Received: 11 April 2023 | Revised: 6 May 2023 | Accepted: 8 May 2023

DOI: 10.1002/cav.2191

SPECIAL ISSUE PAPER

WILEY

Vectorizing binary image boundaries with symmetric shape detection, bisection and optimal parameterization

Zaiping Zhu | **Lihua You** | **Jian Jun Zhang**

The National Centre for Computer Animation, Bournemouth University, Poole, Dorset, UK

Correspondence
Zaiping Zhu, The National Centre for Computer Animation, Bournemouth University, Poole, Dorset, UK.
Email: s5319266@bournemouth.ac.uk

Funding information
European Union Horizon 2020 Research and Innovation Programme, Grant/Award Number: 778035; China Scholarship Council

Abstract

Binary image boundary vectorization is the process of converting raster images into vector images represented with a sequence of Bézier curves. Two main factors in reconstructing parametric curves are to 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 article, 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

affine transformation, Bézier curve fitting, binary image boundaries, bisection, optimal parameterization, symmetric detection, vectorization

1 | INTRODUCTION

Binary image boundary vectorization is referred to the process of converting raster images represented by two-dimension pixels to vector images represented by a sequence of reconstructed mathematical curves such as widely used Bézier curves, 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. Two main factors in reconstructing mathematical curves for image boundary vectorization are to approximate the underlying structure of the image boundaries as much as possible while using as few numbers of mathematical curves as possible.[5]

There are some varieties in the vectorization pipelines. The pipeline used in this article is based on Reference 6 where cubic Bézier curves are used to fit the boundary of binary images. It 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 classical methods include uniform parameterization, chord length method,[7] centripetal[8,9] and hybrid[10] and so forth. 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[11] treats the point with maximum error as the break point. On the contrary, the point with zero error is chosen as the break point by Pavlidis[12] as he finds "splitting the interval at points where the error is zero is preferable to splitting at points where the error estimate is maximum." The middle point of the segment is also used as the break point in Reference 13. 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, bisection and optimal parameterization. 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.[14] 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. Finally, we investigate the impact of different parameterization methods on fitting the same segment and propose to use different parameterization methods for different segments to reduce the fitting error or even the number of fitting curves, which is named as optimal parameterization in this article.

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,[15,16] pixel art vectorization,[17,18] and artist-generated image vectorization.[19,20] Some methods mainly focus on region boundary vectorization,[21-23] 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 article, 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 Section 1. Specifically, Schneider[11] first locates the corner points by computing the angle between each point and its neighbors, 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 zero error.[12] The middle point of the segment is also used as the break point in Reference 13. 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[24] 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.[25] 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 applying two-dimensional logarithmic and an evolutionary search algorithm. Hoshyari et al.[26] adopt machine learning techniques to vectorize semi-structured boundaries, which are usually distinctly colored 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.[27] present an interactive tool to further refine the output from Image tracer[28] 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.[22] present a method to vectorize binary shape using affine scale-space, which consists of three steps: first, 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 the 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[5,11] 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,[24] as the newly updated parameters may not satisfy $0 = t_0 \leq t_1 \leq \ldots \leq t_n = 1$. 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.[29,30] However, these methods that treat point parameters as free variables have some limitations. First, 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 article, 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 Section 1, but it cannot guarantee the minimum number of curves needed. In this article, we adapt the general pipeline to minimize the number of curves needed while keeping a good approximation. First, 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

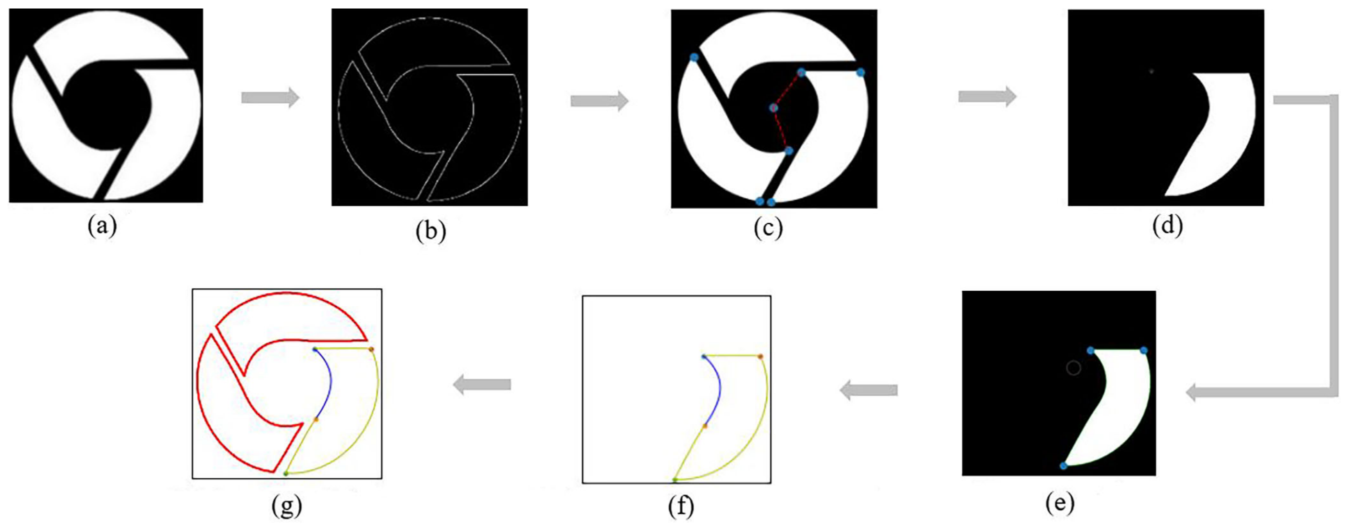

FIGURE 1 Pipeline of our proposed method. (a) Input image. (b) Extracted boundaries. (c) Symmetric detection. (d) Basic part. (e) Corner points detection. (f) Optimal parameterization; adding break points. (g) Final result by symmetric.

