Binary image vectorization based on symmetric detection and bisection method

Anonymous

Abstract

Binary image boundary vectorization is the process of generating vector images represented by a sequence of linked parametric cubic Bézier curves from pixel image boundaries. The reconstructed curves should approximate the underlying structure of the boundaries as much as possible while using as few curves as possible. Existing methods do not perform well when considering both of these two main factors. In this paper, we mimic the process of human vectorizing image boundaries by first segmenting the boundary points into multiple segments with the corner points. For the boundary points in each segment, we adopt the bisection method to find the largest number of points, which a single curve can fit. More curves will be added if the fitting error is larger than a predefined threshold. The process is repeated until all the points in the segment are fitted, thus minimizing the number of Bézier curves. Besides, symmetric image boundaries can be detected and used to further decrease the number of curves required. Our method can also choose the optimal parameterization method case by case to further reduce the fitting error. We make a comparison with both new and classical methods and show that our method outperforms them.

Keywords: vectorization, symmetric detection, bisection, affine transformation

1 Introduction

Binary image boundary vectorization is referred to the process of converting images represented by two-dimension pixels to images represented by a sequence of parametric Bézier curves (vector images), which approximate the underlying structure of the pixel images. The latter representation is more compact as less storage is required and it can also be scaled up or down to any scale level without introducing aliasing or losing information compared to the former one [1]. Moreover, vectorization representation has many other applications, such as remoting sensing [2], feature recognition [3] and many others [4]. The pixel images can also be reconstructed from the vector images without losing significant information if needed. The goal of image boundary vectorization is to approximate the underlying structure of the image boundaries as much as possible while using as few numbers of Bézier curves as possible [5].

In general, the pipeline of vectorizing the images consists of the following steps. Preprocessing, boundaries (contours) detection for reconstructing the Bézier curves, and corner (salient) points detection, which are used to segment each boundary into multiple segments, each of which will be fitted by one or more Bézier curves independently. In the fitting phase, the boundary points in a segment are first parameterized using some points parameterization techniques and the obtained parameters are used for curve fitting. Meanwhile, break points may be added to further break the segment into multiple subsegments if one curve is not enough to fit the whole points in the segment.

Concerning the first three steps for boundary and corner point detection, many methods have been proposed, and we can select a suitable one from them for our task, which will be demonstrated in Section 4. For the last three steps, which include parameterization of segment points, fitting and breaking (if necessary), many techniques have also been presented. In terms of parameterizing the points, some classi-

cal methods include uniform parameterization, chord length method [6], centripetal [7, 8] and hybrid [9], etc. Each parameterization method has its pros and cons. No method can handle all problems of curve fitting. Therefore, it is recommended to adopt these methods based on specific problems and requirements. However, most proposed techniques for vectorizing images just choose one of the methods (normally chord length). Besides, break points may be necessary when one curve is not accurate enough to fit a segment. Thus, it is critical to effectively choose a minimal number of break points while tightly approximating the underlying structure of the segment points. For example, Schneider [10] treats the point with maximum error as the break point. On the contrary, the point with minimum error is chosen as the break point by Pavlidis [11]. The middle point of the segment is also used as the break point in [12]. But all those methods do not guarantee that each curve fits as many points as possible, thus may result in more curves needed to fit a segment.

To further reduce the number of curves required, we propose a method to fit a sequence of cubic Bézier curves to image boundaries using symmetric detection and bisection method. For the symmetric detection, because Bézier curves are invariant under affine transformation, which means for axis and point-symmetric picture boundaries, we just need to vectorize a part of the boundaries and the remaining parts can be vectorized by transforming the already reconstructed Bézier curves. Furthermore, The bisection method we adopt works similarly to the bisection algorithm in matching an element to the same element in an ordered list or finding the roots of a polynomial equation [13]. It is simple but very efficient, and can be adopted to determine the best break point quickly. By doing so, we can further minimize the number of Bézier curves to represent the input image boundaries.

2 Related work

Based on the types of images that are to be vectorized, image vectorization techniques can roughly be classified into several categories, which include natural images vectorization [14, 15], pixel art vectorization [16, 17], and artist-generated image vectorization [18, 19]. Some methods mainly focus on region boundary vectorization [20, 21, 22], which can also be categorized into natural images. Different types of input own different characteristics and the corresponding methods to vectorize them vary. For example, pixel art images are characterized by low resolution, which is not easy to vectorize, and thus requires specific and dedicated methods. In this paper, we focus on region boundary vectorization.

