2008

Finding significant points for parametric curve generation techniques

Mohammed Riazur Rahman

M Ameer Ali
East West University

Golam Sorwar
Southern Cross University

Publication details

Rahman, MR, Ali, MA & Sorwar, G 2008, 'Finding significant points for parametric curve generation techniques', *IETECH Journal of Advanced Computations*, vol. 2, no. 2, pp. 107-116.

IETECH Journal of Advanced Computations, Vol: 2, No: 2, 107 - 116

FINDING SIGNIFICANT POINTS FOR PARAMETRIC CURVE GENERATION TECHNIQUES

Md. Riazur Rahman
Dept of Computer Science & Engineering,
Daffodil International University,
Bangladesh.

M. Ameer Ali
Dept of Electronics and Communication Engineering,
East West University,
Bangladesh

G. Sorwar
School of Commerce and Management,
Southern Cross University
Australia

ABSTRACT

The existing significant point generation techniques such as convex hull are unable to find the proper significant points for irregular shaped objects. To address this issue, an algorithm namely finding significant points for parametric curve generation techniques (FSPP) has been proposed in this paper that is able to find out proper significant points which will further be used to produce required shape descriptor using parametric curve generation techniques. Experimental results confirm the superior performance of FSPP algorithm over convex hull in approximating significant points for all types of both regular and irregular shaped objects.

Keywords*: Shape descriptor, convex hull, Bezier curve, significant points, pattern recognition, and image processing.*

1. INTRODUCTION

Image processing is the most important field in computer vision, image understanding, and coding [1-3]. In a wide variety of image processing and pattern recognition applications such as shape descriptor for characters and objects, surface mapping through to active lip shape modelling, and face recognition [4-9], parametric curve generation techniques i.e. *Bezier curve* (BC) [10-14] are used to generate shape descriptors (shape contour points) for a given set of *significant point* (SP). These SPs can be either given manually or generated automatically for the respective shape. For the automatic generation of SPs for the curve generation techniques [10-11] convex hull [15-18] is used. However, the problem is that convex hull generates SPs in such a way that it is unable to find concave curves properly. As a consequence, if BC is applied on these generated SPs, it generates a distorted shape or contours. To address this issue, this paper presents an algorithm, called *finding significant points for parametric curve generation techniques* (FSPP), which automatically generates the SPs effectively even though the shape is either convex or concave i.e. for all types of shapes. The FSPP algorithm finds SP based on the perpendicular distance of a shape point from the line passing through two respective SPs. A perceptual threshold $(T_{\max})$ is used to identify the SP. If a parametric curve generation technique such as BC is used to generate shape contour points using SPs generated by FSPP, it always provide better curve representation than that of generated by convex hull.

This paper is organized as follows: Section 2 describes the mathematical modelling of Bezier curve while Section 3 details the basic idea of convex hull. The FSPP algorithm together with its constituent three algorithms is presented in Section 4 with the experimental results in Section 5. Finally, some conclusions are provided in Section 6.

2. BEZIER CURVE

The theory of *Bezier curve* (BC) is provided in [10-14]. The BC generates contour points considering global shape information, with the curve always passing through the first and last SPs. The degree of the Bezier polynomial depends on the number of SPs, from which a blending function is used to produce the position vector $P(t)$. If there are $L+1$ SPs, the position is defined as

$P_k : (x_k, y_k), \quad 0 \le k \le L$ considering 2-D shapes. These coordinate points are then blended to form $P(t)$, which describes the path of Bezier polynomial function between P_0 and P_L:

$$P(t) = \sum_{k=0}^{L} P_k BEZ_{k,L}(t) \qquad (1)$$

Where the Bezier blending function $BEZ_{k,L}(t)$ is known as the Bernstein polynomial, which is defined as:

$$BEZ_{k,L}(t) = \binom{L}{k} t^k (1-t)^{L-k} \qquad (2)$$

The recursive formula used to determine coordinate positions is:

$$BEZ_{k,L}(t) = (1-t) BEZ_{k,L-1}(t) + t BEZ_{k-1,L-1}(t) \qquad (3)$$

Where, $BEZ_{k,k}(t) = t^k$ and $BEZ_{0,k}(t) = (1-t)^k$.