parameterization 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 FITTING WITH SYMMETRIC DETECTION AND BISECTION

In order to achieve efficient fitting of piecewise Bézier curves, we first check in Section 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 Sections 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.[31] Next, the principal component analysis 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 principal axes that go through the centric point of the image shape, as shown in Figure 2a, 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 2b,c, the intersection and union shape are almost the same. While for another axis, the ratio between the corresponding intersection and union

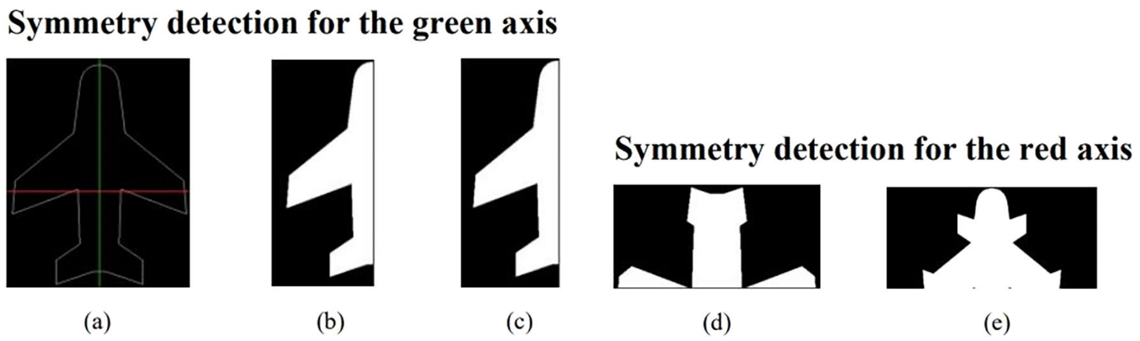
Symmetry detection for the green axis
Symmetry detection for the red axis
(a)
(b)
(c)
(d)
(e)

FIGURE 2 Symmetric axes detection. (a) Boundary and principle axes. (b, d) Intersection. (c, e) Union.

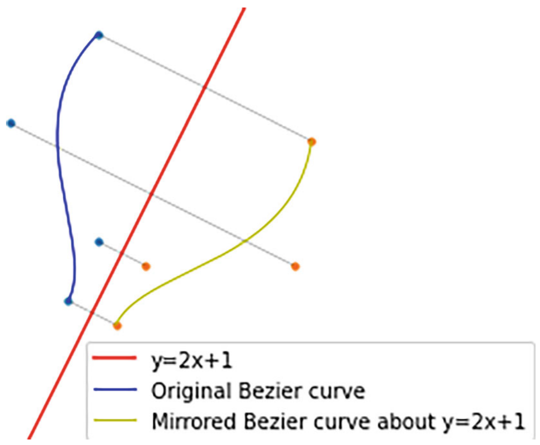

FIGURE 3 A cubic Bézier curve and its mirrored counterpart about an axis.

is much smaller than one, which means the shape is not symmetric about this axis, as shown in Figure 2d,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 $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ 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 $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ 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).

$$p_s = \frac{p * (a^2 - b^2) - 2 * b * (a * q + c)}{a^2 + b^2}, \quad q_s = \frac{q * (b^2 - a^2) - 2 * a * (b * q + c)}{a^2 + b^2}. \tag{1}$$

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.

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 labeling technique[32] 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:

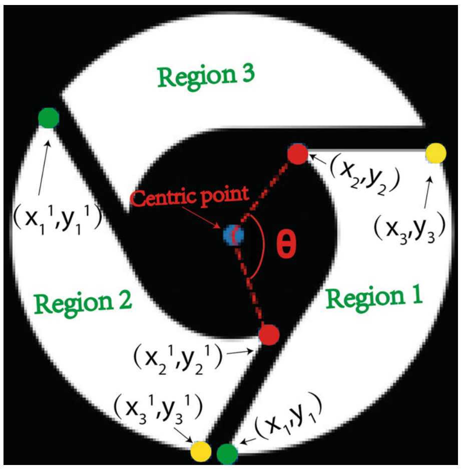

FIGURE 4 Symmetric point detection.

$$\begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \\ x_2 & -y_2 \\ y_2 & x_2 \\ \vdots & \vdots \\ x_t & -y_t \\ y_t & x_t \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_t \\ y'_t \end{bmatrix}, \tag{2}$$

where $(x_1, y_1), (x'_1, y'_1), \dots, (x_t, y_t), (x'_t, y'_t)$ are corresponding corner points for the two regions. This is normally an overdetermined linear system of equations. To find the best solution, we use the pseudoinverse method. After obtaining the potential rotating angle θ, we can rotate one region 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 regular image, which has to be vectorized fully.

