Iranian Journal of Numerical Analysis and Optimization
Vol. 9, No. 2, (2019), pp 103–121
DOI:10.22067/ijnao.v9i2.70045

Research Article

Capturing outlines of generic shapes with cubic Bézier curves using the Nelder–Mead simplex method

A. Ebrahimi, G. B. Loghmani* and M. Sarfraz

Abstract

We design a fast technique for fitting cubic Bézier curves to the boundary of 2D shapes. The technique is implemented by means of the Nelder–Mead simplex procedure to optimize the control points. The natural attributes of the Bézier curve are utilized to discover the initial vertex points of the Nelder–Mead procedure. The proposed technique is faster than traditional methods and helps to obtain a better fit with a desirable precision. The comparative analysis of our results describes that the introduced approach has a high compression ratio and a low fitting error.

AMS(2010): 65D17; 65D10.

Keywords: Interpolation; Splines; Curve fitting; Nelder–Mead simplex method; Computer aided design; Computer graphics.

1 Introduction

Capturing the outlines of 2D objects is a substantial topic in computer aided geometric designs (CAGD), computer graphics, as well as vision and imaging; see [1, 2, 6, 7, 9–12, 15, 20–25, 32, 38, 39]. Curve fitting with Bézier curves is a traditional problem in this process. The goal of a curve fitting method is to

*Corresponding author
Received 8 January 2018; revised 12 January 2019; accepted 8 March 2019
Alireza Ebrahimi
Computer Geometry and Dynamical Systems Laboratory, Faculty of Mathematical Sciences, Yazd University,Yazd, Iran. e-mail: a.ebrahimi@stu.yazd.ac.ir

Ghasem Barid Loghmani
Computer Geometry and Dynamical Systems Laboratory, Faculty of Mathematical Sciences, Yazd University,Yazd, Iran. e-mail: loghmani@yazd.ac.ir

Muhammad Sarfraz
Department of Information Science, College of Computing Sciences & Engineering, Kuwait University, Kuwait. e-mail: prof.m.sarfraz@gmail.com, muhammad.sarfraz@ku.edu.kw

detect a collection of the control points that can precisely indicate the given target shape.

In the literature, many methods have been suggested to represent outline objects [3, 11, 21, 29, 34]. A vast majority of these methods pertain to an optimization problem that finds an appropriate curve for the data generated by the outlines of 2D shapes. Most common algorithms for solving this model are not suitable optimization algorithms. Since they involve complex computations, they cannot be used for real time applications. Itoh and Ohno [11] exerted the least square fitting to approximate cubic Bézier curves without fixing the end points of the curves. Plass and Stone [21] proposed an iterative procedure for fitting a parametric piecewise cubic polynomial curve with an selective endpoint and tangent vector specifications. Srafraz and Khan [28] employed the least square method to certify an appropriate fit. Their method contains several phases, such as extraction of boundaries, exploration corner points and break points, and fitting curves. In another study, the same authors added reparameterization steps to ameliorate the efficiency of the fit [29]. Sarfraz and Razzak [34] employed a generalized Hermite cubic spline to capture the outline of digital character images using characteristic points and the least squares method. The enhanced Bézier curve scheme proposed by Sohel et al. [36] reduces the distance among the Bézier curve and its control polygon without extra computational complexity. An object coding technique using Bézier curves was introduced by Masood and Haq [16]. They determined the control points by searching along the endpoint tangents. Sarfraz and Masood [30] determined the appropriate position of the control points by using the natural attributes of cubic Bézier curves. Masood and Sarfraz [18] presented an outline capturing scheme using Bézier cubic approximation. Their method operates in two phases to find intermediate control points. In Phase 1, the location of the detected control points gets closer to that of the original control points. Phase 2 is exerted to hit the target location very exactly. Masood and Sarfraz [17] used the properties of cubic Bézier curves and analysed the control points spread (CPspread) to subdivide complex segments into two or more segments. Some other techniques based on the search algorithm include evolutionary algorithm [35], simulated annealing approach [26, 27], genetic algorithm [32, 33], and wavelets [37].

In the above mentioned methods, the control points are specified by the least square algorithm or using the properties of cubic Bézier curves. These processes are computationally expensive and not appropriate for practical applications. Our contribution in this study is to introduce a scheme to avoid these time consuming operations and to guarantee a desirable equivalence between compression ratios and fitting errors. We perform the Nelder–Mead algorithm to find the intermediate control points of the cubic Bézier curve. The initial situation of the control points is specified using extracting some effective virtues of the Bézier curve.

The rest of the paper is arranged as follows. Section 2 reviews the segmentation of an outline by corner detection. Section 3 introduces the Nelder–

Mead algorithm. The process of determining the control points for the cubic Bézier curve approximation using the Nelder–Mead simplex method is described in section 4. Section 5 illustrates the experimental observations and compares them with those of other methods. Finally, the paper is closed up with a conclusion in Section 6.