The individual BC coordinates are represented by the following pair of parametric equations:

$$x(t) = \sum_{k=0}^{L} x_k BEZ_{k,L}(t) \qquad (4)$$

$$y(t) = \sum_{k=0}^{L} y_k BEZ_{k,L}(t) \qquad (5)$$

An example of a BC is provided in **Figure 1** with 4 SPs [13].

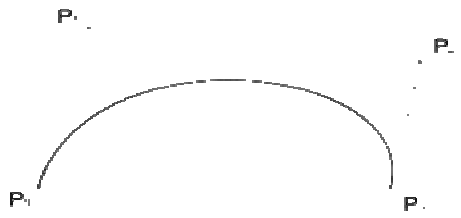


Fig. 1 Curve generated by Bezier Curve

However, since BC considers all the SPs to generate the curve points and hence unable to produce complex curves correctly and smoothly. To represent complex curves higher degree Bezier curves (BCs) can be used. But higher degree BCs suffer from the problem of computational complexity and undulation and can produce unwanted oscillations [10, 11]. To avoid these problems, alternatively, *composite Bezier curve* (CBC) [10-14] is used to produce the desired level of curves and which was also used in the experiments for the proposed FSPP algorithm. CBC uses the technique of joining several lower degree BC segments together to approximate a shape. Joining smaller segments also gives better control over the shape of the curve in smaller regions. To ensure the smoothness of the curve, continuity of the BC segments should be maintained.

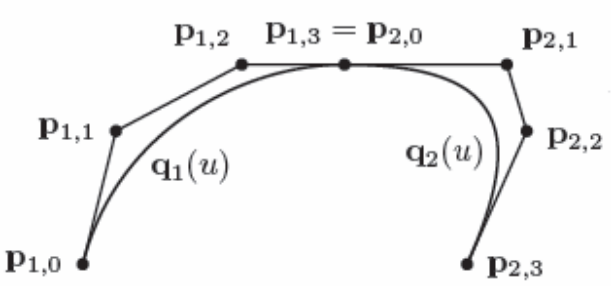


Fig. 1: Curve generated by composite Bezier curve

For a given set of SPs from the original shape, a CBC can be formed by subdividing the SPs to smaller sets of SPs each of which can be used to produce lower degree Bezier segments and then can be joined together according to the continuity constraints. Mathematically, for n given SPs, m sets of SPs can be produced each having (n/m) SPs which can produce a CBC joining m BC segments. A suitable example of a CBC is presented in **Figure 2** [14] joining two BC segments.

3. CONVEX HULL

Computing a convex hull [15-18] is one of the first sophisticated geometry algorithms. It works based on determining the smallest convex set containing a discrete set of points. Convex set (C) of a set of points S is the set such that for all x and y in C and t in interval {0, 1} the point at position $(1-t)x + ty$ is in C. Thus, each point on the line segment joining x and y resides within

C which implies that convex set is connected. The convex hull of an object is the minimal convex set containing that object. The convex hull of a set of points S in n dimensions is the intersection of all convex sets containing S. The convex hull can be described as the convex combination of the points in S. For N points $p_1, p_2, \ldots, p_N$ in S, the convex hull is given by the following expression:

$$CH \equiv \left\{ \sum_{i=1}^{N} \lambda_i p_i : \lambda_i \geq 0 \ \textit{for all } i \textit{ and } \sum_{i=1}^{N} \lambda_i = 1 \right\} \quad (6)$$

Where, CH denotes convex hull. If all the points are not on the same line then convex hull is a polygon. In case of collinear points the convex hull is a straight line. A suitable example of convex hull from [17] is shown in **Figure 3** where it is clearly visible that all the data points are surrounded by a minimum number of points each of which is treated as a SP.

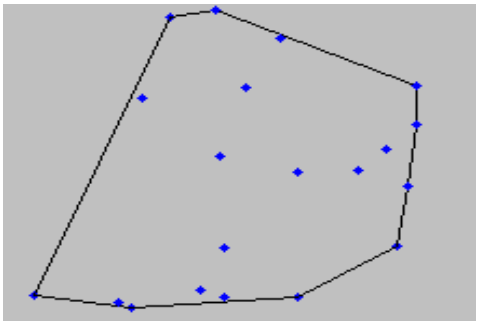


Fig. 2: Convex hull representation

There exists several methods for computing convex hull of which Graham scan, Jarvis march, Divide & conquer, and Quickhull are the most popular ones [18-21]. The properties of all the convex hull algorithms restrict them to only convex objects. They can not be used to produce SPs for a concave object limiting their capability in terms of describing boundary of any object. To resolve this problem an algorithm for generating proper SPs for the curve generation techniques is presented in the following section.