4.2 | Cubic Bézier curve fitting

After a symmetric part of a symmetric image boundary is determined, the bisection method investigated in Section 4.3 is used to obtain breaking points of symmetric and unsymmetric image boundaries. Then a cubic Bézier curve $\mathbf{C}(t, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ is used to fit the points between two breaking points or between a corner point and its adjacent breaking points where $\mathbf{p}_i$ $(i = 0, 1, 2, 3)$ are four unknown control points and $0 \le t \le 1$.

Assuming there are N points $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \dots, \mathbf{P}_N$ between two adjacent breaking points or between a corner point and its adjacent breaking point and the two adjacent breaking points or a corner point and its adjacent breaking point are denoted by $\mathbf{P}_0$ and $\mathbf{P}_{N+1}$, respectively, these $N+2$ points are parameterized to obtain the parametric values t_i $(i = 0, 1, 2, \dots, N+1)$.

As a cubic Bézier curve goes through its two end control points, the first point $\mathbf{P}_0$ and the last point $\mathbf{P}_{N+1}$ coincide with the first control point $\mathbf{p}_0$ and the last control point $\mathbf{p}_3$, respectively, that is, $\mathbf{p}_0 = \mathbf{P}_0$ and $\mathbf{p}_3 = \mathbf{P}_{N+1}$. Only the control points $\mathbf{p}_1$ and $\mathbf{p}_2$ are unknown, which can be determined by using the least squared method below.

$$\frac{\partial}{\partial \mathbf{p}_i} \sum_{n=0}^{N+1} \left[\mathbf{C}(t_n, \mathbf{p}_1, \mathbf{p}_2) - \mathbf{P}_n\right]^2 = 0 (i = 1, 2). \tag{3}$$

After the four control points $\mathbf{p}_i$ $(i = 0, 1, 2, 3)$ of a cubic Bézier curve have been determined, the fitting quality can be evaluated with the largest distance between the fitting curve and the input points, which can be obtained using the following expression:

Algorithm 1. Bisection method to determine the break point (**input**:: points in a segment)

```
//comments: In a segment, $l$ is the index of the left point, $r$ is the index of right point, and $m$ is the index of the middle point.
$l_0 = 0$, $r_0 = $ len(points)-1, $l = l_0$, $r = r_0$, $m = (l+r)/2$
if a curve can fit the points between $l_0$ and $r_0$ thenreturn
    // no break point needed
else
    while ($m$ changes) do
        $m = (l+r)/2$
        if a curve can fit points between $l_0$ and $m$ then
            $l = m$
        else
            $r = m$
        end if
    end while
return $m$ as break point index
end if
```

$$Err = \max\{|\mathbf{C}(t_i, \mathbf{p}_1, \mathbf{p}_2) - \mathbf{P}_i|\}(i = 0, 1, 2, 3, \dots, N, N+1). \tag{4}$$

4.3 | Bisection method

As we mentioned in Section 1, the bisection method is applied when one cubic Bézier curve cannot 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. The pseudocode algorithm using the bisection method to find breaking points is presented in Algorithm 1. In the algorithm, we use l_0 to indicate the index of the left point and r_0 to indicate the index of the right point. If l_0 and r_0 are two adjacent corner points and a cubic Bézier curve can fit all the points from l_0 to r_0, no breaking points are added between the two adjacent corner points l_0 and r_0. Otherwise, we find a middle point m so that all the points from l_0 to m can be fitted by a cubic Bézier curve with the following iteration algorithm below. First, we set $l = l_0$ and $r = r_0$. Then we calculate the middle point $m = (l+r)/2$. If a cubic Bézier curve can be used to fit all the points from l_0 to the middle point m, we set $l = m$. Otherwise, we set $r = m$. Then we calculate the new middle point $m = (l+r)/2$ and check whether a cubic Bézier curve can be used to fit all the points from l_0 to the new middle point m. A breaking point m is found by repeating the iteration until the middle point m does not change. After the middle point m becomes a breaking point, we set $l_0 = m$ and check whether all the points from l_0 to r_0 can be fitted by a cubic Bézier curve. If yes, no breaking points are added between l_0 and r_0. Otherwise, the above iteration is used to find another braking point m. This process is repeated until all the points from l_0 to r_0 can be fitted with cubic Bézier curves.

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 5b,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 5d, with which a Bézier curve cannot be determined. However, using the bisection method, the number of curves is further minimized as shown in Figure 5e.

