Journal of Computer Science & Computational Mathematics, Volume 6, Issue 2, June 2016
DOI: 10.20967/jcscm.2016.02.001

Preservation of Sharp Edges on 2D Drawing Using B-Spline Curve

Nur Soffiah Sahubar Ali[*], Ahmad Ramli, Nur Shafiqah Daud

School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia,
[*]Corresponding author email: nsoffiah_91@yahoo.com.my

Abstract: B-spline is widely used in computer graphics in order to construct a smooth curve. In this paper, we discuss about the detection of sharp points on a B-spline curve and the preservation of the sharpness at the corner of the edges as the degree of the curve increases. We have made the removal process for the data that use a huge number of points and then marked the sharp edge points which will be used to calculate the Euclidean distance between the control points. The formulation of automatic knot vector generation by manual marking of points close to the sharp edges is implemented to obtain an algorithm that is capable of detecting the sharp points. The smoothness of the curve is based on the increase in the degree or order of the curve. Thus, to preserve the sharpness of the edges, knot vector repetition is applied corresponding to the sharp points. In addition, knot vectors play an important role in preserving the sharpness of the edges because the changes in knot vectors will produce various shapes of the B-spline curve.

Keywords: hand drawing, sharp edge preservation, knot vector repetition, B-spline curve.

1. Introduction

In Computer Aided Geometric Design (CAGD), various applications have been expanded rapidly into the broad aspects in our real life which is from pharmaceutical design to animation. Rockwood and Chambers [1] stated that visualization using computer reduces the design cycle by easing the process of design modification and tool production. In CAGD, the creation of curves and surfaces is described as curve and surface modeling. Designers create and refine their ideas to produce complex results using CAGD tools with elaborate user interfaces (an interface between a user and a computer system). A curve is defined with a series of polynomials. Curves with parametric equation form are known as parametric curves. A parametric curve is called a polynomial curve if the parametric curve has a polynomial parameterization. Meanwhile, parametric curve is also a standard forms to describe curves and surfaces in this field. The rather common types of parametric curve are namely Bezier and B-spline curves.

B-spline curve is more complex as compared to a Bezier one as it requires more information (such as order or degree of the curve and knot vector). Since a curve is defined with a series of polynomials, the number of polynomials needed to define a curve depends on the number of control points and the order of the curve. They are called “knots” because the values connect the polynomials. The collection of knots for a particular curve is called the knot vector.

In term of flexibility, B-spline curves produce more control than Bezier curves. We can use a lower degree curve but still maintain a large number of control points. The shape of the curve can be altered by using different knot vector configurations.

2. Background Study

Here, we discuss some relevant definitions that are related to B-spline. The terminology of the B-spline formula will be introduced as well as the basic definition and data structures will be detailed. Our understanding is then demonstrated on a programming language, which is Mathematica program in this problem.

B-spline function is a generalization of the Bezier curve. A spline curve is a sequence of curve segments that are connected together to form a single continuous curve. For example, in a piecewise collection of a Bezier curve, when they are connected end to end, we can call it as B-spline curve. In a B-spline curve, each control point is associated with a basis function. For further details, one may refer to Bertka [2] and Salomon [3]. B-spline curve has more advantages and more control flexibility than Bezier curve. Saito and Serikawa [4] discussed a method for automatic generation of B-Spline curve for the representation of handwritten characters. They proposed this new method which can be applied with handwritten characters specifically to aid senior citizens and children who may have difficulties in communication using e-mail. They also introduced Polytope method, where the passing point of B-Spline curve is quickly obtained.

B-spline curves are flexible and we can use lower degree curves to maintain a powerful and useful approach in order to generate a smooth curve. B-spline curves of polynomial order k are the unique functions that are globally in C^{k-2} and a piecewise polynomial of degree $(k-1)$. A B-spline curve is infinitely differentiable, except possibly at knot positions. This can be expressed as C^1 continuity which mean that the first derivative are continuous whereas C^2 continuity means the second derivative are continuous. Generally, B-spline curve is defined as

$$B(t)=\sum_{i=0}^{n} d_i N_{i,k}(t)$$

for $t_{k-1} \le t < t_{n+1}$ where, $\{d_0, d_1, ..., d_n\}$ are control points and $n+1$ is the number of control points. The value of n must be larger or equal to k.