2 Outline segmentation

Outline segmentation is used to divide the shape outline into small part and simplify the curve fitting procedure. We use the corner points to partition the outline into disparate segments from the natural break points. Authors have suggested various corner detection algorithms in the literatures [3,4,31]. These algorithms do curvature analysis with numerical techniques. In this paper, the algorithm designed by Sarfraz et al. [31] is used for the corner detection, because this algorithm is accurate, effective, and robust to noise. The method is concisely explained in this section (readers are referred to [31] for details). The algorithm detects corner points in two steps. Candidate corner points are discovered from the outline data points in the first step. If all the boundary points are $Q_i, 1 \leq i \leq n$, then the boundary point Q_k can be given as

if $(i + L) \leq n$,
then $Q_k = Q_{i+L}$,
else $Q_k = Q_{(i+L)-n}$,

where L represents the length parameter and preserves of the shape scaling and resolution. The default value of L is 14. The perpendicular distance d_j from point $Q_j(x, y)$ to the direct line joining the points $Q_i(x, y)$ and $Q_k(x, y)$, can be given as follows:

if $m_x = 0$,
then $d_j = |Q_{j,x} - Q_{i,x}|$,
else $d_j = \dfrac{|Q_{j,y} - mQ_{j,x} + mQ_{i,x} - Q_{i,y}|}{\sqrt{m^2 + 1}}$,
where $m = \dfrac{m_y}{m_x} = \dfrac{Q_{k,y} - Q_{i,y}}{Q_{k,x} - Q_{i,x}}$.

The point Q_j is specified as a volunteer corner point if its perpendicular distance (d_j) from the direct line Q_iQ_k goes beyond D. The local sharpness and the opening angle of the corners are investigated by the distance parameter D. It also checks the incorrect determination of corner points that may occur due to noise and disarray. The default value of D is set to be 2.6. The new direct line through increasing both i and k detects the next volunteer

corner points. In the second step, the superfluous corner points are determined. A superfluous corner point is one of that any other volunteer with a higher value of d_j is in the domain R. In the other words, a candidate corner point will stay if it has the greatest value of d_j between the R number of points on its both sides. The default value of R is equal to length parameter. In the present study, we utilize this method at its default values for corner detection. All steps of this corner detection method is shown in Algorithm 1.

Algorithm 1 The corner detection algorithm

Set values of L, D, and R;
Detect boundary points $\{Q_i\}_{i=0}^{n}$ of 2D object using the Canny edge detector;
First step: detect candidate corner points;
for $i = 1$ **to** n **do**
  if $(i + L) \leq n$ **then**
    $Q_k = Q_{i+L}$,
  else
    $Q_k = Q_{(i+L)-n}$;
  end if
  for $j = i$ **to** k **do**
    Calculate d_j the perpendicular distance from point Q_j to the straight line joining points Q_i and Q_k;
    if $d_j > D$ **then**
      Add Q_j to candidate corner point;
    end if
  end for
end for
Second step: detect superfluous corner points;
A candidate corner point will stay if it has the greatest value of d_j between the R number of points on its both sides;

3 The Nelder–Mead simplex method (NMSM)

The Nelder–Mead algorithm is a numerical optimization procedure for solving unconstrained optimization problems in a multidimensional space; see [5,14,19]. This method is one of the enormously famous derivative-free techniques. It is a direct search algorithm that only needs a numerical evaluation of the objective function at a finite number of points per iteration. This method is planed for prevalent unconstrained minimization problems, such as nonlinear least squares and nonlinear simultaneous equations; see [8,13].

The problem discussed in this study is the following unconstrained optimization problem

$$\min f(x) \quad x \in \mathbb{R}^n,$$

where f is a nonlinear function from $\mathbb{R}^n$ into $\mathbb{R}$ and $x \in \mathbb{R}^n$. For the function $f(x)$, the Nelder–Mead method begins with $n+1$ vertices as $y_1, y_2, \ldots, y_{n+1}$ and then evaluates the objective function value of each vertices. We assign to y_1 as the best vertex where $f(x)$ is the lowest and to y_{n+1} as the worst point where $f(x)$ is the highest. The most common Nelder–Mead iterations perform a succession of primary geometric transformations including reflection, expansion, contraction, and shrinkage to find a better point and make it replace the worst point. Geometric transformations are controlled by four parameters including coefficients of reflection ρ, expansion χ, contraction γ, and shrinkage σ. The introduced parameters should assure the following consternations:

$$\rho > 0, \quad \chi > 1, \quad 0 < \gamma < 1, \quad \text{and} \quad 0 < \sigma < 1.$$

The common values, exerted in this paper, are

$$\rho = 1, \quad \chi = 2, \quad \gamma = \frac{1}{2}, \quad \text{and} \quad \sigma = \frac{1}{2}.$$

We set $\epsilon = 10^{-10}$ for the termination criterion in the Nelder–Mead method. The principal foundation of the Nelder–Mead method is described in Algorithm 2. Readers are referred to [8,19] for a detailed study of the Nelder–Mead method.