4.4 | Implementation detail

For a binary image whose boundary to be vectorized, the findContours in the open-source library OpenCV is used to detect its boundary. Next, the methods investigated in Section 4.1 are implemented to find a symmetric region. After that,

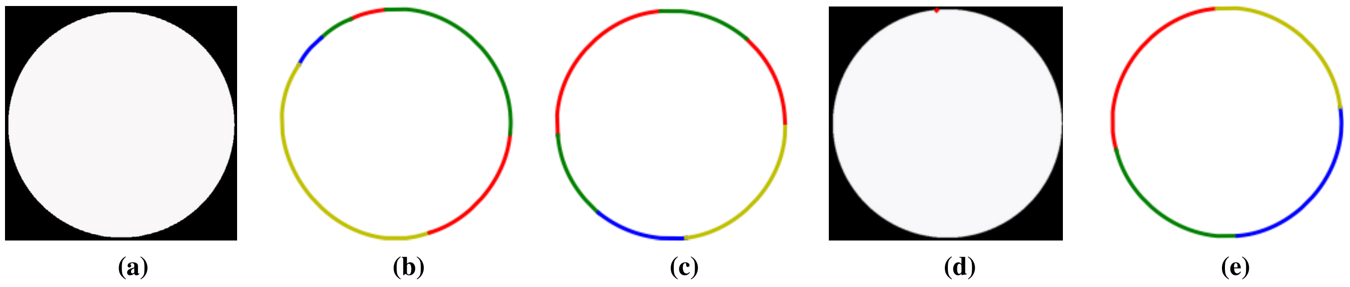

FIGURE 5 (a) Input images; (b) adding break points at positions with maximum error[11] (six curves); (c) positions at middle point[13] (six curves); (d) at positions with minimum error;[12] (e) our bisection method (four curves).

the corner points of the symmetric region for a symmetric image shape or the corner points for a regular image shape are detected with the corner point detection method in the OpenCV library.

To use Equation (3) to conduct cubic Bézier curve fitting and Equation (4) to calculate the maximum error, the points between two corner points or between a corner point and its adjacent point plus their two endpoints must be parameterized. To find the optimal parametrization method, different parameterization methods are used to obtain the point parameterization as discussed below.

Given $N + 2$ points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_N, \mathbf{P}_{N+1}$ of a segment, we calculate the parameter t_i associated with each point using different parameterization methods, such as the chord length method,[7] the centripetal method,[8,9] and the hybrid method[10] among which the mathematical expressions of the centripetal method and the hybrid method respectively have the form of Equations (5) and (6) below.

$$t_i = t_{i-1} + \frac{|\mathbf{P}_i - \mathbf{P}_{i-1}|}{\sum_{l=1}^{N} |\mathbf{P}_i - \mathbf{P}_{i-1}|}, \tag{5}$$

$$t_i = t_{i-1} + \frac{\sqrt{|\mathbf{P}_i - \mathbf{P}_{i-1}|}}{\sum_{l=1}^{N} \sqrt{|\mathbf{P}_i - \mathbf{P}_{i-1}|}}, \tag{6}$$

where $i = 1, 2, 3, \ldots, N$, $t_0 = 0$, $t_{N+1} = 1$, and $|.|$ means the distance between adjacent points.

The parameters t_i $(i = 0, 1, 2, 3, \ldots, N, N + 1)$ obtained with one of the parameterization methods are introduced into Equation (3) to obtain a linear system of equations. The matrix inverse technique is used to solve the linear system and obtain a fitted cubic Bézier curve. The maximum error of the fitted curve is calculated with Equation (4). It is used to evaluate different parameterization methods. The parameterization method whose fitting maximum error is the smallest among the parameterization methods under consideration is identified as the optimal parameterization method.

After that, the identified optimal parameterization method, the cubic Bézier curve fitting, and the bisection method investigated in Section 4.3 are combined to obtain breaking points. This is achieved by introducing the parameters obtained with the optimal parameterization method into Equation (3) to obtain the fitted cubic Bézier curve. The maximum error of the fitted cubic Bézier curve calculated with Equation (4) is used to add break points based on whether the maximum error is very close to but does not exceed the predefined threshold.

The obtained corner points and breaking points divide the boundary of a regular image shape or the boundary of a symmetric region of a symmetric image shape into the fewest number of segments. Each of the segments is fitted with a cubic Bézier curve to obtain the optimal result of vectorizing binary image boundaries.

5 | RESULTS AND COMPARISON

In this section, we present the results of vectorizing image boundaries using our method and compare it with the classical and new methods. We set the error tolerance the same for our method and the new method to make the comparison fair. For the classic method, setting the error tolerance is not available so we keep the default setting.

First, we use some less complex image shapes to compare our method with two existing methods, that is, the classical image tracer method and the new Affine scale-space method. The fitted boundary curves are shown in Figure 6