To vectorize image region boundaries, many methods have been proposed. Most of them adopt the pipeline we demonstrated in the introduction section. Specifically, Schneider [10] first locates the corner points by computing the angle between each point and its neighbours, and then for each segment between two adjacent corner points, a cubic Bézier curve is used to fit it. The break points are added if necessary to the place where there exists the maximum error between the fitting curve and the original points. The process is repeated until all the subsets in each segment are fitted successfully. On the contrary, Pavlidis adds the break point at the place with the minimum error [11]. The middle point of the segment is also used as the break point in [12]. But all these methods do not guarantee that the number of curves needed is as few as possible while keeping small fitting errors. To use as few as possible curves, people could try all possible arrangements of the corner and break points and choose the best one. But the time complexity is very high especially when the number of input points is large. To decrease time complexity, the dynamic programming technique [5] is adopted by Plass and Stone, and it can reduce the time complexity to $O(n^3)$ by dividing the problems into sub-problems recursively, but it's still relatively expensive. Chang and Yan [23] present a new optimization function and error metric to vectorize hand-drawn images, but they mention that obtaining the minimal number of Bézier curves is not their main aim. Pal et al. [24] propose an adaptive method to detect the break point, and then the initial approximated Bézier control points are obtained using the interpolation technique, from which the control points of the best fitting curves are found by ap-

plying two-dimensional logarithmic and an evolutionary search algorithm. Hoshyari et al. [25] adopt machine learning techniques to vectorize semi-structured boundaries, which are usually distinctly coloured and piecewise continuous. The supervised learning technique is used to detect the corner points, which are combined with global cues that both consider simplicity and continuity to output results that comply with human perception of semi-structured boundaries. Their framework is computationally expensive when the size of the images is large and other machine-learning techniques may give better results. Xie et al. [26] present an interactive tool to further refine the output from Image tracer [27] by mainly using the merging and splitting tool, which outperforms fully manual or automatic methods. As it is time-consuming and experience-dependent for manual methods, and for the automatic methods, most of them output too many small and noisy marks if the input image is complex, thus postprocessing is necessary. Besides, automatic methods may require more curves than necessary. He et al. [21] present a method to vectorize binary shape using affine scale-space, which consists of three steps: firstly, at the sub-pixel level, the curvature extrema of the boundary are computed, then the control points are detected as the curvature extrema in the affine scale space, and finally the points between adjacent control points are fitted with cubic Bézier curve using least square method under the condition that the fitting error is less than a predefined accuracy.

Concerning the optimization methods, many techniques have been proposed. For example, least square is a widely adopted method for minimizing an error function because of its effeteness, efficiency and simplicity. To further reduce the fitting error, a method named reparameterization[5, 10] is adopted to find a better distribution of parameters for each point iteratively. This method works in some cases but may fail in some other cases because the solution obtained using the Newton-Raphson method may only find the local optimum other than the global optimum solution [23], as the newly updated parameters may not satisfy $0 = t_0 \leq t_1 \leq \cdots \leq t_n = 1$. These optimization methods take the determined parameters of the points as input and optimize the error function to obtain the optimal positions of the control points of the fitting curve, which means the output results also depend on the parameter distributions of the points. There are many techniques to choose to parameterize the points, but it's difficult to decide which one works better for a certain case. So, some other methods treat the parameters of the points as free variables to be optimized rather than calculate them in one shot using a chosen parameterization method. It has also been shown that better results can be obtained by doing so [28, 29]. However, these methods that treat point parameters as free variables have some limitations. Firstly, it's not easy to tune the design variables such as the search range and the number of iterations, which are mostly empirical and problem-dependent. What's more, there is no guarantee that the process would converge.

Overall, all the aforementioned methods have both pros and cons, and they do not guarantee the number of Bézier curves is as few as possible while keeping good fitting quality. In this paper, we present a method to vectorize image boundaries to minimize the number of Bézier curves needed while maintaining high fitting accuracy by using both bisection and optimal parameterization and taking advantage of the affine transformation invariant property of Bézier curves. Experimental results demonstrate our method outperforms classical and new methods.

