JCBI
Journal of Computing & Biomedical Informatics Volume 10 Issue 01
ISSN: 2710 - 1606 2025

Research Article
https://doi.org/10.56979/1001/2025/1105

Designing Fonts with Trigonometric B*e*´Zier Functions

Sareena Shahid[1], Uzma Bashir[1*], and Shazia Javed[1]

[1]Department of Mathematics, Lahore College for Women University, Lahore, Pakistan.
[*]Corresponding Author: Uzma Bashir. Email: uzma.bashir@lcwu.edu.pk

Received: August 04, 2025 **Accepted:** October 28, 2025

Abstract: Designing and fitting fonts play an important role in computer graphics and are widely used in printing, digital media, animation, and computer systems. This study aims to create accurate font outlines minimizing geometric deviation between the detected and fitted contours. The process includes scanning the font to obtain a digital image, detecting its boundary, finding corner points, dividing the boundary into segments, applying the chord length method for parameterization, identifying break points, and fitting the final outline. Trigonometric Bézier functions are used for fitting because they provide better control over the curves of complex characters than traditional Bézier functions. An algorithm has been developed to automatically fit the outlines of selected Arabic and English characters, showing improved accuracy and smoothness in the results.

Keywords: Bézier Curves; Contour Optimization; Curve Fitting and Interpolation; Trigonometric Splines

1. Introduction

Fonts play an important role in field of media, graphics, and animation and computer systems. Font is an array of presentable and displayable characters of design and size. There are almost 300,000 individual fonts in world. Fonts enhance the worth of text, make it legible and understandable. Curve fitting is the most powerful and widely used analysis technique. It is the process of fitting outlines of given shapes, surfaces, curves, and data points. This technique is applied to have smooth and optimal outline of 2D and 3D images. Like images, outlines of fonts are also fitted. Thus, fitting fonts has a fundamental importance in computer graphics.

Usually, computer based font systems use two representations: Bitmap and Outline. In bitmap fonts, each character is stored as a fixed grid of pixels [9], which limits flexibility in scaling or transformation. On the other hand, outlines fonts use smooth parametric curves, like Bézier curves, to manipulate and resize the contours of characters [7]. However, the polynomial nature of Bézier curves makes it difficult to capture the smooth, connected shapes of Arabic letters, which the proposed trigonometric approach aims to overcome. Since Arabic is understood by nearly half the world's population and includes complex scripts, this method can greatly enhance the precision and visual quality of digital Arabic fonts.

Trigonometric Bézier functions are an extension of traditional Burnstein Bézier functions that are composed of trigonometric functions. These functions allow modelling of periodic, circular arcs or oscillatory shapes more accurately as compared to polynomial Bézier functions. For the reason Bézier like trigonometric functions, has become more popular in the field of computer graphics and geometric modelling [1].

This paper provides an algorithmic approach for fitting fonts outline using trigonometric Bézier functions. The technique is very advantageous for designing and it does not involve any human interaction. In traditional method, images are obtained in gray scale form or from any electronic device [5]. After that, contour of the character is extracted [10]. It includes threshold, morphology techniques [13]. Boundary detection leads to determination of corner (high curvature) points which partition the outline into different

parts. They are determined by using method of chain encoding [2] [6], curvature technique [12] and numerical approaches [3] [8]. At times, only corner points are not enough for optimal fit so break points are determined [11]. After that, Bézier functions are applied to get outline fit. The technique used in this study determines break points using chord length parameterization and least squares. Trigonometric Bézier functions have been applied to get outline fit.

2. Materials and Methods

The proposed methodology for boundary extraction and outline fit follows a structured sequence. The process begins with digitizing the character image, followed by boundary extraction to isolate the outline. Corner points are then detected to mark key shape transitions. Using these points, trigonometric Bézier functions are applied for smooth curve fitting. Finally, break points are identified where additional refinement is needed to improve accuracy. The steps may be repeated iteratively until satisfactory results are achieved.

2.1. Digitized Image

The foremost step involved in font fitting method is to get the image that may be obtained from electronic devices or can be plotted. Plotted images have less noise. Resolution of electronic devices, source and type of image are the factors upon which quality of digitized image depends. Digitized images of daal, yea and h are shown in Fig. 1, Fig. 2 and Fig. 3 respectively.