4. THE MODELLING OF FSPP ALGORITHM

This section formally presents an algorithm to generate SPs for curve generation techniques like Bezier curve (BC) namely *finding significant points for parametric curve generation techniques* (FSPP) together with its three constituent algorithms called *rearrange boundary points* (**Algorithm 1**), *generate significant points* (**Algorithm 2**) and *interpolate significant points* (**Algorithm 3**). The first constituent algorithm of the proposed FSPP algorithm is detailed in the following section.

4.1 Rearranging Boundary Points

Since the boundary points scanned from an object may not be arranged in a sequential manner and in this case, it is mandatory to rearrange the boundary points in either clockwise or anti-clockwise to generate SPs using FSPP, an algorithm is presented in this section to rearrange the boundary points. To rearrange the boundary points, a point is taken arbitrarily as the first point (Step 1 of **Algorithm 1**) and considered as the current point (Step 2). It needs to find the Euclidian distances of all points from the current point (Step 3) to determine the next point by applying the concept of both absolute and relative chain codes [22, 23]. To select the next point, firstly the direction of the previous point is considered as shown in **Figure 4**. If a point with unit distance is found in the same direction of the previous point that will be treated as the next point regardless of the directional sequence presented in **Figure 4**. But, if there is no point which has the similar direction as the previous point, then the next point will be selected based on the sequence of the directions i.e. if a point in sequence 1 is found it will be selected as the next point, if not and a point in sequence 2 is found then it will be selected and so on.

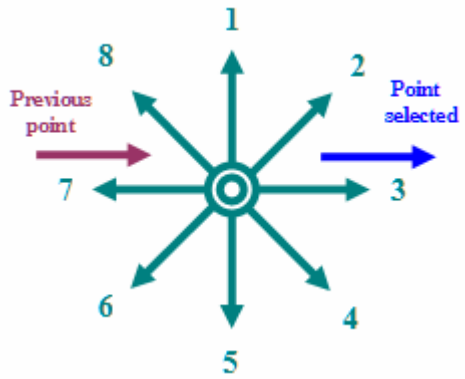


Fig. 4 Directional sequence

For scanning reasons, there may not be found any point with unique distance due to missing boundary points at the time of scanning. In this case the closest point from the current point at minimum distance will be the next point. Afterwards, the current point is stored in a sequential manner along with the previously stored points and the next point is considered as the current point (Step 5). This

process is repeated until a closed boundary is found (Step 6). But if the distance of the first point from the current point (D_f) is smaller or equal to that of the closest point from the current point ($D_{\min}$) i.e. $D_f \le D_{\min}$, then the start or the first point will be taken as the current point and the process is closed as the closed boundary is received.

Algorithm 1: *Rearrange boundary points.*
Precondition: Boundary Points.
Post condition: Arranged Boundary points in B.

1) Choose a boundary point arbitrarily as the first point P_F.
2) Current point $P_C = P_F$.
3) Calculate the distances of all boundary points from P_C.
4) Find the next point P_N as mentioned in Section 4.1.
5) Store P_C and $P_C = P_N$.
6) Repeat step 3 to 5 until a closed boundary is reached.

4.2 Generating Significant Points

The significant point generation technique is detailed in **Algorithm 2**. After rearranging the boundary points using **Algorithm 1,** SPs are generated. To begin the process any one of the boundary points is chosen as the first SP shown in Step 2. The distances of all the boundary points from the first SP are calculated (Step 3) and the boundary point with the largest distance is treated as the second SP (Step 4). All the boundary points between any two consecutive significant points are extracted and then the perpendicular distances of all of these boundary points from the line passing through these two corresponding SPs are calculated in Step 5(a) and then the maximum distance $(\max\_dist)$ is obtained (Step 5(b)). If $(\max\_dist > T_{\max})$, then the respective boundary point will be another SP (Step 5(c)). This process is applicable for all consecutive pair of SPs. For the next process, the SPs are rearranged and the Flag is set to TRUE. If there is no boundary point found left for which $(\max\_dist > T_{\max})$ is true, the process will stop (Step 5 (d)).

Algorithm 2: *Generate significant points.*
Precondition: Arranged set of boundary points and $T_{\max}$.