3 Overview

The general pipeline is introduced in the introduction section, but it can not guarantee the minimum number of curves needed. In this paper, we adapt the general pipeline to minimize the number of curves needed while keeping a good approximation. Firstly, after the boundaries are extracted, some techniques are used to determine whether they are symmetric about an axis or a point. If yes, the symmetric axis or point is stored and only a basic part of the boundaries is needed to be vectorized. What's more, to determine the best break point efficiently, the bisection method is used, which can reduce the time complexity from $O(n)$ to $O(\log n)$. Its principle is similar to the bisection algorithm in matching an element in an ordered list. In the parame-

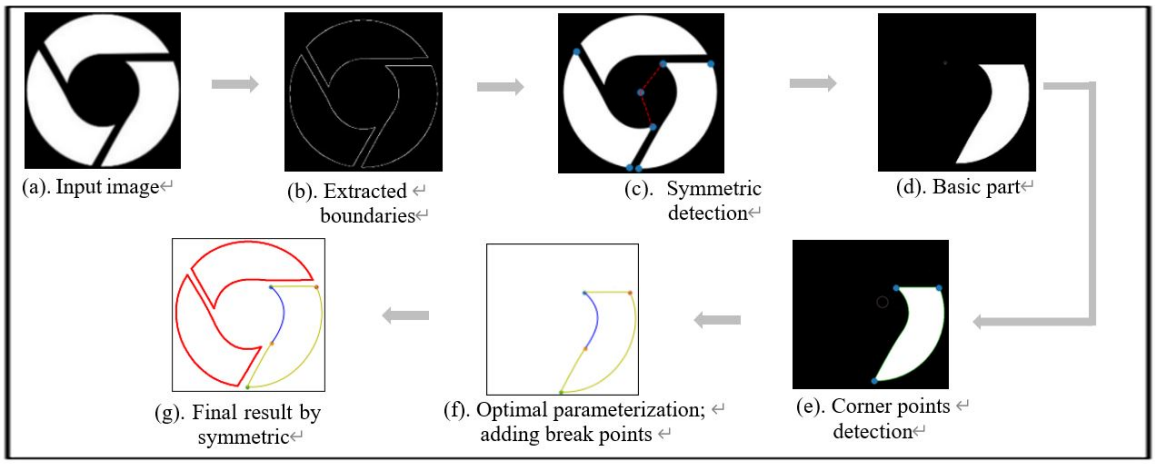


Figure 1: Pipeline of our proposed method

terization phase, we choose the optimal parameterization method to further reduce the fitting error by trying all kinds of existing methods in a brute-force way, and the speed is fast since we adopt the least square method to optimize the fitting function error, which is a very general, efficient and effective way of minimizing a function. The adapted pipeline of our method is shown in Figure 1.

4 Piecewise Bézier curves fitting

In order to achieve efficient fitting of piecewise Bézier curves, we first check in Subsection 4.1 whether an image shape is symmetric about an axis or a point. If yes, only one symmetric region of the image will be vectorized and the vectorization of other regions can be quickly obtained through symmetry. If the image is not symmetric, the image has to be vectorized fully. After the symmetry of an image has been checked, we investigate the fitting of piecewise Bézier curves in Subsection 4.2 and 4.3.

4.1 Symmetric axis and point detection

In order to determine whether the boundary of a given image is symmetric about an axis or a point, we propose the following idea. First, we determine whether the boundary is symmetric about an axis. If not, we further test whether it is symmetric about a point. If the boundary of the image is not symmetric about both an axis and a point, the image is treated as a regular one.

To determine whether the image boundary is symmetric about an axis or not, the boundaries are first detected using the findContours function in the OpenCV library [30]. Next, the principal component analysis (PCA) technique is adopted to find the two main axes of the boundaries. Since only the directions of the two axes are determined and they are free in their perpendicular direction, and we need to find one point they should go through to fix both their position and direction. We notice that a symmetric axis should go through the centric point of the boundary points, which can be calculated by adding all the coordinates of the boundary points and dividing the sum by the number of boundary points. After obtaining the two princi-