The knot sequence is a set of non-decreasing real number and the basis function is obtained based on the differences between the knots. The specific values of the knots are less important as the knot spacing will determine the shape of the basis function. (Read Farin [5] and Sederberg *et al.* [6]). There are several types of B-splines. In the uniform (also called periodic) B-spline, the knot values are uniformly spaced with the same shape. In an open uniform B-spline, *k* repeated and equal value knots are found at each ends and we can use them to describe a closed curve that interpolate the first and last control points. Non-uniform B-spline allows any spacing of the knots, including multiple knots which adjust the continuity of the curve at the knot values. The changes of the configuration of knot vectors on B-spline curve are discussed by Hilman [7]. Awad and Man [8] proposed a high performance in efficiency and implementation on edge detection techniques in images. There are many edge detection techniques in literature and Hassanpour *et al.* [9] have made a comparison among several techniques for edge detection in image processing. The literature analysis was carried out using a drawing tool. A user will make use of the input device, which is the mouse to create a drawing. A series of points will be collected when each mark is made. Simply, these points can be the input data for the construction of the curve. However, as a mark does not define corner or edges information in the data, a B-spline curve will be smooth. Altering B-spline properties such as control point repetition or knot vector repetition may preserve the sharp edges or corner.

In this paper, our main contribution is to generate an aesthetically nice looking curve with sharp edge preservation from data set produced by hand drawing. Firstly, we will go through the process of removing redundant data points or dense data. Secondly, we will mark a point which is closed to the sharp edge and automatically generate the knot vectors to achieve our targeted results.

3. Removal of points and detection of sharp edges

In this section, we discuss how the removal of points is done when involving huge number of points and since a particular curve has more points, the effect of moving any one of them is smaller. Then, we create an algorithm to detect the sharp edges.

3.1 Removal of Points

In using the drawing tool to draw a specific figure having a better edge will require too many points to be involved. Particularly, we remove some points and thus reduce the number of points involved and this can enlighten our task to work out the result and to get a better understanding. For the data that consist less points to be involved, the removal of points would not be required.

In our context, we have drawn, as in Figure 1.

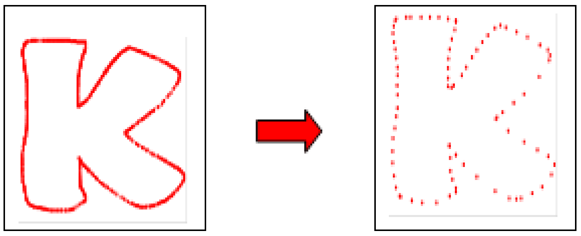

Figure 1. Before and after removal of points for the letter K

After removing some of the points from the exact data which involve many points, this can make our task easier. The next process of obtaining the sharp points from the exact data and merging the points to the newly removed point data can be done to achieve our aim of this dissertation approach.

3.2 Detection of Sharp edges

In this section, proceeding with the images drawn earlier using the drawing tool and applying the necessary removal process, now we want to fit the specific corners which are the sharp edges. In detecting the sharp point, firstly, we manually mark the sharp edges at the corners that we want and then the merging of the sharp points are conducted in the next step as follows. As our experimental setting is done on Mathematica, marking a sharp point produces a coordinate which will be used later.

Referring to Figure 2, we mark the sharp points that are close to the sharp edges (shown in blue point) for letter K. This is done by the user to calculate the Euclidean distance between the points that are marked in “blue” and all the respective points to determine the points that are going to be declared as the sharp points. In this case, the letter K consists of 11 sharp edges and 11 blue points that are marked. These marked points will be used in the next section in order to get the respective sharp points.

For illustration purpose, we intentionally marked our sharp points in blue somewhere close but not exactly on the data points to avoid visual overlapping.

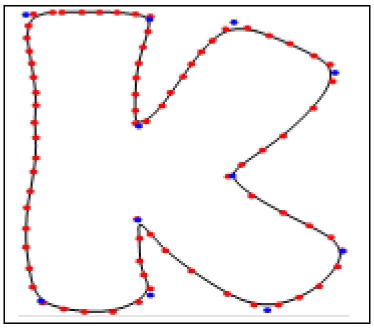

Figure 2. The position of edges marked manually by the user in blue dots and removed points in red