Post condition: Generated Significant Points C.

1) Flag=TRUE.
2) Choose one point randomly which is the first SP.
3) Calculate distances of the first SP with all shape points.
4) The point with the largest distance will be the second SP.
5) DO WHILE (Flag),
   a) Calculate perpendicular distances of the shape points which lie between two consecutive SPs from the line passing through these two corresponding SPs.
   b) Find maximum distance $\max\_dist$
   c) IF $(\max\_dist > T_{\max})$ THEN
      i) The corresponding shape point will be another SP.
      ii) Do this for every consecutive pair of SPs.
      iii) Rearrange SPs in the similar way of boundary points.
      iv) Flag=TRUE.
   d) ELSE
      IF For every consecutive pair of SPs $(\max\_dist \le T_{\max})$ THEN
      Flag=FALSE.
   e) END IF
6) END DO WHILE

If there is any long straight line boundary, this algorithm will produce SPs with large gap within that line and hence CBC will be unable to generate the expected contour points. To address this issue, it needs to interpolate some new SPs when there is a large gap between two SPs and an algorithm for this purpose is described in the following section.

4.3 Interpolating Significant Points

This section presents an algorithm to interpolate new SPs between a pair of SPs having large gap which is detailed in **Algorithm 3**. After generating SPs using **Algorithm 2**, the distances $(\Delta_{i,i+1})$ between every consecutive pair of SPs are calculated (Step 1 of **Algorithm 3**). To generate a smooth curve, it needs to find out the SPs approximately after the same gap as CBC generates curve points based on the SPs. For this reason, the

average distance (Δ_{avg}) of $\Delta_{i,i+1}$ where $1 \le i \le n$ and n is the number of SPs, is used as a measurement to insert additional SP (Step 2) where:

$$\Delta_{avg} = \frac{1}{n}\sum_{i=1}^{n}\Delta_{i,i+1} \quad (7)$$

If $(\Delta_{i,i+1} > \Delta_{avg})$, then a boundary point having Δ_{avg} distance from i^{th} SP is considered as another SP (Step 3). This process will continue till $(\Delta_{i,i+1} > \Delta_{avg})$ is true for any pair of consecutive SPs (Step 4).

Algorithm 3: *Interpolate significant point.*
Precondition: Significant Points of an object.
Post condition: Final Interpolated Significant Points C.

1) Calculate the distance $(\Delta_{i,i+1})$ between every consecutive pair of SPs.
2) Calculate the average distance (Δ_{avg}) of distances $(\Delta_{i,i+1})$.
3) FOR all i, IF $(\Delta_{i,i+1} > \Delta_{avg})$ THEN consider a boundary point as SP that is Δ_{avg} distance apart from the i^{th} SP.
4) REPEAT Step 3 Until for all i, $\Delta_{i,i+1} \le \Delta_{avg}$.

Based on the above mentioned three constituent algorithms, the formal FSPP algorithm is detailed in the following section.

4.4 The FSPP Algorithm

The FSPP algorithm is detailed in **Algorithm 4**. The boundary points are arranged in a sequence using **Algorithm 1** (Step 1). Then, the SPs are generated using **Algorithm 2** (Step 2) from the rearranged boundary points of the corresponding object. As mentioned in Section 4.2, CBC is unable to generate a smooth curve if there is large gap between two consecutive SPs which motivated to find some new SPs applying **Algorithm 3** (Step 3). Finally, the CBC is used to generate a proper shape descriptor to prove the superiority of the FSPP algorithm (Step 4).

Algorithm 4: *Finding significant points for parametric curve generation techniques* (FSPP).
Precondition: Boundary points of an object.
Post condition: Final Significant Points C.

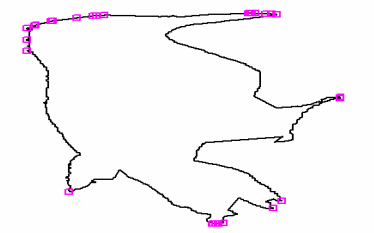
1. Arrange the boundary points using **Algorithm 1.**
2. Generate SPs using **Algorithm 2.**
3. Interpolate new SPs using **Algorithm 3**.
4. Apply CBC with generated SPs to approximate the shape contour.