4 Curve approximation

We partition the outline of a 2D shape into curve segments based on its corner points. It means that all the outline points between two consecutive corner points constitute one curve segment. In this literature, cubic Bézier curves have been used to fit each segment separately using the Nelder–Mead algorithm.

4.1 Bézier curve

The Bézier curve is a parametric curve $C(t)$ that widely used to describe and model curves and surfaces in computer aided designs and computer graphics. The Bézier curve of degree n is given as

Algorithm 2 The Nelder–Mead algorithm [8]

1. Choose an initial $n+1$ vertex points $\{y_1, y_2, \ldots, y_{n+1}\}$ and choose a stopping criterion ϵ;
2. Order. Order and re-label the $n+1$ vertices to assure $f(y_1) \leq f(y_2) \leq \cdots \leq f(y_{n+1})$;
3. Reflect. calculate the reflection point y_r by $y_r = \bar{y} + \rho(\bar{y} - y_{n+1})$, where $\bar{y} = \sum_{i=1}^{n}(\frac{y_i}{n})$ is the centroid of the n best points;
if $f(y_1) \leq f(y_r) < f(y_n)$ **then**
replace y_{n+1} with the reflected point y_r and go to Step 7;
end if
4. **Expand.**
if $f(y_r) < f(y_1)$ **then**
calculate the expansion point y_e by $y_e = \bar{y} + \chi(y_r - \bar{y})$;
end if
if $f(y_e) < f(y_r)$ **then**
exchange y_{n+1} with y_e and go to Step 7;
else
exchange y_{n+1} with y_r and go to Step 7;
end if
5. Contract.
if $f(y_r) \geq f(y_n)$ **then**
a contraction is performed among $\bar{y}$ and the better of y_{n+1} and y_r;
end if
a. Outside.
if $f(y_n) \leq f(y_r) < f(y_{n+1})$ **then**
apply an outside contraction: compute $y_{oc} = \bar{y} + \gamma(y_r - \bar{y})$;
end if
if $f(y_{oc}) \leq f(y_r)$ **then**
exchange y_{n+1} with y_{oc} and go to Step 7;
else
go to Step 6 (perform a shrink);
end if
b. Inside.
if $f(y_r) \geq f(y_{n+1})$ **then**
apply an inside contraction: compute $y_{ic} = \bar{y} + \gamma(y_{n+1} - \bar{y})$;
end if
if $f(y_{ic}) \geq f(y_{n+1})$ **then**
exchange y_{n+1} with y_{ic} and go to Step 7;
else
go to Step 6 (perform a shrink);
end if
6. Shrink. Measure the n new vertices
$y' = y_1 + \sigma(y_i - y_1), i = 2, \ldots, n+1.$
exchange the vertices $y_2, \ldots, y_{n+1}$ with the new vertices $y'_2, \ldots, y'_{n+1}$;
7. Termination Criterion. Arrange and re-label the vertices of the new simplex as $y_1, y_2, \ldots, y_{n+1}$ such that $f(y_1) \leq f(y_2) \leq \cdots \leq f(y_{n+1})$;
if $f(y_{n+1}) - f(y_1) < \epsilon$ **then**
stop;
else
go to Step 3.
end if

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), \quad 0 \le t \le 1,$$

where P_i refers to the control points and $B_{i,n}(t)$ is the Bernstein basis polynomials of index i and degree n expressed as

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n.$$

From the defining equation of the Bézier curve, it can be seen that the properties of Bernstein polynomials are passed on to the Bézier curve; see [20]. These properties are as follows:

- Partition of unity.
- Affine invariance.
- Convex hull property.
- Endpoint interpolation.
- Value of Bernstein polynomials, for all $0 \le t \le 1$, does not appertain to the location of any control point(s); see [17].
- If the location of any control point is specified, its efficacy all over the curve can be deprived; see [17].