3.2.1 Algorithm for finding the sharp point

After obtaining the drawing and marking the edges, we calculate the Euclidean distance between the marked points (in blue) with all of the control points (red). The distance that is too small close to the marked point (in blue) will be declared as the sharp point. We have run an algorithm and tested it with another drawing. This algorithm is developed to find the sharp points which will be used to construct the respective of B-spline curve later. For the case of less control points involved, we can find the sharp points directly using the data without removing the control points.

```
Algorithm: Sharp point marking

Input: Control points, $d_i$, the point that we marked at the
respective sharp edges, $d_j$, and distance, $d$
Output: The points where the sharpness occur, $d_p$
Number of control points = $n+1$
Number of marked respective sharp edges = $m$

1. For every marked sharp point, $d_j$,
        $j = 1, j \le m, j++$, where $d$ = 0.50
2. For every point $d_i$, $i = 1, i \le n+1, i++$,
    2.1 If the Euclidean distance between $d_j$ and $d_i$ is
        less than $d$, then $d_p = i$, $j++$ and we will get to
        join all the sharp point, $d_p$
```

After applying the coding, the sharp points that are obtained for letter K from the eleven respective edges are the following points which are the 7th, 15th, 24th, 30th, 36th, 42nd, 47th, 53rd, 59th, 65th and 83rd.

As an additional note, if the points are dense (such as in Figure 1), the removal of points is required. In this case, we obtained the sharp points from the initial data before the removal of control points. Then, we merged those sharp points into the removed points before proceeding to the next step.

4. Knot Insertion

In a B-spline curve, knot insertion is one of the most important algorithms since many other useful algorithms are based on knot insertion. After obtaining the sharp points, we use knot insertion algorithm to construct the desired curve. As we know, the control points, order of the curve and knot vector are all related to each other in the process of obtaining the total number of knot vectors of the curve.

Let $u_1 = \{u_0, ..., u_i\}$ and the new vector is $u_2 = \{u_0, ..., u_{i+k}\}$, we can observe that u_1 is a subset of u_2. The repetition of knot values causes the B-spline to start and end at its corresponding control points. We can interpret that at every subinterval $[u_i, u_{i+1})$ of knots corresponds to its segment of the curve. This segment reduces to a point when $u_i = u_{i+1}$. This means that for each repetition of a knot value, it decreases the continuity at joint point by one.

4.1 Construction of the 2D images using uniform knot vector

In this context here, we increase the degree up to quartic to observe the change in the shape of the curve using uniform knot vector. We generate the B-spline curve with comparison of three different degrees to see the changes and we made use of the 2D images that has been created and discussed earlier as an example to demonstrate our understanding. The curves obtained are shown in Figure 3.

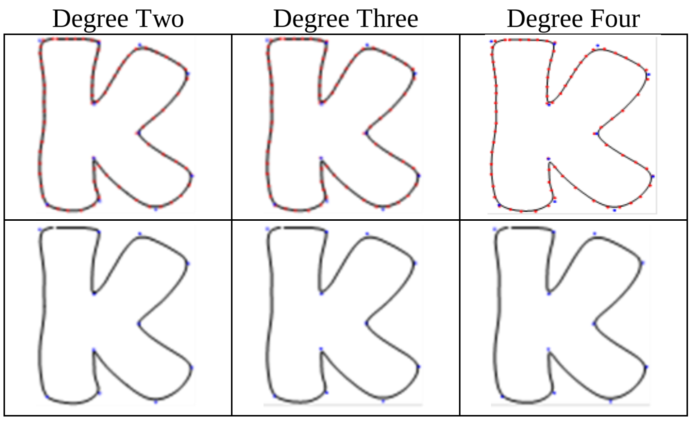

Figure 3. Letter K with degree two, degree three and degree four

We can see that, the shape of the curve becomes smoother and there is no sharpness at the corners or edges due to the curve generation using a higher degree and the uniform knot vectors. Therefore, in the next section, we generate knot vectors repetition and create the B-spline curve to see the changes in the shape of the curve.

4.2 Construction of the 2D images using knot vector repetition

We have observed that the sharp points at the particular edges for the letter K and we constructed the curve using uniform knot vectors. Our next aim is to preserve the sharpness at the edges by increasing the repetition of knot vectors at the position of the sharp points in the construction of knot vectors for the curve. Now, from the control points obtained, we interpret the data of the image drawn as follows.

The letter K consists of 85 control points, and the position of sharp points are the 7th, 15th, 24th, 30th, 36th, 42nd, 47th, 53rd, 59th, 65th and 83rd. Next, we generate the knot vectors to see the differences of the curve. The differences of the curve are as follows when there is an increase in degree 2 up to degree 4.