5. EXPERIMENTAL RESULTS

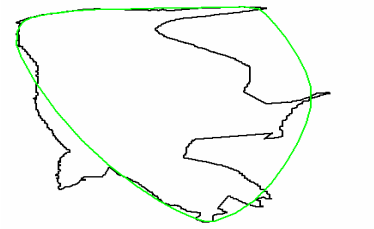
To assess the performance of the proposed FSPP algorithm, all the related algorithms are implemented using Matlab 6.1. The experiments have been conducted using different images having different shapes and orientation. The generated contour produced by CBC for SPs produced by convex hull is compared with that for SPs produced by FSPP. To represent foreground objects, the background of images are manually removed by setting it zero. Any zero valued foreground pixels are replaced by one which does not impact upon the visual perception. For better visual representation, the generated significant points and the contour points are represented using different colours and symbols such as '□' and '-' respectively. The SPs generated by convex hull is represented by magenta '□' and the corresponding curve points produced by CBC is represented by green '-' while these for FSPP are represented by cyan '□' and red '-' with different colour rather than their original gray-scale pixel intensities. The value of perceptual threshold $T_{\max}$ is set to 1 in all experiments.

(a) Original image

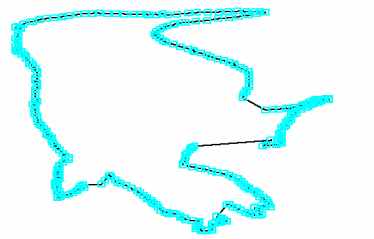
(b) Reference shape



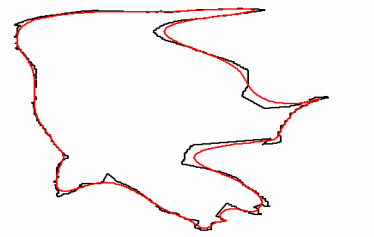
(c) Significant points by convex hull



(d) Generated curve by CBC



(e) Significant points by FSPP

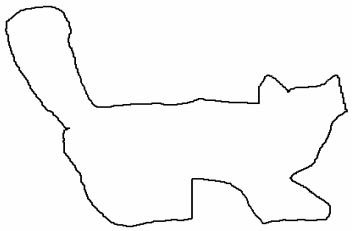


(f) Generated curve by CBC

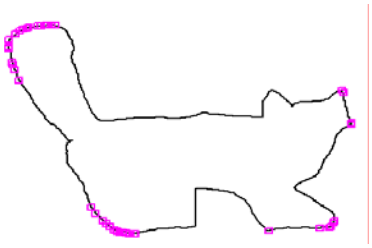
Fig. 5 (a) Original *bird* image, (b) Reference shape, (c) Significant points by convex hull, (d) Generated curve by convex hull, (e) Significant points by FSPP and (f) Generated curve by CBC.

The first set of experimental results relates to the **Figure 5**(a) image having a *bird* object with arbitrary shape. The reference shape or boundary points which are achieved by scanning the boundary is shown in **Figure 5**(b). The SPs generated by convex hull and the generated contour produced by CBC are given in **Figure 5**(c) and **Figure 5**(d) respectively. Similarly, the SPs generated by the proposed FSPP algorithm and the corresponding contour points generated by CBC are shown in **Figure 5**(e) and **Figure 5**(f) respectively. It is clear from **Figures 5**(c)-(d) that convex hull generated ineffective SPs for this arbitrary shaped object and hence CBC produced an erroneous shape with losing its originality. In contrast, the FSPP generated the sufficient number of correct SPs shown in **Figure 5**(e) and hence CBC produced approximately similar contour in comparison with the reference boundary in (**Figure 5**(b)). It proves the superior performance of FSPP algorithm over convex hull.