4.2 Curve fitting with the cubic Bézier curve

Let $\{X_i\}_{i=1}^{N} \subset \mathbb{R}^2$ denote an ordered set of contour points of a segment. Our goal is to find control points $P_i \quad (i = 0, 1, 2, 3)$ so that the cubic Bézier curve

$$C_3(t) = P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t), \qquad 0 \le t \le 1,$$

can be a good representation of $\{X_i\}_{i=1}^{N}$. For this purpose, we minimize the linear least squares fitting error, E, specified as the sum of the squares of the deviations:

$$E = \sum_{k=1}^{N} ||C_3(t_k) - X_k||^2 = \sum_{k=1}^{N} \left\| \sum_{i=0}^{3} P_i B_{i,3}(t_k) - X_k \right\|^2,$$

where the parameter values t_i are assigned to X_i using the uniform parameterization method; see [20]. The control points P_0 and P_3 are the two corner

points of the segment. So, the effect of P_0 and P_3 is removed from a given cubic Bézier curve as

$$E = \sum_{k=1}^{N} ||C_3'(t_k) - X_k'||^2, \tag{1}$$

where

$$C_3'(t) = P_1 B_{1,3}(t) + P_2 B_{2,3}(t)$$

and

$$X_k' = X_k - (P_0 B_{0,3}(t_k) + P_3 B_{3,3}(t_k)), \quad k = 1, 2, ..., N.$$

Therefore, we have to find the control points P_1 and P_2 by minimizing the problem defined in (1).

Let $P_i = (P_{x_i}, P_{y_i}), i = 1, 2$, and write all the intermediate control points in one vector:

$$x = \begin{bmatrix} P_{x_1} \\ P_{y_1} \\ P_{x_2} \\ P_{y_2} \end{bmatrix}.$$

With this vector, the fitting problem is formulated as

$$\min_x \sum_{k=1}^{N} ||C_3'(t_k) - X_k'||^2. \tag{2}$$

That is, there are four parameters to estimate. We employ the Nelder–Mead method to solve the least square fitting problem defined in (2) and find these parameters.

4.3 Initialization

Before Algorithm 2 is implemented, a starting point should be selected for the intermediate control points. The initial vertex points can be chosen arbitrarily, but using the appropriate initial situation of the control points increases the speed of convergence and quality of solutions. Duo to $x \in \mathbb{R}^4$ in the objective function (2), five vertices have to be chosen. The details on the selection of the initial vertex points are explained as follows.

Let

$$P_1 B_{1,3}(t) + P_2 B_{2,3}(t) = \overline{C}_3(t), \tag{3}$$

where

$$\overline{C}_3(t) = C_3(t) - P_0 B_{0,3}(t) - P_3 B_{3,3}(t).$$

Based on the properties of the cubic Bézier curves, we get

$$B = B_{1,3}(0.5) = B_{2,3}(0.5). \tag{4}$$

From equations (3) and (4), we have

$$P_1 + P_2 = \widetilde{C}_3, \tag{5}$$

where

$$\widetilde{C}_3 = \frac{\overline{C}_3(0.5)}{B}.$$

By solving equations (3) and (5), we have

$$\begin{cases} P_1 = \dfrac{\overline{C}_3(t) - B_{2,3}(t)\widetilde{C}_3}{B_{1,3}(t) - B_{2,3}(t)}, \\ P_2 = \widetilde{C}_3 - P_1. \end{cases}$$

Suppose ${P_1}^i$ and ${P_2}^i$ are the computed intermediate control points at $(0 \leq t_i \leq 1, t_i \neq 0, 0.5, 1,)$, and

$$\begin{cases} {P_1}^i = \dfrac{\overline{C}_3(t_i) - B_{2,3}(t_i)\widetilde{C}_3}{B_{1,3}(t_i) - B_{2,3}(t_i)}, \\ {P_2}^i = \widetilde{C}_3 - {P_1}^i. \end{cases}$$

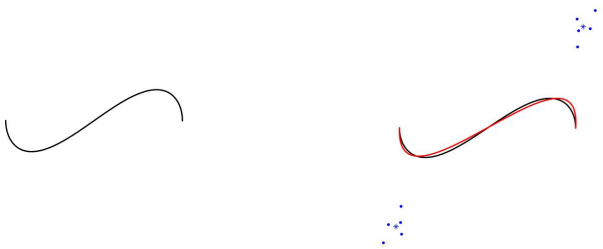
where $i = 1, 2, \ldots, 5$.

Let $P_1^i = (P_{x_1}^i, P_{y_1}^i)$ and $P_2^i = (P_{x_2}^i, P_{y_2}^i)$. The vector x_i can be identified as

$$x_i = \begin{bmatrix} P^i_{x_1} \\ P^i_{y_1} \\ P^i_{x_2} \\ P^i_{y_2} \end{bmatrix}, \quad i = 1, 2, \ldots, 5.$$

Hence, these vectors can be used for the initial vertex points of the Nelder–Mead method.