For the purpose of discussion, these knot generation are done manually here and will later be obtained using the algorithm in section 4.3. As the degree of the curve increases, manual knot generation will be tedious and the aid of algorithm 4.3 that was further modified from Daud [10] will be required. The specification of these three different degree are done accordingly. Note that the knot vector corresponding to the sharp points has been bold.

a) For the second degree $k = 3$, insert three repeated knots sequentially

$$t = \{t_0, t_1, t_2, ..., t_6, t_7, t_8, ..., t_{14}, t_{15}, t_{16}, ..., t_{23}, t_{24}, t_{25}, ..., t_{29},$$
$$t_{30}, t_{31}, ..., t_{35}, t_{36}, t_{37}, ..., t_{41}, t_{42}, t_{43}, ..., t_{46}, t_{47}, t_{48}, ..., t_{52}, t_{53},$$
$$t_{54}, ..., t_{58}, t_{59}, t_{60}, ..., t_{64}, t_{65}, t_{66}, ..., t_{82}, t_{83}, t_{84}, t_{85}, t_{86}, t_{87}\}$$

$$= \{0,0,0,1,2,3,4,4,4,5,6,7,8,9,10,10,10,11,12,13,$$
$$14,15,16,17,17,17,18,19,20,21,21,21,22,23,24,25,$$
$$25,25,26,27,28,29,29,29,30,31,32,32,32,33,34,$$
$$35,36,36,36,37,38,39,40,40,40,41,42,43,44,44,$$
$$44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,$$
$$59,60,60,60,61,61,61\}$$

b) For the third degree $k = 4$ insert the four same knots sequentially. The step is the same as done earlier, but the insertion of the knots is increased by one.

$$t = \{t_0, t_1, t_2, t_3, ..., t_6, t_7, t_8, t_9, ..., t_{14}, t_{15}, t_{16}, t_{17}, ...,$$
$$t_{23}, t_{24}, t_{25}, t_{26}, ..., t_{29}, t_{30}, t_{31}, t_{32}, ..., t_{35}, t_{36}, t_{37}, t_{38}, ..$$
$$., t_{41}, t_{42}, t_{43}, t_{44}, ..., t_{46}, t_{47}, t_{48}, t_{49}, ..., t_{52}, t_{53}, t_{54}, t_{55}$$
$$, ..., t_{58}, t_{59}, t_{60}, t_{61}, ..., t_{64}, t_{65}, t_{66}, t_{67}, ..., t_{82}, t_{83}, t_{84}, t_{85}$$
$$, t_{86}, t_{87}, t_{88}\}$$

$$= \{0,0,0,0,1,2,3,3,3,3,4,5,6,7,8,8,8,8,9,10,11,12,13,$$
$$14,14,14,14,15,16,17,17,17,17,18,19,20,20,20,20,21,$$
$$22,23,23,23,23,24,25,25,25,25,26,27,28,28,28,28,$$
$$29,30,31,31,31,31,32,33,34,34,34,34,35,36,37,38,39,$$
$$40,41,42,43,44,45,46,47,48,49,49,49,50,50,50,50\}$$

c) For the fourth degree (k = 5), insert the five same knots sequentially

For the fourth degree and so on we apply the same concept to obtain the knot vector and those knot vectors are used to create the 2D images and are illustrated in Figures 4, 5 and 6. The blue dots represent points marked by the user.

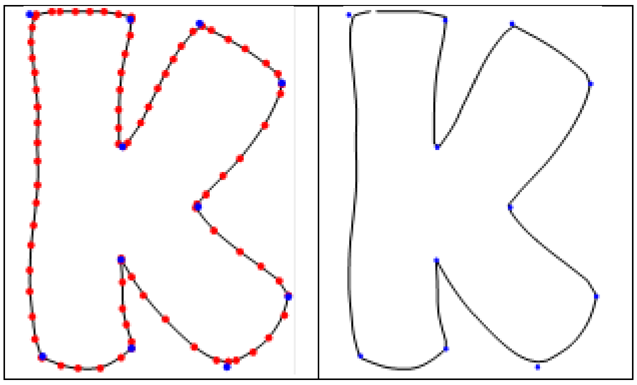
Figure 4.B-spline curve with degree 2.