Figure 1. Digitized Image of Arabic character daal

Figure 2. Digitized Image of Arabic character yea

Figure 3. Digitized image of English letter h

2.2. Boundary Extraction

The second step involved in this process is to obtain the boundary/contour of the proposed image. Colored images are first converted into binary form. In this paper, Eddin's method with some modification has been utilized for detection of boundary. This results in several boundary points $(\boldsymbol{a_i}, \boldsymbol{b_i}), \boldsymbol{i} = \boldsymbol{1}, \boldsymbol{2}, \boldsymbol{3}, \ldots, \boldsymbol{n}$ which are also termed as contour points. The detected boundary of characters, yea and h has been shown in Fig. 4, 5 and 6, respectively. These boundaries consist of a single piece only.

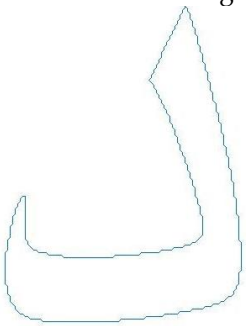

Figure 4. Detected contour of Arabic character daal

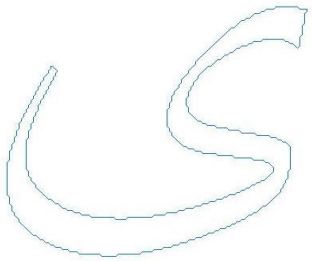

Figure 5. Detected contour of Arabic character yea.

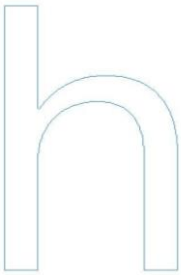

Figure 6. Detected contour of character English letter h

2.3. Corner Detection

The points of high curvature are known as corner points. These points divide the contour of given character into multiple parts. Each part is termed as a segment. The distance between two adjacent corner points forms a segment. Corner points are detected using boundary points determined in boundary extraction process. MATLAB was used to detect the corner points of characters. The detection parameters worked reliably for different Arabic letters, showing that the method produces steady and consistent results. Character daal having 7 corner points has been partitioned into 7 segments while yea and h have been partitioned into 11 and 12 segments respectively depending upon corner points. Corner points have been highlighted with the symbol *.

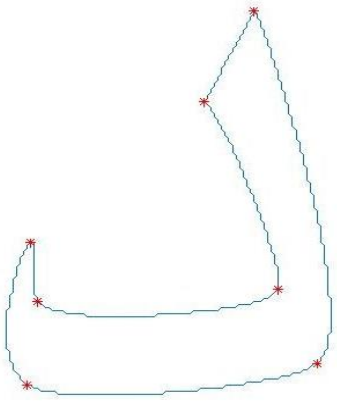

Figure 7. Corner points of character daal

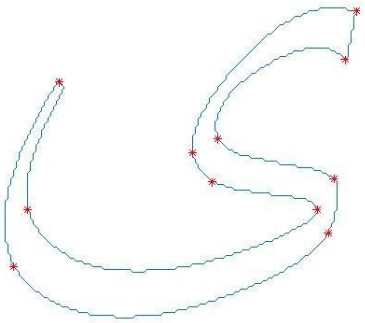

Figure 8. Corner points of character yea

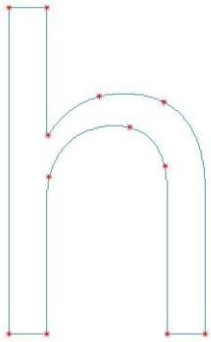

Figure 9. Corner points of character h.

2.4. Fitting of Outline

In this step, trigonometric Bézier functions have been used to each segment of the boundary individually to get proper outline.