In order to illustrate the curve fitting using the Nelder–Mead algorithm, we consider a curve as shown in Figure 1(a). Figure 1(b) shows the fitting of a given curve (black) using a cubic Bézier curve (red). The initial vertex points are marked by $\bullet$, and the intermediate control points computed by the Nelder–Mead algorithm are marked by $*$. The fitting error versus the number of iterations for this curve with different initial vertex points is demonstrated in Figure 2. As shown in Figure 2, the Nelder–Mead method based on the proposed initial vertex points converges faster than the one based on random initial vertex points.



(a) Given curve. (b) Cubic Bézier approximation.

Figure 1: Fitting a given curve (black) using a cubic Bézier curve (red) with the proposed algorithm (NMSM).

4.4 Segment subdivision

When curve fitting is done by means of the proposed method (i.e. NMSM), there is a possibility for a generated complex curve not be acceptable and for the deviation error to be high. In such a case, the segment subdivision is applied to decrease the fitting deviation. In this paper, the complex segment of the outline is partitioned into two parts at the point of the maximum deviation error if the maximum deviation error oversteps the given error threshold limit (ETL). The process of subdivision will continue constantly and recursively until the fitting error reaches a value under the given error threshold. An example of subdivision into five curves is demonstrated in

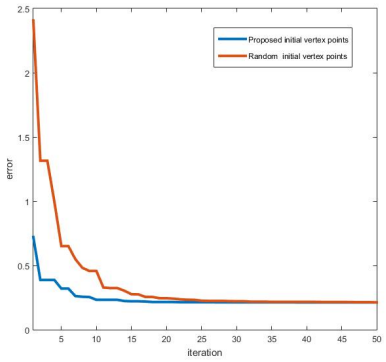
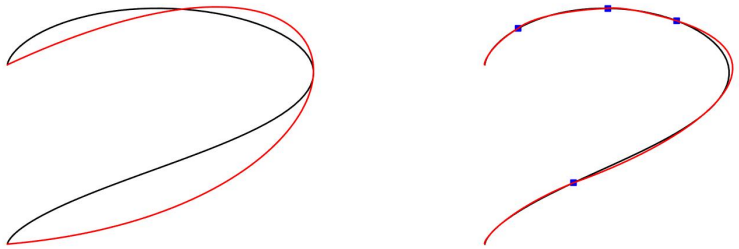


Figure 2: Error vs iteration.

Figure 3. Figure 3(a) is the curve fitting without segment subdivision and Figure 3(b) is curve fitting after segment subdivision. The black and red lines display the original and the approximated curves, respectively. The subdivided points are marked by ■. All the steps of the outline capturing system are shown in Algorithm 3.



(a) Without subdivision. (b) After subdivision.

Figure 3: Curve fitting with recursive subdivision.

Algorithm 3 The outline capturing system

```
Get a digitized image and choose an error threshold limit (ETL) ;
Extract an outline;
Detect the corner points and divide the outline into segments;
for each segment do
    Compute the initial vertex points;
    Perform curve fitting over the segments using the Nelder–Mead algorithm;
    Compute the maximum deviation error (MDE);
    if MDE ≤ ETL then
        break
    else
        Find the subdivision point and partition into two segments;
    end if
end for
```

5 Experimental results and comparative study

The proposed algorithm explained in the Section 4 has been used to some explanatory examples corresponding to generic shapes. Then, the outputs have been compared with those of algorithms [17, 29, 34]. In order to allow a fair comparison, we have tested the same shapes of the criterion as in other papers. In this study, the results are measured based on the following parameters:

- Number of segments: The outline of the shape is partitioned into a number of segments using a corner detection algorithm and subdivision points. An increase in the number of segments, leads to an increase in the number of control points. Therefore, the number of segments must be decreased.

- Compression ratio (CR): This is a significant criterion in the evaluation of the amount of compression accomplished by the method. A large compression ratio is favorable. This criterion indicates the ratio of the number of data points in an actual outline (n) to the obtained data points in curve fitting (nDP). It can be computed as:

$$CR = \frac{n}{nDP}.$$

- Average error: This parameter refers to all the errors generated in the fitted outline of the shape and is given as:

$$\text{Average error } = \frac{1}{n}\sum_{i=1}^{n} e_i,$$

where e_i is the Euclidean distance between the ith point of the outline and the corresponding point of the parametric curve.

- Maximum deviation: This parameter calculates the maximum deviation of the fitted outline from the original shape. It is calculated by the following equation:

$$\text{Maximum deviation } = \max_{i=1}^{n}\{e_i\}$$

- Computation time: This parameter indicates the amount of time taken by the algorithm for capturing the outline of the shape. It appertains to the performance approach and the processor used.