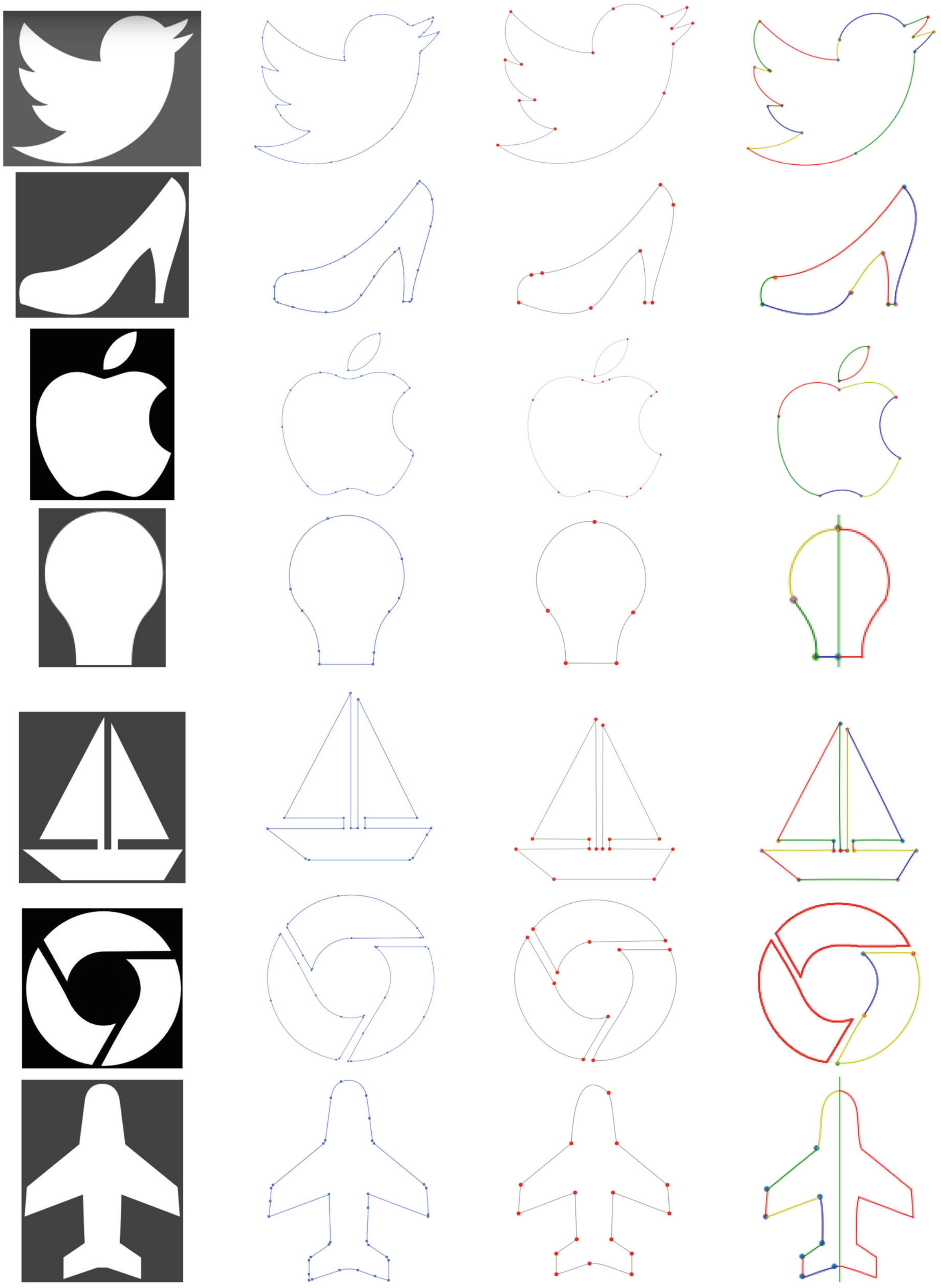

FIGURE 6 Results using different techniques to vectorize the boundary of less complex image shapes. First column: input image. Second column: Image tracer.[28] Third column: Affine scale-space.[22] Last column: our method.

TABLE 1 Number of curves required for our method and other methods for less complex image shapes.

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**

Note: The numbers in bold indicate that our method requires fewer number of curves than the other methods, which are demonstrated in Figure 6 and Figure 7.

and the number of curves needed for each of the input images shown in Figure 6 is given in Table 1. As we can see from Figure 6 and Table 1, our proposed method outperforms the Image tracer method and the Affine scale-space method in terms of using a minimal number of cubic Bézier curves while achieving a similar good approximation.

To further demonstrate the effectiveness and advantages of our proposed method, we compare our method with the Image tracer method and the Affine scale-space method on more complex and diverse image shapes and present the comparison results in Figure 7 and Table 2. Once again, Figure 7 and Table 2 show our proposed method outperforms the Image tracer method and the Affine scale-space method on these complex and diverse image shapes.

The image shapes shown in Figures 6 and 7 are noise-free. For such noise-free image shapes, the above comparisons demonstrate the advantages of our proposed method over the existing methods. We have also used our proposed method, the Image tracer method, and the Affine scale-space method to vectorize the boundary of a noisy image shape shown in Figure 8a and present the results in Figure 8b–d where Figure 8b,c are obtained with the Image tracer method and the Affine scale-space method, respectively, and Figure 8d is obtained with our proposed method. The fitted curves in these figures clearly indicate our proposed method uses fewer curves and achieves better quality, which indicates that our proposed method also outperforms the Image tracer method and the Affine scale-space method in vectorizing the boundary of the noisy image shape.

Lastly, we investigate the impact of different parameterization methods, as shown in Figure 9b,c, each colored 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 3. As we can see, the 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.

Using a small number of curves while keeping tight approximation, that is, high accuracy in converting binary image boundaries into vector representations has many impacts. It has been acknowledged that presenting images as a list of mathematical curves can compress data to provide a more compact representation and reduce data storage, increase data transmission speed, and achieve faster processing to raise computational efficiency.[1] The proposed method obviously reduces the number of cubic Bézier curves to further maximize its impact on these applications. Fitting quality quantified by fitting errors plays an important role in representing original image boundaries accurately. Due to its importance, existing vectorization methods mainly focus on accuracy during the conversion from raster images to vector images.[33] The proposed method has smaller fitting errors than existing methods, demonstrating its advantage in accurately representing original image boundaries.

