization, the first segment and the tangent direction of the first control points of the second segment are fixed. Then the second segment will be optimized with the third segment again. Algorithm 1 shows the details.

Algorithm 1 Fit(curve C)

Require: BreakpointSet $\{T_i\}$, SegmentSet $\{c_i\}$
Ensure: $P_1^{i+1} = P_0^{i+1} + \beta^{i+1} * norm(P_3^i - P_2^i)$
Output: SolutionSet $\{P_0^i, P_1^i, P_2^i, P_3^i\}$
  Initialize $\mathbf{P^0}$,$\mathbf{P}^1$
  GradientDescent($\mathbf{P}^0, \mathbf{P}^1$)
  $i \leftarrow 1$
  while $i \neq$ size(BreakpointSet) **do**
    Initialize β^i,$\mathbf{P}^{i+1}$
    $P_1^i = P_0^i + \beta^i * norm(P_3^{i-1} - P_2^{i-1})$
    $iter \leftarrow 0$
    for $iter < threshold_{iter}$ **do**
      EnergyLoss = GradientDescent(β^i,P_2^i,$\mathbf{P}^{i+1}$)
      $iter \leftarrow iter + 1$
    end for
    if EnergyLoss $> threshold_{energy}$ **then**
      $NewBreakpoint \leftarrow CurveMiddlePoint(T_i, T_{i+1})$
      Insert $NewBreakpoint$ to BreakpointSet
      Continue
    end if
    $i \leftarrow i + 1$
  end while

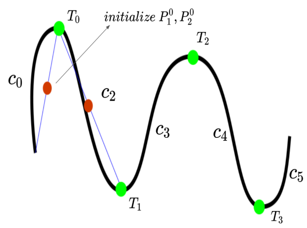

Fig.3: Curve fitting

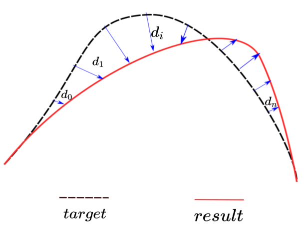

Fig.4: Average distance error

2 Experiments

To evaluate the performance of our method, we defined the average distance error E_{ave}, as

$$E_{ave} = \frac{\sum_{i=0}^{n} d_i}{arc\_length}, \tag{2}$$

where d_i is the distance from a sampling point of a target curve to the fitted curve along its normal direction. Fig.4 supposes the black dash curve is the target, and the red line is the fitted curve, and the blue arrows are d_i.

Fig.5 shows the results of our method. It can be seen that our method can fit long curve successfully with a very small error.

However, even with a small loss, the results do not look smooth enough sometimes, such as in the last small segment of Line3. This is due to our method does not perform well enough at points with large curvature.

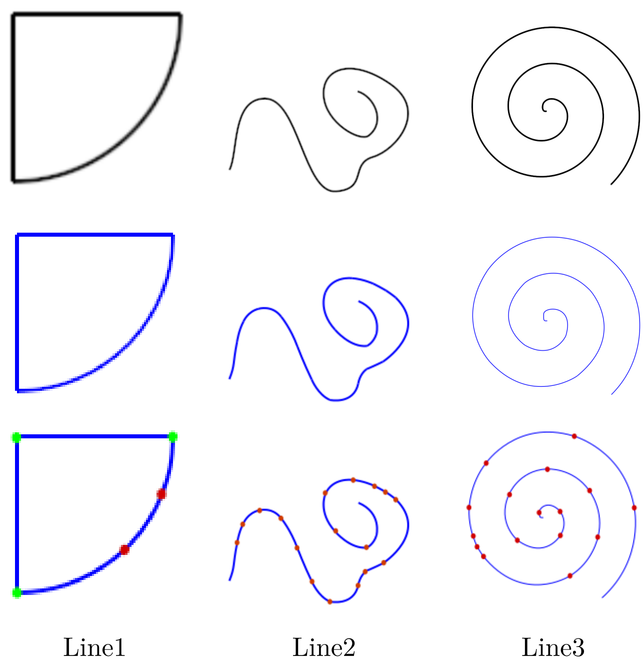

Fig.5: The first row is the targets, the second part is our outputs, and the third row shows the breakpoints(red points) and split points(green points) of our method.

Table.1: Evaluation results

Evaluation	Line1	Line2	Line3
E_{ave}	0.0659	0.1072	0.0927
$E_{max(d_i)}$	0.1816	0.3388	0.2747

3 Conclusion

In this paper, we proposed a vectorization method that fits a long curve with no sharp corners successfully using several G1 continuous cubic Bézier curves. We are going to refine our algorithm that set more breakpoints at points with large curvature.

References

[1] Z. Chiyu and S. Saito, "Vectorization of contours of lines in line drawing," IIEEJ Media Computing Conference, pp.#97 1–4, 2022.
[2] Z. Yan, S. Schiller, G. Wilensky, N. Carr, and S. Schaefer, "K-curves: Interpolation at local maximum curvature," ACM Transactions on Graphics (TOG), vol.36, no.4, pp.1–7, 2017.
[3] L. Shao and H. Zhou, "Curve fitting with bézier cubics," Graphical models and image processing, vol.58, no.3, pp.223–232, 1996.
[4] E.K. Ueda, A.K. Sato, T. Martins, R.Y. Takimoto, R.S.U. Rosso, and M. Tsuzuki, "Curve approximation by adaptive neighborhood simulated annealing and piecewise bézier curves," Soft Computing, vol.24, no.24, pp.18821–18839, 2020.