The algorithm has been performed and compared for Figures 4 and 5, which are Arabic words "Sabr", and "Kanji" characters, respectively. Figure 4(a) shows the image of the Arabic word "Sabr" and its boundary is given in Figure 4(b). Figure 4(c) shows the boundary along with the corner points marked with a square (■). Figure 4(d) shows the captured outline based on our approach, and the subdivision points are denoted with a circle (•). The outline computed by means of the method presented by Masood and Sarfraz [17] is shown in Figure 4(e). Figures 4(f) and 4(g), respectively, represent the outline evaluated by Sarfraz and Razzak [34] at a threshold of 3 and 2. The quantitative comparison of the different methods in Figure 4 can be observed in Table 1.

The results obtained for the "Kanji" character are presented in Figure 5 and tabulated in Table 2. The image of "Kanji" character and its boundary are shown in Figures 5(a) and 5(b), respectively. Figure 5(c) shows the boundary along with the corner points which are marked with a square (■). The results of the presented method for this shape are given in Figure 5(d). The outline captured with algorithm [17] is shown in Figure 5(e). The result obtained by Sarfraz and Khan [29] can be seen in Figure 5(f). The outline generated by means of method [34] is given in Figure 5(g). It should be noted that the results of Tables 1 and 2 are achieved by different systems.

Through a visual inspection of the results in Tables 1 and 2, one may come to the following conclusions:

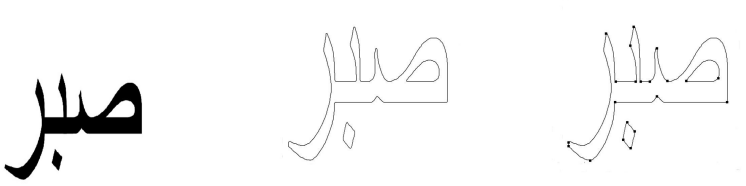
Table 1: Quantitative comparison for the Arabic word "Sabr".

Algorithm	Number of segments	Compression ratio	Maximum deviation	Average error	Computation time
Proposed (NMSM)	28	46.43	1.92	0.44	0.79
Masood and Sarfraz [17]	27	48.15	2.21	1.38	0.83
Sarfraz and Razzak [34] at $\tau = 3$	66	19.70	1.71	1.07	3.79
Sarfraz and Razzak [34] at $\tau = 2$	98	13.27	1.39	0.91	2.58

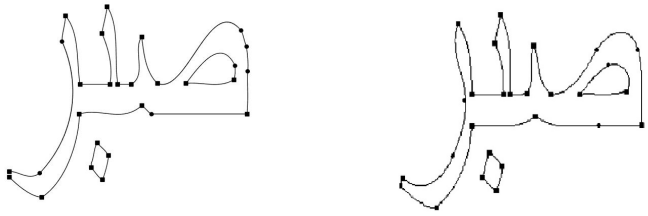
Table 2: Quantitative comparison for the "Kanji" character.

Algorithm	Number of segments	Compression ratio	Maximum deviation	Average error	Computation time
Proposed (NMSM)	33	64.81	1.53	0.40	0.55
Masood and Sarfraz [17]	33	64.81	1.55	0.73	0.59
Sarfraz and Khan [29]	31	69	3.78	1.92	3.61
Sarfraz and Razzak [34]	66	33.42	1.40	1.13	3.83

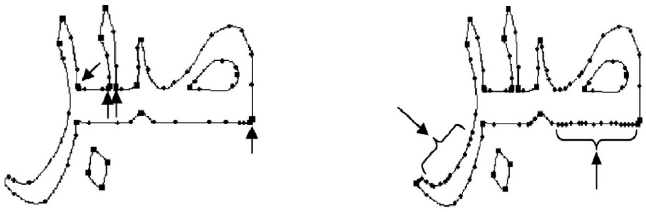
- The method proposed in this study protects an appropriate equilibrium among the compression ratio and the fitting error.
- The proposed method avoids time consuming procedures, such as least square fitting, and uses one of the popular derivative-free free operations (i.e., the NelderMead simplex method) that is simple and fast.
- The average error, maximum deviation, and the computation time are lesser than the results obtained by other methods in the current literature [17, 29, 34].
- The number of segments generated by method [34] is very high because the corner detection and the subdivision algorithms are suboptimal. The suboptimal corner points are indicated by arrows in Figures 4(f), 4(g), and 5(g). This method also suffers from a long computation time. This is due to the use of least squares fitting and control parameters.
- Computation of the outline by means of method [29] takes too long and has errors. The least squares curve fitting and the noise filtering procedure are the main causes for these disadvantages. It is shown with an arrow in Figure 5(f).
- The average error with algorithm [17] is higher than that in the proposed method, because this method does not use the standard optimization process.



(a) Original image. (b) Boundary of original image. (c) Boundary with corner points.