Trigonometric Bézier equation is defined

$$\boldsymbol{\Lambda(\kappa) = \sum_{\omega=0}^{3} \varsigma_{\omega}(\kappa)\chi_{\omega}}\,, \kappa \in [0,1] \qquad (1)$$

where the basis polynomials are [1].

$$\left.\begin{aligned} \varsigma_0 &= (1-\sin\Theta)^2 \\ \varsigma_1 &= 2\sin\Theta\,(1-\sin\Theta) \\ \varsigma_2 &= 2\cos\Theta\,(1-\cos\Theta) \\ \varsigma_3 &= (1-\cos\Theta)^2 \end{aligned}\right\}$$

Substituting expression of basis polynomials, equation (5.1) takes following form:

$$\boldsymbol{\Lambda(\kappa) = (1-\sin\Theta)^2\chi_0 + 2\sin\Theta\,(1-\sin\Theta)\chi_1 + 2\cos\Theta\,(1-\cos\Theta)\chi_2 + (1-\cos\Theta)^2\chi_3} \qquad (2)$$

Where $\boldsymbol{\chi_0}$ and $\boldsymbol{\chi_3}$ are known end (corner) points of each segment while $\boldsymbol{\chi_1}$ and $\boldsymbol{\chi_2}$ are termed as intermediate points which are to be determined. In each segment, these four points are used for outline fitting and hence outline of whole boundary is fitted.

2.4.1. Intermediate points detection.

For determination of intermediate points, trigonometric equation (5.2) is generalized in coordinate form as given in equation (5.3). In this formulation, the parameter $\boldsymbol{\kappa}$ represents the normalized position parameter that varies along the curve, controlling the interpolation between boundary points. Its value defines the location of each intermediate point within the segment. The chord length parameterization method has been used to compute $\boldsymbol{\kappa}$ at all boundary points for accurate and uniform point distribution.

$$\left.\begin{aligned} \boldsymbol{\Lambda_x(\kappa)} &= \boldsymbol{(1-\sin\Theta)^2\chi_{x_0} + 2\sin\Theta\,(1-\sin\Theta)\chi_{x_1} + 2\cos\Theta\,(1-\cos\Theta)\chi_{x_2} + (1-\cos\Theta)^2\chi_{x_3}} \\ \boldsymbol{\Lambda_y(\kappa)} &= \boldsymbol{(1-\sin\Theta)^2\chi_{y_0} + 2\sin\Theta\,(1-\sin\Theta)\chi_{y_1} + 2\cos\Theta\,(1-\cos\Theta)\chi_{y_2} + (1-\cos\Theta)^2\chi_{y_3}} \end{aligned}\right\} \quad (3)$$

2.4.2. Chord Length Method

This method has been utilized for determination of parameter κ at all boundary points. Its value lies between 0 and 1. At initial point of each segment, parameter value is 0 while at the other end it is 1. The formula of the corresponding method is given in equation (5.4):

$$\kappa_\alpha = \begin{cases} 0 & \text{if } \alpha = 0 \\ \dfrac{|\chi_1\chi_2| + |\chi_2\chi_3| + \cdots + |\chi_{\alpha-1}\chi_\alpha|}{|\chi_1\chi_2| + |\chi_2\chi_3| + \cdots + |\chi_{\lambda-1}\chi_\lambda|} & \text{if } \alpha < \lambda \\ 1 & \text{if } \alpha = \lambda \end{cases} \quad (4)$$

Determination of parametric value at every point leads to the evaluation of intermediate points.

2.4.3. Least Square Approximation

In this method, distance formula defined by equation (5.5) is applied between contour points and parametric curve for approximation of the curve.

$$\Delta = \sum_{\Pi=1}^{\sigma}[\Lambda_\Pi(\kappa) - \chi_\Pi]^2. \quad (5)$$

Employing the coordinate form, equation (5.5) takes the following form:

$$\Delta = \sum_{\Pi=1}^{\sigma}\left[\Lambda_{x_\Pi}(\kappa) - \chi_{x_\Pi}\right]^2 + \sum_{\Pi=1}^{\sigma}\left[\Lambda_{y_\Pi}(\kappa) - \chi_{y_\Pi}\right]^2 \quad (6)$$

Here $\Lambda(\kappa) = \left(\Lambda_x(\kappa), \Lambda_y(\kappa)\right)$ represents parametric curve and $\chi_\Pi = \left(\chi_{x_\Pi}, \chi_{y_\Pi}\right)$ represents corresponding boundary points.