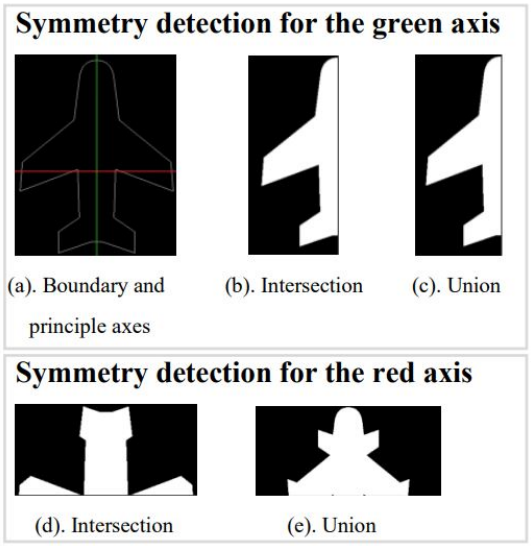


Figure 2: Symmetric axes detection

pal axes that go through the centric point of the image shape, as shown in Figure 2(a), we then test whether the shape is symmetric about any of these two axes. To do so, for each axis, we flip one side of the shape around the axis and use it to compute the intersection and union with another side of the shape. If the ratio between the intersection and union is close to one, it means that the shape is symmetric about this axis, as demonstrated in Figure 2(b) and 2(c), the intersection and union shape are almost the same. While for another axis, the ratio between the corresponding intersection and union is much smaller than one, which means the shape is not symmetric about this axis, as shown in Figure 2(d) and 2(e). By setting a threshold, we can decide whether a shape is symmetric about an axis or not.

Given the image shape is symmetric about an axis whose equation can be expressed as $ay + bx + c = 0$, in which a, b, c are known constants. In such a case, only half of the image boundaries that lie on one side of the symmetric axis need to be vectorized. Supposing half of the image boundaries have been fitted with one cubic Bézier curve defined by four control points p_1, p_2, p_3, p_4 whose coordinates are $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ respectively in the same coordinate system as the symmetric axis. Then the boundary points lying on the other side of the symmetric axis can be vectorized by another cubic Bézier curve, whose control can be obtained by mirroring p_1, p_2, p_3 and p_4 around the symmetric axis using the following equation: given a point coordinate (p, q) and a symmetric axis equation $ay + bx + c = 0$, the symmetric point (p_s, q_s) of the given point about the symmetric axis is shown in equation 1.

$$
\begin{aligned}
p_s &= \frac{p * (a^2 - b^2) - 2 * b * (a * q + c)}{a^2 + b^2} \\
q_s &= \frac{q * (b^2 - a^2) - 2 * a * (b * q + c)}{a^2 + b^2}
\end{aligned} \tag{1}
$$

As demonstrated in Figure 3, for a symmetric shape, only half of the shape needs to be vectorized. So, for an axis-symmetric image shape, we just need to keep half of the control points and three variables which represent the symmetric line. By doing so, the number of curves can be further reduced.

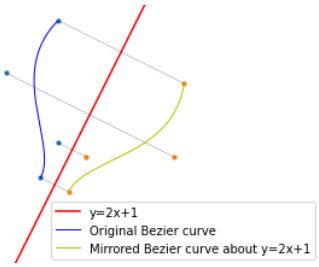


Figure 3: A cubic Bézier curve and its mirrored counterpart about an axis

To determine whether an image shape is symmetric around a point, if so, it can be observed that the centric point of the image boundaries should be the symmetric point. In such cases, we then use the Connected Component Labelling technique [31] to segment the images into multiple regions (number= r). Because the corner points ((number= n)) detected are ordered based on the quality level, there are $t = \frac{n}{r}$ corner points correspondences for each region with other regions, as shown in Figure 4, the corner points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) in Region 1 correspond to the corner points $(x_1', y_1'), (x_2', y_2')$, and (x_3', y_3') in Region 2, respectively. With these corresponding points, we can estimate the potential rotational angle (θ) of one region with respect to its adjacent region around the symmetric point using the following equation:

$$
\begin{bmatrix}
x_1 & -y_1 \\
y_1 & x_1 \\
x_2 & -y_2 \\
y_2 & x_2 \\
\vdots & \vdots \\
x_t & -y_t \\
y_t & x_t
\end{bmatrix}
\begin{bmatrix}
\cos(\theta) \\
\sin(\theta)
\end{bmatrix}
=
\begin{bmatrix}
x_1' \\
y_1' \\
x_2' \\
y_2' \\
\vdots \\
x_t' \\
y_t'
\end{bmatrix} \tag{2}
$$

where $(x_1, y_1), (x_1', y_1'), \ldots, (x_t, y_t), (x_t', y_t')$ are corresponding corner points for the two regions. This is normally an overdetermined linear system of equations. To solve find the best solution, we use the pseudoinverse method. After obtaining the potential rotating angle θ, we can rotate one region with about the centric point by θ and calculate the ratio between the intersection and union. By doing so, we can also reduce the number of curves needed. Lastly, if an image shape is neither symmetric

about an axis nor about a point, then it is treated as a normal image, which has to be vectorized fully.