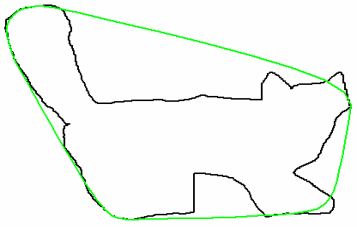
(a) Original image



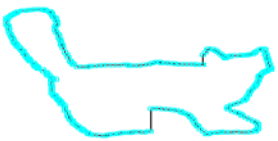
(b) Reference shape



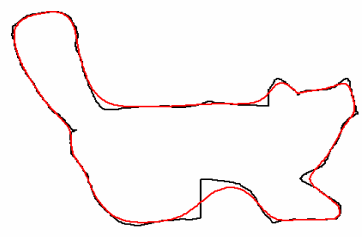
(c) Significant points by convex hull



(d) Generated curve by CBC



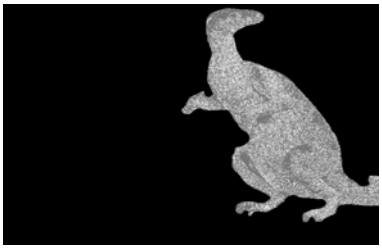
(e) Significant points by FSPP



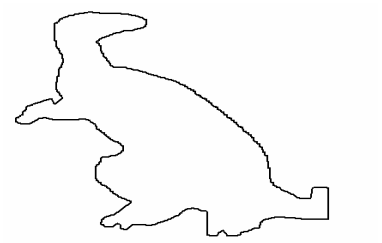
(f) Generated curve by CBC

Fig. 6: (a) Original *cat* image, (b) Reference shape, (c) Significant points by convex hull, (d) Generated curve by CBC, (e) Significant points by FSPP and (f) Generated curve by CBC.

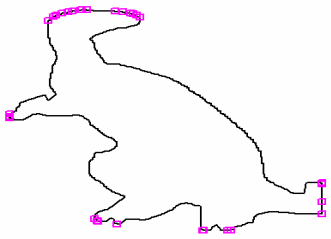
Another sample image analyzed was a *cat* image which has arbitrary shape in **Figure 6**(a) with its corresponding reference boundary points in **Figure 6**(b). Since the convex hull produced inappropriate SPs (**Figure 6**(c)) and as a consequence of this, the CBC generated inaccurate contour points to represent the object properly which is clearly visible in **Figure 6**(d) being impossible to identify the *cat*. On the other hand, the FSPP algorithm produced sufficient and correct SPs (**Figure 6**(e)) and hence CBC generated a more accurate contour of the *cat* shown in **Figure 6**(f). Therefore, this result also proves the superior performance of FSPP over the convex hull.



(a) Original image

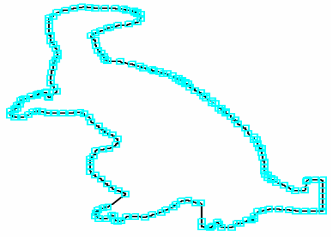


(b) Reference shape

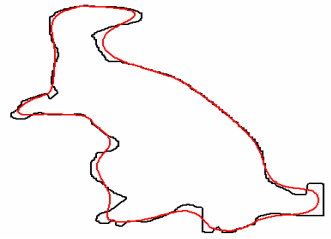


(c) Significant points by convex hull

(d) Generated curve by CBC



(e) Significant points by FSPP

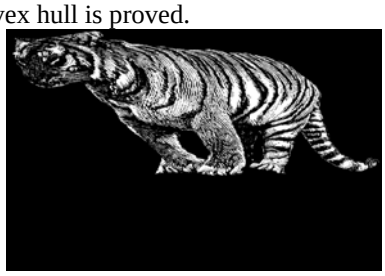


(f) Generated curve by CBC

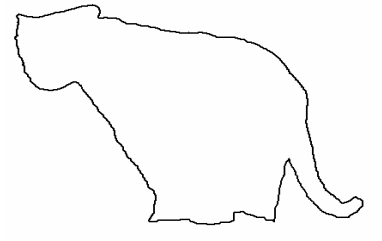
Fig. 7 (a) Original *kangaroo* image, (b) Reference shape, (c) Significant points by convex hull, (d) Generated curve by CBC, (e) Significant points by FSPP and (f) Generated curve by CBC.

Another additional sample image analyzed was *kangaroo* image in **Figure 7**(a) and its corresponding reference boundary points are shown in **Figure 7**(b). The SPs generated by convex hull and the resultant shape produced by CBC are in **Figures 7**(c)-(d)**,** and it is clearly visible that CBC fails to produce proper shape as convex hull

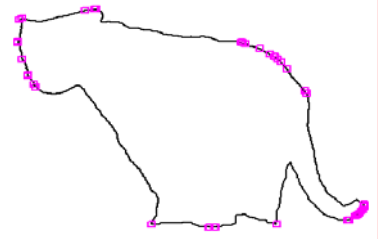
generated insufficient SPs to represent the object. Whereas, the SPs produced by FSPP algorithm and its respective shape produced by CBC in **Figures 7**(e)-(f) precisely approximates the object. Hence, once again the supremacy of FSPP algorithm over the convex hull is proved.