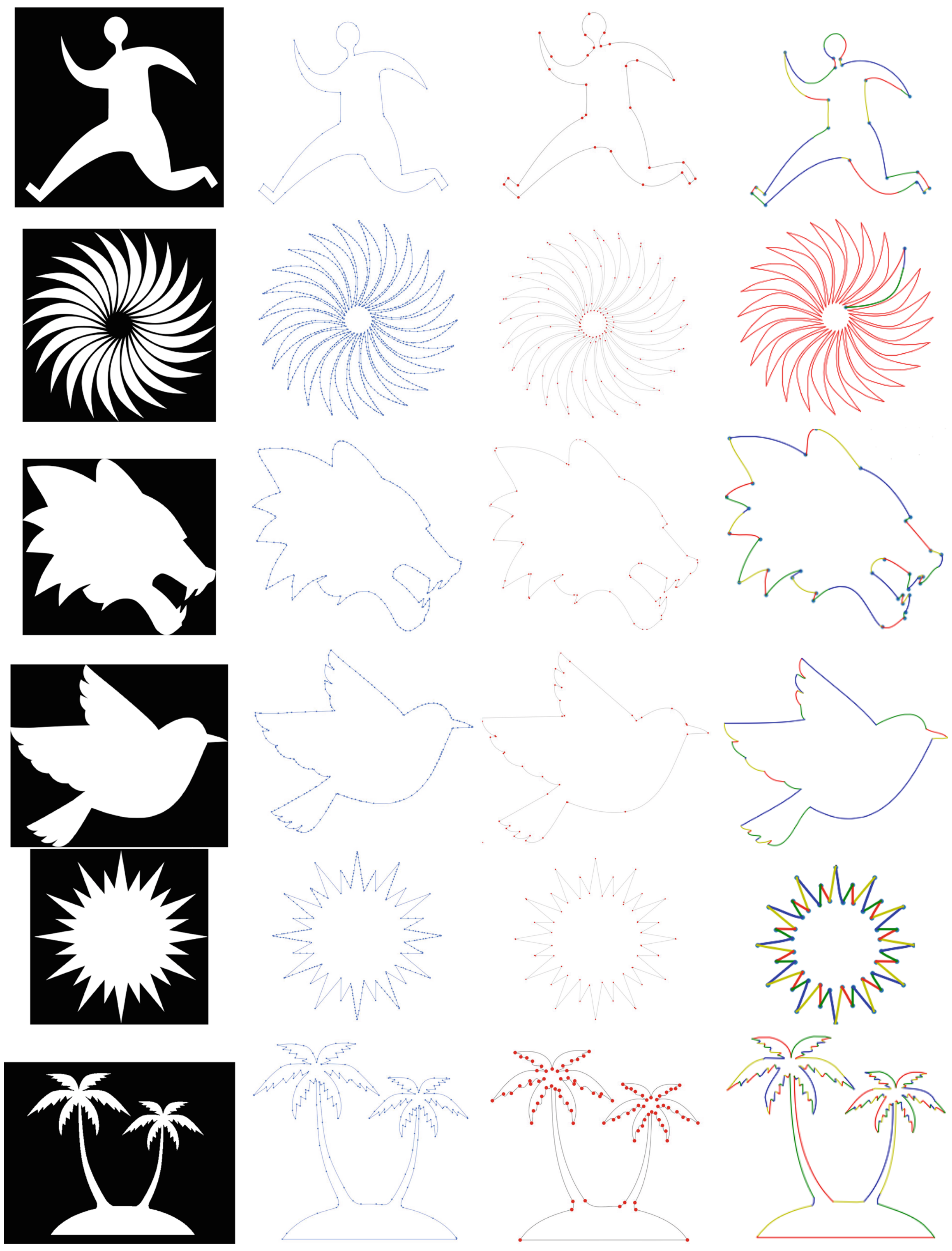

FIGURE 7 Results using different techniques to vectorize the boundary of more complex image shapes. First column: Input image. Second column: Image tracer.[28] Third column: Affine scale-space.[22] Last column: our method.

TABLE 2 Number of curves required for our method and other methods for more complex image shapes.

Input	Image tracer	Affine scale-space	Our method
Man	44	29	**28**
Flower	>126	126	**6**
Wolves	>55	55	**40**
New bird	>36	36	**26**
Star	>52	52	**48**
Tree	>142	Not good (too smooth)	**142**

Note: The numbers in bold indicate that our method requires fewer number of curves than the other methods, which are demonstrated in Figure 6 and Figure 7.

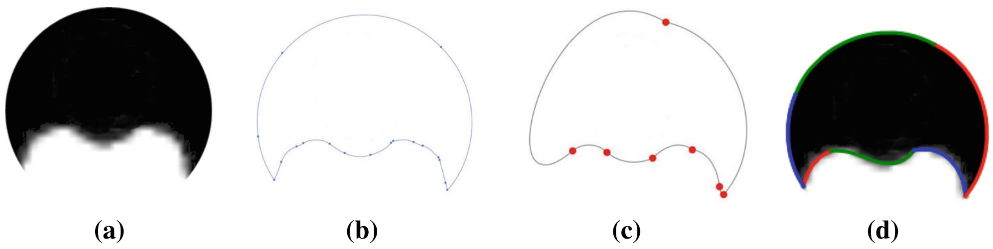

FIGURE 8 (a) Input noise images; (b) image tracer; (c) affine scale-space; (d) our method.