The main goal is to minimize Δ given in equation (5.2). For this purpose, partial derivative of Δ with respect to χ_1 and χ_2 are taken equal to zero. That is:

$$\frac{\partial\Delta}{\partial\chi_1} = 0 \quad \frac{\partial\Delta}{\partial\chi_2} = 0 \quad (7)$$

Equation (5.5) gives,

$$\left.\begin{aligned} \frac{\partial\Delta}{\partial\chi_1} &= 2\sum_{\Pi=1}^{\sigma}\frac{\partial\Lambda(\kappa_\Pi)}{\partial\chi_1}[\Lambda(\kappa_\Pi) - \chi_\Pi] \\ \frac{\partial\Delta}{\partial\chi_2} &= 2\sum_{\Pi=1}^{\sigma}\frac{\partial\Lambda(\kappa_\Pi)}{\partial\chi_2}[\Lambda(\kappa_\Pi) - \chi_\Pi] \end{aligned}\right\} \quad (8)$$

From equation (5.2), we have,

$$\left.\begin{aligned} \frac{\partial\Lambda(\kappa)}{\partial\chi_1} &= 2\,sin\,\theta\,(1 - sin\,\theta) \\ \frac{\partial\Lambda(\kappa)}{\partial\chi_2} &= 2\,cos\,\theta\,(1 - cos\,\theta) \end{aligned}\right\} \quad (9)$$

Using equations (5.8) and (5.9), equation (5.7) yields:

$$\frac{\partial\Delta}{\partial\chi_1} = 2\sum_{\Pi=1}^{\sigma}\varsigma_1(\kappa_\Pi)[\Lambda(\kappa_\Pi) - \chi_\Pi] = 0 \quad (10)$$

$$\frac{\partial\Delta}{\partial\chi_2} = 2\sum_{\Pi=1}^{\sigma}\varsigma_2(\kappa_\Pi)[\Lambda(\kappa_\Pi) - \chi_\Pi] = 0 \quad (11)$$

Equivalently:

$$\left.\begin{aligned} \sum_{\Pi=1}^{\sigma}\varsigma_1(\kappa_\Pi)[\Lambda(\kappa_\Pi) - \chi_\Pi] &= 0 \\ \sum_{\Pi=1}^{\sigma}\varsigma_2(\kappa_\Pi)[\Lambda(\kappa_\Pi) - \chi_\Pi] &= 0 \end{aligned}\right\} \quad (12)$$

Taking

$$\mu_\phi = \sum_{\Pi=1}^{\sigma}\left[\varsigma_\phi(\kappa_\Pi)\right]^2$$

$$\mu_{1,2} = \sum_{\Pi=1}^{\sigma}\varsigma_1(\kappa_\Pi)\varsigma_2(\kappa_\Pi)$$

$$\Omega_\Phi = \sum_{\Pi=1}^{\sigma}\left[\varsigma_\phi(\kappa_\Pi)[\chi_\Pi - \varsigma_0(\kappa_\Pi)\Lambda_0 - \varsigma_3(\kappa_\Pi)\Lambda_3]\right]$$

In coordinate form, we have

$$\left.\begin{aligned} \Omega_{x_\Phi} &= \sum_{\Pi=1}^{\sigma}\left[\varsigma_\phi(\kappa_\Pi)\left[\chi_{x_\Pi} - \varsigma_0(\kappa_\Pi)\Lambda_{x_0} - \varsigma_3(\kappa_\Pi)\Lambda_{x_3}\right]\right] \\ \Omega_{y_\Phi} &= \sum_{\Pi=1}^{\sigma}\left[\varsigma_\phi(\kappa_\Pi)\left[\chi_{y_\Pi} - \varsigma_0(\kappa_\Pi)\Lambda_{y_0} - \varsigma_3(\kappa_\Pi)\Lambda_{y_3}\right]\right] \end{aligned}\right\} \quad (13)$$

where,

$\varsigma_0 = (1 - \sin\Theta)^2$ and $\varsigma_3 = (1 - \cos\Theta)^2$

To determine χ_1 and χ_2, equations (5.11) are solved for χ_{x_1} and χ_{x_2}

$$\mu_1\chi_{x_1} + \mu_{1,2}\chi_{x_2} = \Omega_{x_1}$$
$$\mu_{1,2}\chi_{x_1} + \mu_2\chi_{x_2} = \Omega_{x_2}$$

and for χ_{y_1} and χ_{y_2}

$$\mu_1\chi_{y_1} + \mu_{1,2}\chi_{y_2} = \Omega_{y_1}$$
$$\mu_{1,2}\chi_{y_1} + \mu_2\chi_{y_2} = \Omega_{y_1}$$