(a) Original image

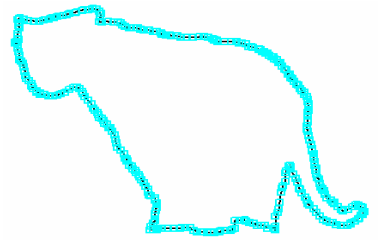


(b) Reference shape



(c) Significant points by convex hull

(d) Generated curve by CBC



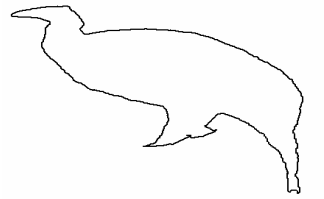
(e) Significant points by FSPP

(f) Generated curve by CBC

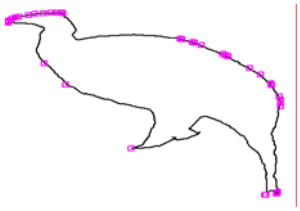
Fig. 8: (a) Original *tiger* image, (b) Reference shape (c) Significant points by convex hull, (d) Generated curve by CBC, (e) Significant points by FSPP and (f) Generated curve by CBC.

One more example of a *tiger* image is given in **Figure 8**(a) while the boundary points of its reference shape are depicted in **Figure 8**(b). The convex hull generated SPs and CBC generated corresponding curve are shown in **Figure 8**(c) and **Figure 8**(d) respectively. The FSPP algorithm generated SPs and CBC generated corresponding curve are shown in **Figure 8**(e) and **Figure 8**(f) respectively. It is found from the **Figures 8**(c)-(f) that the FSPP algorithm produced better SPs than convex hull and hence CBC generated better shape for tiger object. As a result, the dominance of FSPP algorithm over convex hull is once again confirmed.

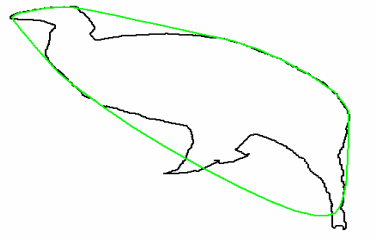
(a) Original image



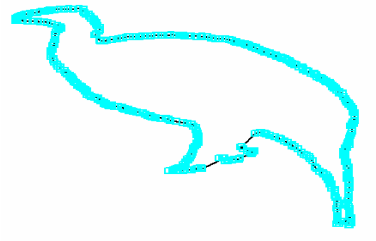
(b) Reference shape



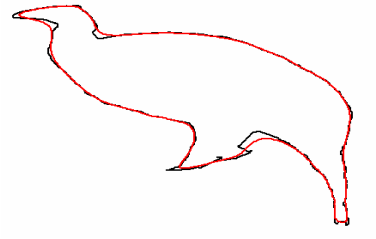
(c) Significant points by convex hull



(d) Generated curve by CBC



(e) Significant points by FSPP



(f) Generated curve by CBC

Fig. 9 (a) Original *peacock* image, (b) Reference shape, (c) Significant points by convex hull, (d) Generated curve by CBC, (e) Significant points by FSPP and (f) Generated curve by CBC.

The final example of the *peacock* image is shown in **Figure 9**(a) while its reference shape descriptor in **Figure 9**(b). As CBC produced approximately accurate contour points of *kangaroo* with the SPs generated by the FSPP algorithm shown in **Figures 9**(e)-(f) in comparison with the contour points produced by CBC with SPs generated by convex hull shown in **Figures 9**(c)-(d), this also proves that the FSPP algorithm outperforms the convex hull to generate proper significant points for any arbitrary shaped object.

To assess the robustness of the FSPP algorithm, the SPs are generated for 420 images having different shaped objects while the FSPP algorithm always generated better SPs than convex hull for any shaped objects for all images and hence CBC always produced better or equal shape descriptor with the SPs produced by FSPP than that of produced by convex hull.