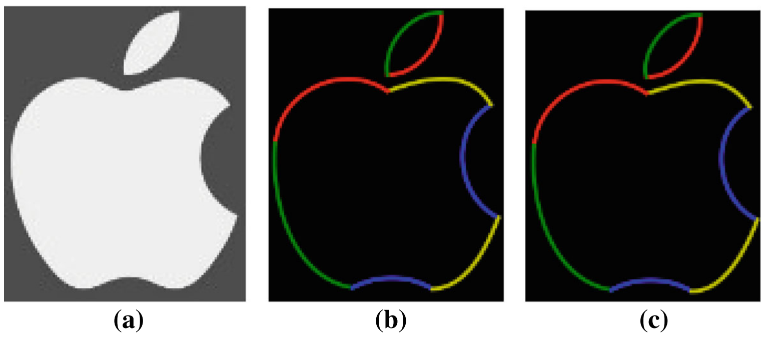

FIGURE 9 (a) Input images, (b) traditional parameterization method, (c) our optimal parameterization method.

TABLE 3 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)

Note: The values in bold indicate that using the optimal parameterization method achieves smaller errors than using one parameterization method, which is also demonstrated in Figure 9.

6 | CONCLUSIONS AND DISCUSSION

In this article, we propose a method which incorporates symmetric image shape detection, bisection, and optimal parameterization to vectorize image boundaries, aiming to minimize the number of Bézier curves needed while keeping good approximation. We take advantage of 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 that can be adopted to further reduce the fitting error or even the number of curves.

However, there are some topics which can be investigated in the future to extend the applicability and further improve the performance of the proposed method. These topics are briefly discussed below.

This article uses the corner point detection method in the open-source library OpenCV to find corner points. It is simple but requires adjusting the parameters for different inputs. To tackle the problem of this method, a more suitable corner point detection method could be introduced to avoid parameter adjustment. Corner point detection has been extensively investigated in existing research studies, and various corner point detection methods have been developed. Recently, Zhang et al. made a comprehensive evaluation of state-of-the-art corner point detection methods and identified several corner point detection methods with top performance.[34] In our following work, we will evaluate these identified methods and implement a more suitable corner point detection method to avoid parameter adjustment.

Quite a number of noise-free image shapes and a noisy image shape have been used to compare our proposed method with the existing methods. All of them demonstrate the advantages of our proposed method over the existing methods. In our following work, we will make more comparisons with more noisy image shapes to demonstrate the good applicability of our proposed method in vectorizing the boundary of both noise-free and nosy image shapes.

The affine transformation includes scaling, translation, rotation, reflection, and shearing. This article only investigates how to take advantage of rotation and reflection properties to improve image boundary vectorization. Other properties such as shearing could be further investigated, which requires developing other techniques to find such patterns. We believe the deep learning technique is a potential tool, which deserves further research.

FUNDING INFORMATION

PDE-GIR project which has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie Grant agreement No. 778035. Zaiping Zhu is also sponsored by the China Scholarship Council.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Zaiping Zhu https://orcid.org/0000-0001-5810-8346

REFERENCES

1. Joshi P. Image vectorization and significant point detection. Proceedings of the International Conference on Advances in Engineering and Technology. AIAA; 2014. p. 74–8.
2. Kirsanov A, Vavilin A, Jo KH. Contour-based algorithm for vectorization of satellite images. Proceedings of 2010 International Forum on Strategic Technology. IEEE; 2010. p. 241–5.
3. Nadal C, Legault R, Suen CY. Complementary algorithms for the recognition of totally unconstrained handwritten numerals. Proceedings of 10th International Conference on Pattern Recognition. IEEE; 1990. p. 443–9.
4. Zou JJ, Yan H. Cartoon image vectorization based on shape subdivision. Proceedings of Computer Graphics International 2001. IEEE; 2001. p. 225–31.
5. Plass M, Stone M. Curve-fitting with piecewise parametric cubics. ACM SIGGRAPH Comput Graph. 1983;17(3):229–39. https://doi.org/10.1145/964967.801153
6. Sarfraz M, Khan M. An automatic algorithm for approximating boundary of bitmap characters. Future Gener Comput Syst. 2004;20(8):1327–36.