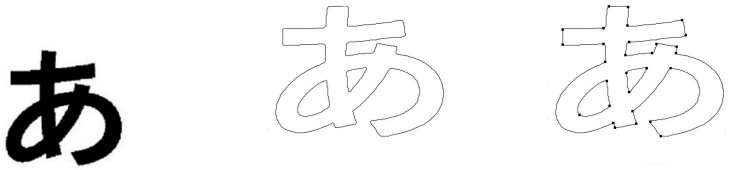


(d) Proposed method (NMSM). (e) Masood and Sarfraz [17].

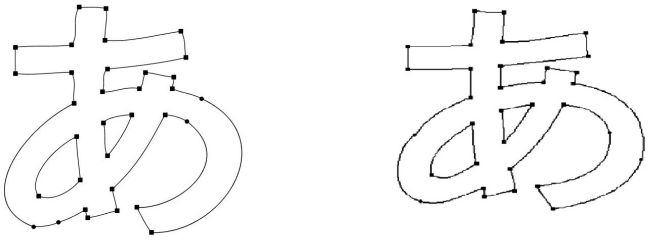


(f) Sarfraz and Razzak [34] at $\tau = 3$. (g) Sarfraz and Razzak [34] at $\tau = 2$.

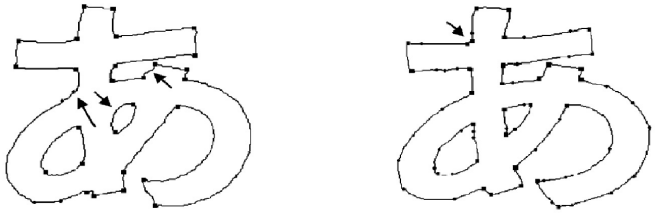
Figure 4: Captured shape of the Arabic word "Sabr" with different methods.



(a) Original image. (b) Outline of original image. (c) Boundary with corner points.



(d) Proposed method (NMSM). (e) Masood and Sarfraz [17].



(f) Sarfraz and Khan [29]. (g) Sarfraz and Razzak [34].

Figure 5: Captured shape of the Kanji character with different methods.

6 Conclusion

In this study, we introduced a novel outline capturing scheme for 2D shapes based on the Nelder–Mead simplex method. The Nelder–Mead simplex algorithm is straight to perform, easy to calculate, and does not call for much time to compute the optimization problem. The unique feature of this method is that it does not require to consider time consuming least squares fitting. The initial situation of the control points is obtained through the properties of

the cubic Bézier curve. It increases the speed of convergence and the quality of curve fitting. As a result, the introduced approximation scheme is much faster than traditional approaches and helps to capture a better fit with a desirable precision. Through a comparison of our method with previous approaches and considering the simulation results, it emerges that the method has such advantages as appropriate equivalency between the compression ratio and the fitting error, low deviation error, and low computation time.

References

1. Bézier, P. *Numerical control; mathematics and applications*, John Wiley & Sons, 1972.

2. Biswas, S. and Lovell, B.C. *Bézier and splines in image processing and machine vision*, Springer Science & Business Media, 2007.

3. Cabrelli, C.A. and Molter, U.M. *Automatic representation of binary images,* IEEE Trans. Pattern. Anal. Mach. Intell., 12 (1990), 1190–1196.

4. Chetverikov, D. and Szabo, Z. *A simple and efficient algorithm for detection of high curvature points in planar curves,* proc. of 23rd workshop of Australian Pattern Recognition Group, Steyr, (1999), 175–184.

5. Conn, A.R., Scheinberg, K. and Vicente, L.N. *Introduction to derivative-free optimization*, MOS-SIAM Series on Optimization, 2009.

6. Farin, G. *Curves and surfaces for computer-aided geometric design: a practical guide*, Academic Press, 1997.

7. Freeman, H. *On the encoding of arbitrary geometric configurations,* IEEE Trans. Comput., 2 (1961), 260–268.

8. Hassanien, A.E., Grosan, C. and Tolba, M.F. *Applications of intelligent optimization in biology and medicine: Current trends and open problems*, Springer, 2015.

9. Hussain, M., Hussain, M.Z., and Sarfraz, M. *Shape-preserving polynomial interpolation scheme,* Iran. J. Sci. Technol. Trans. A Sci., 40 (2016), no. 1, 9–18.

10. Hussain, M.Z., Hussain, F. and Sarfraz, M. *Shape-preserving positive trigonometric spline curves,* Iran. J. Sci. Technol. Trans. A Sci. 42 (2018), no. 2, 1–13.

11. Itoh, K. and Ohno, Y.*A curve fitting algorithm for character fonts,* Electronic publishing , 6 (1993), no. 3, 195–205.