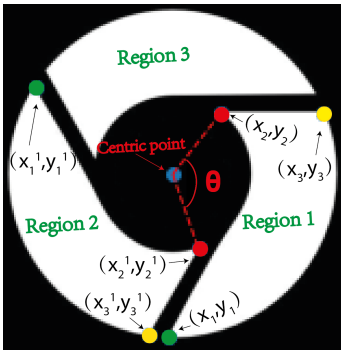


Figure 4: Symmetric point detection

4.2 Fitting

A cubic Bézier curve is defined by the four control points in the following expression:

$$C(t, \mathbf{p}) = \sum_{k=0}^{3} p_k B_k(t) \tag{3}$$

where $p_k \in \mathbf{p}$ (k = 0,1,2,3) are the four control points, $B_k(t)$ are called the Bernstein polynomials. Specifically, $B_0(t) = (1-t)^3$, $B_1(t) = 3*t*(1-t)^2$, $B_2(t) = 3*(1-t)*t^2$ and $B_3(t) = t^3$. And t is the parameter variable associated with each point of the segments, $t \in [0, 1]$.

Given N points $(P_1, P_2, \ldots, P_n)$ of a segment, the parameter t_i associated with each point can be obtained using one of the mentioned points parameterization methods, then a cubic Bézier curve, defined by equation 3, will be used to fit to those points. So, the following error function should be minimized:

$$min \sum_{i=1}^{N} (C(t_i, \mathbf{p}) - P_i)^2 \tag{4}$$

As a cubic Bézier curve would go through its two end control points, it means that the two end control points p_0 and p_3 are fixed at the first and last point of the segment. So, $p_0 = P_1, p_3 = P_N$ only p_1 and p_2 are to be optimized. To minimize the error function, we take its derivative with respect to p_1 and p_2, and set the derived equations to zeros, which can be used to derive p_1 and p_2. Specifically, let

$$f(p_1, p_2) = \sum_{i=1}^{N} (C(t_i, \mathbf{p}) - P_i)^2 = \sum_{i=1}^{N} [(C(t_i, \mathbf{p}) - P_i)]^T [C(t_i, \mathbf{p}) - P_i] \tag{5}$$

We also define $a_i = 3*t_i*(1-t_i)^2$, $b_i = 3*(1-t_i)*t_i^2$ and $c_i = (1-t_i)^3*p_0 + t_i^3*p_3 - P_i$, which are known once given the points of a segment, then equation 5 becomes:

$$f(p_1, p_2) = \sum_{i=1}^{N} [(a_i * p_1 + b_i * p_2 + c_i)]^T [(a_i * p_1 + b_i * p_2 + c_i)] \tag{6}$$

taking the derivate of $f(p_1, p_2)$ with respect to p_1, p_2, we obtain

$$\frac{\partial f}{\partial p_1} = \sum_{i=2}^{N-1} [2 * a_i^2 * p_1^T + 2 * a_i * b_i * p_2^T + 2 * a_i * c_i] \tag{7}$$

$$\frac{\partial f}{\partial p_2} = \sum_{i=2}^{N-1} [2 * a_i^2 * p_1^T + 2 * a_i * b_i * p_2^T + 2 * a_i * c_i] \tag{8}$$

We then define $a_1 = \sum_{i=2}^{N-1} 2 * a_i^2$, $b_1 = \sum_{i=2}^{N-1} 2 * a_i * b_i$, $b_1 = \sum_{i=2}^{N-1} 2 * a_i * c_i$, $b_1 = \sum_{i=2}^{N-1} 2 * b_i^2$, $b_1 = \sum_{i=2}^{N-1} 2 * b_i * c_i$ which are all known given points of a segment. Then equation (7) and (8) becomes:

$$a_1 * p_1^T + b_1 * p_2^T = -c_1 \tag{9}$$

$$b_1 * p_1^T + a_1 * p_2^T = -c_2 \tag{10}$$

Let matrix $A = \begin{bmatrix} a_1 & b_1 \\ b_1 & a_2 \end{bmatrix}$, $b = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix}$. Then we can obtain our solution $\begin{bmatrix} p_1^T \\ p_2^T \end{bmatrix} = A^{-1} * b$. After determining the four control points (p_0, p_1, p_2, p_3) of a cubic Bézier curve that is fitted to a subset of points in a segment. We measure the fitting quality by treating the largest distance between the fitting curve and the input points, which can be obtained using the following expression:

$$Err = max\{|C(t_i, \mathbf{p}) - P_i|\} \quad i = 2, 3, \ldots, N-1 \tag{11}$$

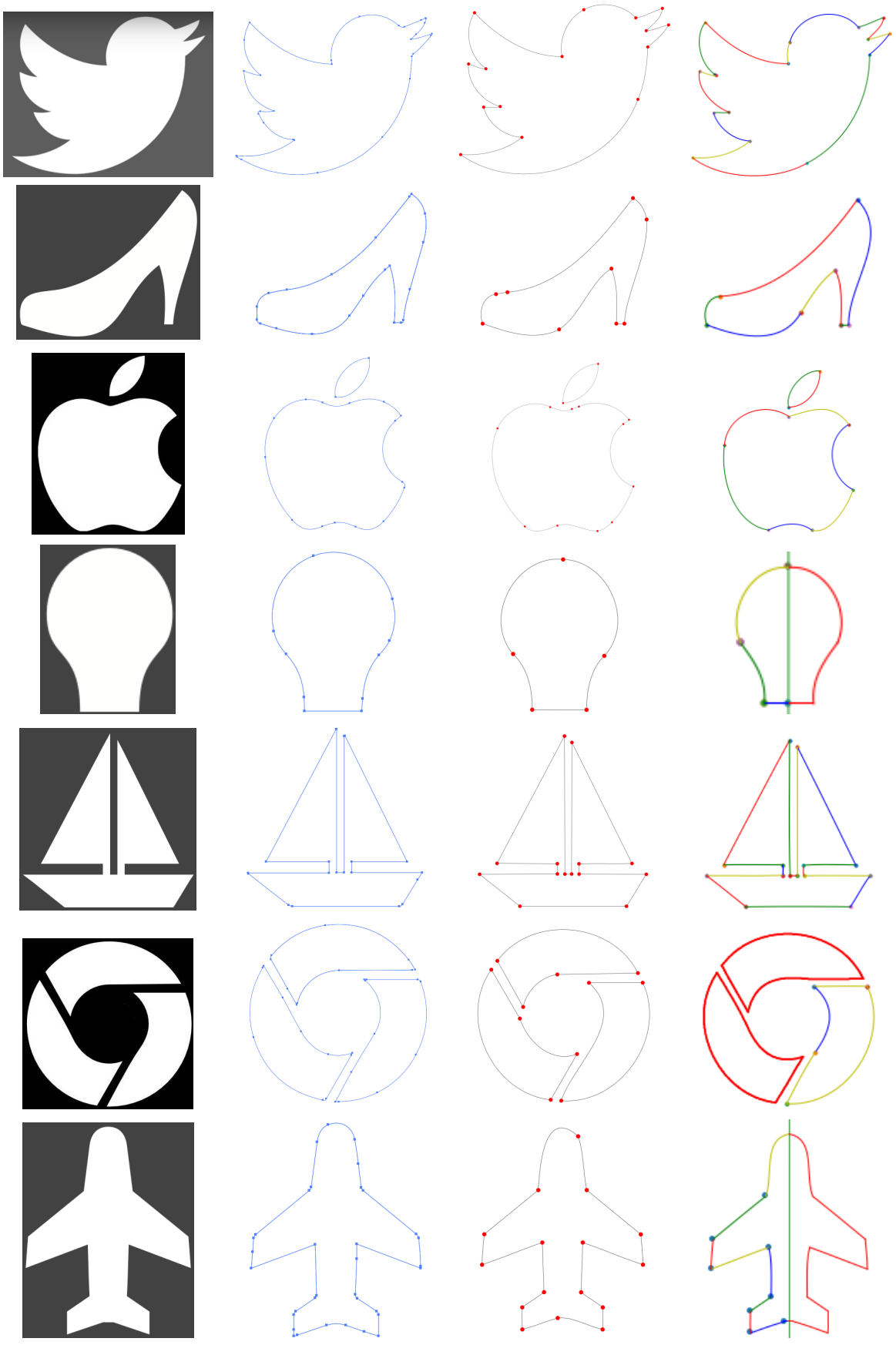
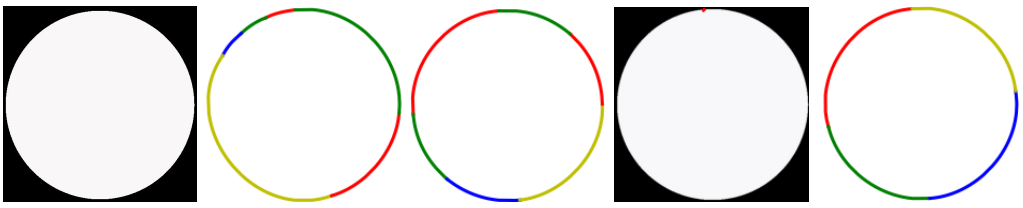


Figure 5: Results using different techniques, (first column) input image; (second column) Image tracer[27]; (third column) Affine scale-space[21]; (last column) our method.