7. Farin G. Curves and surfaces for computer-aided geometric design: a practical guide. Amsterdam, The Netherlands: Elsevier; 2014.
8. Piegl L, Tiller W. The NURBS book. Berlin: Springer Science & Business Media; 1996.
9. Fang JJ, Hung CL. An improved parameterization method for B-spline curve and surface interpolation. Comput-Aided Des. 2013;45(6):1005–28.
10. Shamsuddin SMH, Ahmed MA. A hybrid parameterization method for NURBS. Proceedings of International Conference on Computer Graphics, Imaging and Visualization. IEEE; 2004. p. 15–20.
11. Schneider PJ. An algorithm for automatically fitting digitized curves. In: Glassner AS, editor. Graphics gems. Volume 1. San Diego, CA: Academic Press; 1990. p. 612–26.
12. Pavlidis T. Curve fitting with conic splines. ACM Trans Graph. 1983;2(1):1–31.
13. Gonczarowski J. A fast approach to auto-tracing (with parametric cubics). Conference Proceedings on Raster Imaging and Digital Typography II. RIDT 91. Cambridge University Press; 1991. p. 1-15.
14. McNamee JM, Pan V. Numerical methods for roots of polynomials—part II. Amsterdam, The Netherlands: Elsevier Science; 2013.
15. Sun J, Liang L, Wen F, Shum HY. Image vectorization using optimized gradient meshes. ACM Trans Graph. 2007;26(3):11:1–7.
16. Xia T, Liao B, Yu Y. Patch-based image vectorization with automatic curvilinear feature alignment. ACM Trans Graph. 2009;28(5): 1–10.
17. Mazzoleni A. https://www.scale2x.it/ Accessed 29 Dec 2022.
18. Kopf J, Lischinski D. Depixelizing pixel art. ACM Trans Graph. 2011;30(4):99:1–8.
19. Yang M, Chao H, Zhang C, Guo J, Yuan L, Sun J. Effective clipart image vectorization through direct optimization of bezigons. IEEE Trans Vis Comput Graph. 2015;22(2):1063–75.
20. Sỳkora D, Buriánek J, Zara J. Sketching cartoons by example. Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling. The Eurographics Association; 2005. 27–33.
21. Magic V. https://vectormagic.com/ Accessed 10 Jan 2023.
22. He Y, Kang SH, Morel JM. Binary shape vectorization by affine scale-space. Image Process On Line. 2023;13:22–37.
23. FontLab. https://www.fontlab.com/font-editor/fontlab/ Accessed 29 Dec 2022.
24. Chang HH, Yan H. Vectorization of hand-drawn image using piecewise cubic Bézier curves fitting. Pattern Recognit. 1998;31(11): 1747–55.
25. Pal S, Ganguly P, Biswas P. Cubic Bézier approximation of a digitized curve. Pattern Recognit. 2007;40(10):2730–41.
26. Hoshyari S, Dominici EA, Sheffer A, Carr N, Ceylan D, Wang Z, et al. Perception-driven semi-structured boundary vectorization. ACM Trans Graph. 2018;37(4):1–14.
27. Xie J, Winnemöller H, Li W, Schiller S. Interactive vectorization. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery; 2017. p. 6695–705.
28. Industry-leading vector graphics software. Adobe Illustrator. Software. https://www.adobe.com/uk/products/illustrator.html Accessed 10 Jan 2023.
29. Iglesias A, Gálvez A, Collantes M. Bat algorithm for curve parameterization in data fitting with polynomial Bézier curves. Proceedings of 2015 International Conference on Cyberworlds. IEEE; 2015. p. 107–14.
30. Gálvez A, Iglesias A. A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing. Appl Soft Comput. 2013;13(3):1491–504.
31. Culjak I, Abram D, Pribanic T, Dzapo H, Cifrek M. A brief introduction to OpenCV. Proceedings of the 35th International Convention MIPRO. IEEE; 2012. p. 1725–30.
32. He L, Ren X, Gao Q, Zhao X, Yao B, Chao Y. The connected-component labeling problem: a review of state-of-the-art algorithms. Pattern Recognit. 2017;70:25–43.
33. Xiong X, Feng J, Zhou B. Real-time contour image vectorization on GPU. In: Braz J, Magnenat-Thalmann N, Richard P, Linsen L, Telea A, Battiato S, et al., editors. Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. VISIGRAPP 2016. Cham: Springer International Publishing; 2017. p. 35–50.
34. Zhang Y, Zhong B, Sun X. A benchmark for the evaluation of corner detectors. Appl Sci. 2022;12(23):11984. https://doi.org/10.3390/app122311984

AUTHOR BIOGRAPHIES

Zaiping Zhu is Ph.D. candidate at the National Centre for Computer Animation, Bournemouth University, UK. He received his B.Sc. degree in Mechanical Design and Manufacturing Automation from Shandong University, China and a M.Sc. degree in Industrial Design Engineering from Peking University, China. His research interests are related to surface and curve reconstruction from point clouds and images, deep learning, augment reality, and virtual reality.

Lihua You is Professor at the National Centre for Computer Animation, Bournemouth University, UK. He received his Ph.D. degree from Chongqing University, China, and another Ph.D. degree from Bournemouth University, UK. His current research interests are in computer graphics, computer animation, and geometric modeling.

Jian Jun Zhang is Professor of Computer Graphics at the National Centre for Computer Animation, Bournemouth University, UK where he leads the National Research Centre. He is also a co-founder of the UK's Centre for Digital Entertainment, funded by the Engineering and Physical Sciences Research Council. His research focuses on 3D virtual human modeling, animation, simulation, and immersive technology.

How to cite this article: Zhu Z, You L, Zhang JJ. Vectorizing binary image boundaries with symmetric shape detection, bisection and optimal parameterization. Comput Anim Virtual Worlds. 2023;e2191. https://doi.org/10.1002/cav.2191