12. Khan, M.S., Ayob, A. F.M., Isaacs, A. and Ray, T. *A novel evolutionary approach for 2d shape matching based on B-spline modeling,* Proc. Congr. Evol. Comput., (2011), 655–661.

13. Klein, K. and Neira, J. *Nelder–Mead simplex optimization routine for large-scale problems: A distributed memory implementation,* Comput. Econ., 4 (2014), 447–461.

14. Lagarias, J.C., Reeds, J.A., Wright, M.H. and Wright, P.E. *Convergence properties of the nelder–mead simplex method in low dimensions,* SIAM. J. Optim. , 9 (1998), no. 1, 112–147.

15. Marji, M. and Siy, P. *A new algorithm for dominant points detection and polygonization of digital curves,* Pattern. Recognit., 10 (2003), 2239–2251.

16. Masood, A. and Haq, S.A. *Object coding for real time image processing applications,* Pattern Recognition and Image Analysis. ICAPR 2005. Lecture Notes in Computer Science, vol 3687. Springer, Berlin, Heidelberg.

17. Masood, A. and Sarfraz, M. *An efficient technique for capturing 2d objects,* Comput. Graph., 32 (2008), no 1, 93–104.

18. Masood, A. and Sarfraz, M. *Capturing outlines of 2d objects with bézier cubic approximation,* Image Vis. Comput., 6 (2009), 704–712.

19. Nelder, J. A., and Mead, R. *A simplex method for function minimization,* Comput. J., 7 (1965), no. 4, 308–313.

20. Piegl, L. and Tiller, W. *The NURBS book*. Springer Science & Business Media, 2012.

21. Plass, M. and Stone, M. *Curve-fitting with piecewise parametric cubics*, Comput. Graph., 17 (1983), no 3, 229–239.

22. Powell, S. *Applications and enhancements of aircraft design optimization techniques*, PhD thesis, University of Southampton, 2012.

23. Salomon, D. *Curves and surfaces for computer graphics*, Springer Science & Business Media, 2007.

24. Sarfraz, M. *Some algorithms for curve design and automatic outline capturing of images,* Int. J. Image. Graph., 2 (2004), 301–324.

25. Sarfraz, M. *Computer-aided intelligent recognition techniques and applications*, Wiley Online Library, 2005.

26. Sarfraz, M. *Vectorizing outlines of generic shapes by cubic spline using simulated annealing,* Int. J. Comput. Math., 8 (2010), 1736–1751.

27. Sarfraz, M. *Capturing image outlines using simulated annealing approach with conic splines,* The Proceedings of the International Conference on Information and Intelligent Computing, (2011), 152–157.

28. Sarfraz, M. and Khan, M. *Automatic outline capture of arabic fonts,* Inf. Sci., 3 (2002), 269–281.

29. Sarfraz, M. and Khan, M. *An automatic algorithm for approximating boundary of bitmap characters,* Future. Gener. Comput. Syst., 8 (2004), 1327–1336.

30. Sarfraz, M. and Masood, A. *Capturing outlines of planar images using bézier cubics,* Comput. Graph., 5 (2007), 719–729.

31. Sarfraz, M., Masood, A. and Asim, M.R. *A new approach to corner detection,* Computer Vision and Graphics 32 (2006), 528533

32. Sarfraz, M. and Raza, A. *Visualization of data with spline fitting: a tool with a genetic approach,* CISST, 97(2002), 99–105.

33. Sarfraz, M. and Raza, S.A. *Capturing outline of fonts using genetic algorithm and splines,* Proceedings of IEEE International Conference on Information Visualization-IV'2001-UK, USA:IEEE Computer Society Press, (2001), 738–743.

34. Sarfraz, M. and Razzak, M. *An algorithm for automatic capturing of the font outlines,* Comput. Graph., 5 (2002), 795–804.

35. Sarfraz, M., Riyazuddin, M. and Baig, M. *Capturing planar shapes by approximating their outlines,* J. Comput. Appl. Math., 1 (2006), 494–512.

36. Sohel, F.A., Karmakar, G.C., Dooley, L.S. and Arkinstall, J. *Enhanced bezier curve models incorporating local information,* Proc. IEEE. Int. Conf. Acoust. Speech. Signal. Process., 4 (2005), 253–256.

37. Tang, Y. Y., Yang, F., and Liu, J. *Basic processes of chinese character based on cubic b-spline wavelet transform,* IEEE. Trans. Pattern. Anal. Mach. Intell., 12 (2001), 1443–1448.

38. Zheng, W., Bo, P., Liu, Y., and Wang, W. *Fast B-spline curve fitting by L-BFGS,* Comput. Aided. Geom. Des., 7 (2012), 448–462.

39. Zhu, F. *Geometric Parameterisation and Aerodynamic Shape Optimisation*, PhD thesis, University of Sheffield, 2014.