(a) (b) 6 curves (c) 6 curves (d) error (e) 4 curves

Figure 6: (a) Input images; (b) adding break points at positions with max error[10]; (c) positions at middle point[12]; (d) at positions with minimum error[11]; (e) our bisection method

4.3 Bisection method

As we mentioned in the introduction section, the bisection method is applied when one cubic Bézier curve can't fit all the points of a segment well. The bisection method works similarly to the bisection algorithm in matching an element to an ordered list.

To demonstrate that the bisection method outperforms traditional methods in determining the suitable positions of break points, we compare it with three other methods, which choose the break point at the middle point, the point with the largest error, and the smallest error respectively. We set the threshold the same for all the cases, as we can see from Figure 6(b) and 6(c), more curves will be used if we add the break points at the positions with the maximum error or at the middle point position. Besides, putting break point at the position with the minimum error may cause a failure because the chosen position is just one point away from its near endpoint, resulting in a segment with just three points as shown in Figure 6(d), with which a Bézier curve can not be determined. However, using the bisection method, the number of curves is further minimized as shown in Figure 6(e).

5 Results and comparison

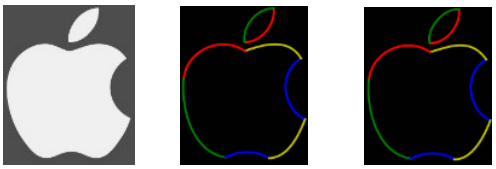
In this section, we present the results of vectorizing image boundaries using our method and compare it with the new and classical methods. We set the error tolerance the same for our method and the relatively new method to make the comparison fair. For the classic method, setting the error tolerance is not available so we keep the default setting. As we can see from Figure 5, our proposed method outperforms existing methods in terms of using a minimal number of cubic Bézier curves while using a similar good approximation. Table 1 shows the number of curves needed for each input image using other methods and our methods. Lastly, we also investigate the impact of different parameterization methods, as shown in Figure 7(b) and 7(c), each coloured curve represents a cubic Bézier curve fitted to a segment, and there are eight of them. Even though two vectorized images using two approaches look similar, their fitting error is different, which is demonstrated in Table 2. As we can see, optimal parameterization should be chosen for each segment to reduce the fitting error further rather than just adopting one method for all cases as most papers did.

Table 1: Number of curves required for our method and other methods

Input	Image tracer	Affine scale-space	Our method
bird	> 20	15	15
heel	> 20	9	**7**
apple	17	13	**8**
bulb	12	5	**3**
ship	17	14	14
Circle like	> 10	10	**5**
airplane	> 18	not good	**10**

6 Conclusions and discussion

In this paper, we propose a method which incorporates symmetric image shape detection and the bisection technique to vectorize image boundaries, aiming to minimize the number of Bézier curves needed while keeping good approximation. We take advantage of



(a) (b) (c)

Figure 7: (a) Input images; (b) traditional parameterization method; (c) our optimal parameterization method

Table 2: Comparison between one parameterization and optimal parameterization methods

Segment NO.	Traditional	Our optimal
1	1.168	1.168 (chord)
2	1.144	**1.054**(uniform)
3	1.484	1.484 (chord)
4	1.468	**1.410**(uniform)
5	1.482	1.482 (chord)
6	1.416	1.416 (chord)
7	1.456	1.456 (chord)
8	1.337	**1.222** (centri)

the affine transformation invariant property of Bézier curves and present a method for detecting a symmetric axis or point for an image shape if it exists, which can further reduce the number of curves required. Besides, we apply the bisection method to add suitable break points in an efficient way. Lastly, we also find different parameterization methods for each segment can be adopted to further reduce the fitting error or even the number of curves.

However, there are also limitations to our methods. For example, for the corner points detection step, we just adopt existing methods, which can be improved by developing a more adaptive and robust technique to make it more generable, as the distribution of the corner points has a great impact on the final results. Besides, the image boundaries we deal with are mainly noise-free, more robust methods that can handle noisy data should also be considered in the future.

References

[1] Preeti Joshi. Image vectorization and significant point detection. 2014.