In coordinate form, we have

$$\chi_{x_1} = \frac{\mu_2\Omega_{x_1} - \mu_{1,2}\Omega_{x_2}}{\mu_1\mu_2 - {\mu_{1,2}}^2} \quad (14)$$

$$\chi_{x_2} = \frac{\mu_1 \Omega_{x_2} - \mu_{1,2} \Omega_{x_1}}{\mu_1 \mu_2 - {\mu_{1,2}}^2} \quad (15)$$

Similarly

$$\chi_{y_1} = \frac{\mu_2 \Omega_{y_1} - \mu_{1,2} \Omega_{y_2}}{\mu_1 \mu_2 - {\mu_{1,2}}^2} \quad (16)$$

$$\chi_{y_2} = \frac{\mu_1 \Omega_{y_2} - \mu_{1,2} \Omega_{y_1}}{\mu_1 \mu_2 - {\mu_{1,2}}^2} \quad (17)$$

All these four points given in equations (5.12) through (5.15) are used in fitting outline of each segment and hence all the boundaries are fitted optimally. Figures. 10, 11 and 12 respectively show fitted outlines of Arabic characters daal and yea and English letter h. Digitized curve is represented by narrow line and fitted curve is represented by solid line. These figures reveal that outlines of characters are not desirably fitted, mainly due to limited corner and break point detection at highly curved segments. These deviations highlight the need for additional refinement in point selection to achieve closer correspondence between the fitted and original contours.

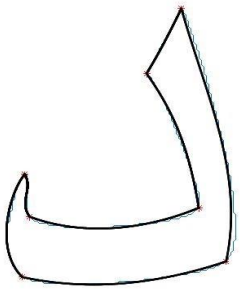

Figure 10. Outline of character daal

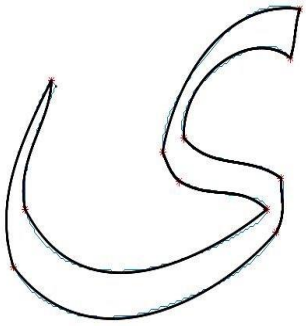

Figure 11. Outline of character yea

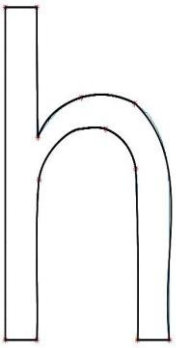

Figure 12. Outline of character h

2.4.4. Determination of Break points

During curve fitting, it was observed that least-squares fitting alone could not accurately follow the complex shapes of Arabic characters, especially in regions with sharp transitions or high curvature. At

times, corner points were not sufficient for achieving an optimal curve fit. To improve accuracy, additional points were identified, known as break points. Corner points together with these break points are referred to as significant points. The break points can be detected using:

$$\boldsymbol{\gamma_\vartheta}^2 = [|\boldsymbol{\chi_\vartheta} - \boldsymbol{\Lambda}(\boldsymbol{\kappa_\vartheta})|]$$

In coordinate form,

$$\boldsymbol{\gamma_\vartheta}^2 = \left[\boldsymbol{\chi}_{x_\vartheta} - \boldsymbol{\Lambda}_x(\boldsymbol{\kappa_\vartheta})\right]^2 + \left[\boldsymbol{\chi}_{y_\vartheta} - \boldsymbol{\Lambda}_y(\boldsymbol{\kappa_\vartheta})\right]^2$$

Among all the points, a point of maximum distance is considered. That is:

$$\boldsymbol{\gamma_{max}} = \boldsymbol{Max}(\boldsymbol{\gamma_1}, \boldsymbol{\gamma_2}, \boldsymbol{\gamma_3}, \ldots)$$

If the maximum point value exceeds threshold limit, the point is termed as break point. Break points subdivide a segment into more segments and fitting method is again applied once again. These points usually occur where the curve experiences sharp transitions or local deviations, indicating areas where the initial fitting was over-smoothed or could not follow the boundary precisely. Figures. 13, 14 and 15 shows detected break points of characters.