Finally, to justify the threshold value, the value of T_{max} was varied from 0 to 5. It was clearly seen that, when T_{max}=0, the FSPP produces maximum amount of SPs with better curve representation by CBC. While increasing the T_{max} value from 0 to 5 will reduce the number of SPs. For most of the cases, FSPP generates approximately similar curves while for some other cases it produces curves with insignificant shape distortion that is negligible. As increasing the number of SPs increases the time complexity, this motivated us to consider the T_{max} value as 1 which reduces the number of generated SPs with shape produced by CBC with acceptable shape distortion using these SPs. But it may be any values according to desired level of SPs and shape approximation.

6. CONCLUSION

This paper has presented a new significant point generation technique namely *finding significant points for parametric curve generation techniques* (FSPP) which is able to generate proper SPs for the curve generation techniques. Using these SPs, it is possible to approximate appropriate shape descriptor of arbitrarily shaped objects using composite Bezier curve. The experimental results have shown that the FSPP algorithm has outperformed the convex hull algorithm in finding appropriate SPs and thereby can be used for better curve generation. As consequences, this increases the use of Bezier curve techniques in the field of image processing.

Corresponding Author

Md. Riazur Rahman
Assistant Professor
Dept of Computer Science & Engineering,
Daffodil International University, Bangladesh.

REFERENCES

[1] I. Gath, and A. B. Geva, Unsupervised Optimal Fuzzy Clustering, *International Journal of Pattern Analysis and Machine Intelligence, 2(7)*, pp.773-781, 1989.

[2] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithm* (New York: Plenum Press, 1981).

[3] M. Ameer Ali, G.C. Karmakar, and L.S. Dooley, Fuzzy image segmentation of generic shaped clusters, *IEEE International Conference on Image Processing*, 2005.

[4] M. Sarfraz, M.R. Asim, and A. Masood, Capturing outlines using cubic Bezier curves, *IEEE International Conference on Information and Communication Technologies*: *From Theory to Applications*, 2004.

[5] S. Pal, P.K. Biswas, and A. Abraham, Face recognition using interpolated Bezier curve based representation, *International Conference on Information Technology: Coding and Computing*, 2004.

[6] I. Shdaifat, I. Grigat, and D. Langmann, Active shape lip modelling, *IEEE International Conference on Image Processing*, 2003.

[7] H. Lin, L. Liu, and G. Wang, Boundary evaluation for interval Bezier curve. *Computer-Aided Design, 34(9)*, p. 637-646, 2002.

[8] M. Sarfraz and M.A. Khan, Automatic outline capture of Arabic fonts. *Information Sciences, 140(3-4)*, p. 269-281, 2002.

[9] H.M. Yang, J.J. Lu, and H.J. Lee, A Bezier curve-based approach to shape description for Chinese calligraphy characters, *International Conference on Document Analysis and Recognition*, 2001.

[10] S.H. Francis, *Computer Graphics* (New Jersey: Prentice Hall, 1994).

[11] D. Hearn and M.P. Baker, *Computer Graphics* (New Jersey: Prentice Hall, 1994).

[12] F.A. Sohel, L.S. Dooley, and G.C. Karmakar, A dynamic Bezier curve model, *IEEE International Conference on Image Processing*, 2005.

[13] http://en.wikipedia.org/wiki/Bezier_curve, Last date of access: 2/6/2007.

[14] S. R. BUSS, *3-D Computer Graphics* (New York: Cambridge University Press, 2003).

[15] P.F. Preparata and I.M. Shamos, *Computational geometry* (New York: Springer Verlag Inc, 1985).

[16] http://mathworld.wolfram.com/ConvexHull.html, Last date of access: 18/05/07.

[17] http://www.cs.princeton.edu/~ah/alg_anim/version1/ConvexHull.html, Last date of access: 17/05/2007.

[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms* (Second Edition, MIT Press and McGraw-Hill, 2001).

[19] P.F. Preparata and S.J. Hong, Convex Hulls of Finite Sets of Points in Two and Three Dimensions, *Commun.ACM*, vol. 20, no. 2, pp. 87–93, 1977.

[20] M. de Berg; M. van Kreveld, Mark Overmars and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications* (Springer, 2000).

[21] C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa, The Quickhull algorithm for convex hulls, *ACM Trans. on Mathematical Software*, 22(4):469-483, Dec 1996.

[22] http://www.mind.ilstu.edu/curriculum/chain_codes_intro/chain_codes_intro.php, Last date of access: 29/05/2007.

[23] R. C. Gonzales and R. E. Woods, *Digital Image Processing* (New Jersey: Prentice Hall, 2002)