[2] A Kirsanov, A Vavilin, and KH Jo. Contour-based algorithm for vectorization of satellite images. In *International Forum on Strategic Technology 2010*, pages 241–245. IEEE, 2010.

[3] Christine Nadal, Raymond Legault, and Ching Y Suen. Complementary algorithms for the recognition of totally unconstrained handwritten numerals. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume 1, pages 443–449. IEEE, 1990.

[4] Ju Jia Zou and Hong Yan. Cartoon image vectorization based on shape subdivision. In *Proceedings. Computer Graphics International 2001*, pages 225–231. IEEE, 2001.

[5] Michael Plass and Maureen Stone. Curve-fitting with piecewise parametric cubics. In *Proceedings of the 10th annual conference on Computer Graphics and Interactive Techniques*, pages 229–239, 1983.

[6] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[7] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 1996.

[8] Jing-Jing Fang and Chia-Lien Hung. An improved parameterization method for b-spline curve and surface interpolation. *Computer-Aided Design*, 45(6):1005–1028, 2013.

[9] S Mariyam Hj Shamsuddin and Mahmoud Ali Ahmed. A hybrid parameterization method for nurbs. In *Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004.*, pages 15–20. IEEE, 2004.

[10] Philip J Schneider. An algorithm for automatically fitting digitized curves. *Graphics gems*, 1:612–626, 1990.

[11] Theodosios Pavlidis. Curve fitting with conic splines. *ACM Transactions on Graphics (TOG)*, 2(1):1–31, 1983.

[12] Jakob Gonczarowski. A fast approach to auto-tracing (with parametric cubics). In *Raster imaging and digital typography*, volume 91, pages 1–15, 1991.

[13] John M McNamee and Victor Pan. *Numerical Methods for Roots of Polynomials-Part II*. Newnes, 2013.

[14] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (TOG)*, 26(3):11–es, 2007.

[15] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)*, 28(5):1–10, 2009.

[16] Andrea Mazzoleni. https://www.scale2x.it/. accessed Dec. 29, 2022.

[17] Johannes Kopf and Dani Lischinski. Depixelizing pixel art. In *ACM SIGGRAPH 2011 papers*, pages 1–8. 2011.

[18] Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, and Jian Sun. Effective clipart image vectorization through direct optimization of bezigons. *IEEE transactions on visualization and computer graphics*, 22(2):1063–1075, 2015.

[19] Daniel Sỳkora, Jan Buriánek, and Jiri Zara. Sketching cartoons by example. In *SBM*, pages 27–33, 2005.

[20] Vector Magic. https://vectormagic.com/. accessed Jan. 10, 2023.

[21] Yuchen He, Sung Ha Kang, and Jean-Michel Morel. Binary shape vectorization by affine scale-space. *Image Processing On Line*, 13:22–37, 2023.

[22] FontLab. https://www.fontlab.com/font-editor/fontlab/. accessed Dec. 29, 2022.

[23] Hung-Hsin Chang and Hong Yan. Vectorization of hand-drawn image using piecewise cubic bézier curves fitting. *Pattern recognition*, 31(11):1747–1755, 1998.

[24] Sarbajit Pal, Pankaj Ganguly, and PK Biswas. Cubic bézier approximation of a digitized curve. *Pattern recognition*, 40(10):2730–2741, 2007.

[25] Shayan Hoshyari, Edoardo Alberto Dominici, Alla Sheffer, Nathan Carr, Zhaowen Wang, Duygu Ceylan, and I-Chao Shen. Perception-driven semi-structured boundary vectorization. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[26] Jun Xie, Holger Winnemöller, Wilmot Li, and Stephen Schiller. Interactive vectorization. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 6695–6705, 2017.

[27] Industry leading vector graphics software — Adobe Illustrator. https://www.adobe.com/uk/products/illustrator.html. accessed Jan. 10, 2023.

[28] Andrés Iglesias, Akemi Gálvez, and Marta Collantes. Bat algorithm for curve parameterization in data fitting with polynomial bézier curves. In *2015 International Conference on Cyberworlds (CW)*, pages 107–114. IEEE, 2015.

[29] Akemi Gálvez and Andrés Iglesias. A new iterative mutually coupled hybrid ga–pso approach for curve fitting in manufacturing. *Applied Soft Computing*, 13(3):1491–1504, 2013.

[30] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. A brief introduction to opencv. In *2012 proceedings of the 35th international convention MIPRO*, pages 1725–1730. IEEE, 2012.

[31] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The

connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017.