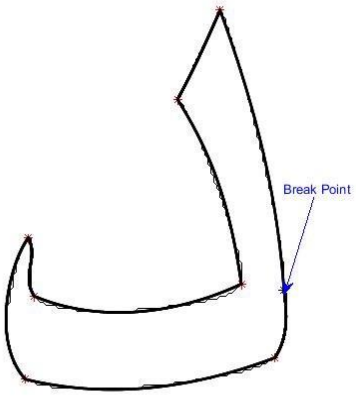

Figure 13. Detected break point of Arabic character daal

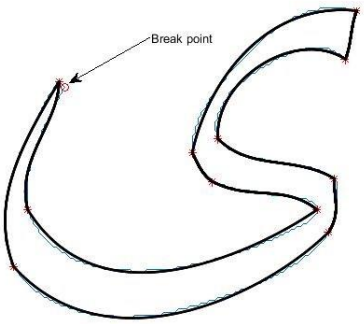

Figure 14. Break points of character yea.

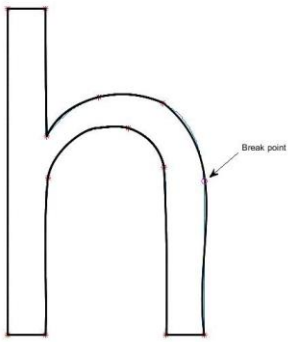

Figure 15. Break points of letter h

5. Conclusion

This study presents an enhanced method for font outline fitting by introducing trigonometric Bézier functions in place of conventional polynomial forms. The proposed technique provides improved control over complex curves and maintains better smoothness across character outlines. Experimental results showed improvements in continuity and accuracy, particularly for Arabic scripts with complex and connected shapes. These findings show that trigonometric Bézier functions can offer a reliable alternative for achieving higher precision in digital font representation and design.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bashir, U. and Ali, J.M., Rational cubic trigonometric Bézier curve with two shape parameters. *Computational and Applied Mathematics*, 2016, 35(1): 285-300.
2. Beus, H.L. and Tiu, S.S., An improved corner detection algorithm based on chain coded plane curves. *Pattern* Recognition, 1987, 20(3): 291-296.
3. Chetverikov, and D., A simple and efficient algorithm for detection of high curvature points in planar curves. In International Conference on Computer Analysis of Images and Patterns, Springer, Berlin, Heidelberg, 2003, 746-753.
4. Davis, L.S., Shape matching using relaxation techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979, (1): 60-72.
5. Karow, P., Digital typefaces: description and formats. Springer Science and Business Media, (2012).
6. Liu H.C. and Srinath M.D., Corner detection from chain-code. Pattern Recognition, 1990, 23(1-2): 51-68.
7. Loop, C.; Blinn, J. Resolution-independent curve rendering using programmable graphics hardware. ACM SIGGRAPH 2005 Papers, 1000–1009. https://doi.org/10.1145/1186822.1073328.
8. Quddus A., Fahmy, M. M., Wavelet transformation for gray level corner detection. Pattern Recognition, 28(6): 853-861, (1995).
9. Rougier, N.P. Higher quality 2D text rendering. Journal of Computer Graphics Techniques, 2013, 2(2), 50-64.
10. https://jcgt.org/published/0002/02/01
11. Sarfraz, M. and Khan, M. A., An automatic algorithm for approximating boundary of bitmap characters. Future Generation Computer Systems, 2004, 20: 1327-1336.
12. Sarfraz, M. and Khan, M.A., Automatic outline capture of Arabic fonts. Information Sciences, 2002, 140 (6): 269-281.
13. Sarfraz, M., Masood, A. and Asim, M.R., A new approach to corner detection. In Computer vision and graphics. Springer, Dordrecht,2006, (528-533).
14. Yahya, F., Ali, J.M., Majid, A.A. and Ibrahim, A., An Automatic Generation of G1 Curve Fitting of Arabic Characters. International Conference on Computer, 2006.