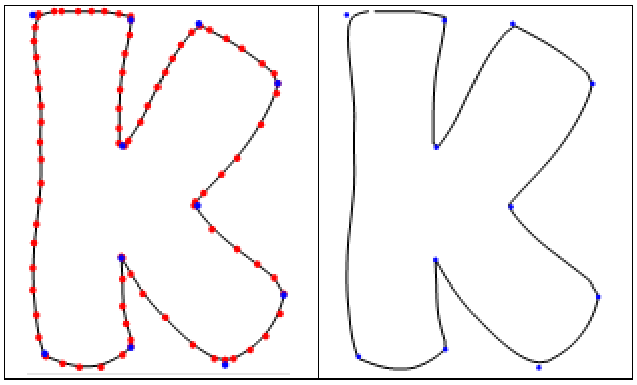
Figure 5.B-spline curve with degree 3

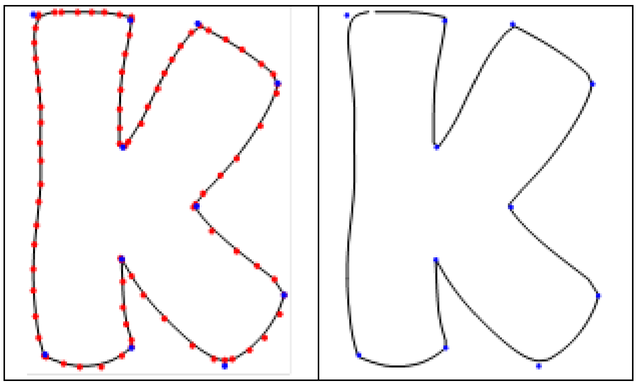
Figure 6.B-spline curve with degree 4

As we increase the order of the curve, we can see that the sharp edges are preserved well in all three cases. This is because we have enough number of control points to allow the repetition of knot vectors. However, if the number of control points less than the number of possible knot vectors repetition and the sharp edges are close to each other, the sharpness will not be preserved at certain edges of the curve. This is due to the earlier removal of a few important control points near the edges. The corresponding knots to the sharp points may be consecutive. Therefore, the repetition of the knots could not be done. In order to smooth the curve by increasing the order *k*, one could take more control points so that the knot repetition can be done. However, if we increase the density globally, the smoothness of the curve may be compromised. Increasing the density only for the points around the sharp edge may be a good approach but is not discussed in this paper.

Knots are added after identifying the control points. It is true that there is nothing new for that particular statement. However, this paper look as a whole for the case of hand drawing where a user click a point not necessarily on the exact sharp point, but close enough to the sharp edge. We also observe the difficulty of retaining the sharp edge in hand drawing compared to a curve which is define by some control points.

After we have constructed the figures with sharp edges by using knot vectors in the previous section, we run the algorithm in section 4.3. As the generated knot vectors enable us to produce the same knot vectors to construct the

B-spline curve obtained in the previous section, the algorithm is modified with further enhancement to solve the problems related to different drawings.

4.3 Algorithm for finding knot vectors

Algorithm: Knot vector generation

Input: Control points, d_i, order, k, and d_p is a point close to the sharp edge.
Output: Knot vector that can be used to construct B-Spline with sharp edges

```
$n$ = Length [control points] -1,
$m = n + k + 1,\ v = 1$

1. For every point, $d_i$, $i = 0 : k$, the first $k$ knot vector = 0
        1.1 If $d_p \le k+1$, so $p = p+1, p++$
2. For every point $d_i$, $i = k+1 : m-k$
        2.1 If $d_i$ is not equal to $d_p$, so the knot vector
            $t_i = v, v++$
        2.2 Otherwise, For $j = 0 : k-1$, knot vector,
            $t_{i+j} = v$ and will get the multiple knot vector
3. For $i = m-(k-1) : m$, we will get the last $k$ knot vector
   $= v$
```

5. Experimental results

Graphical objects can be produced by using the control points and knot vectors of the B-spline curve. One of the applications of B-spline curves is in the production of a 2D drawing. In this paper, we create an example of 2D drawing using the drawing tools shown in Figure 7.

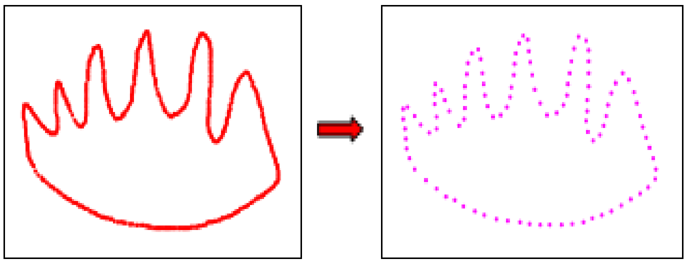

Figure 7. Before and after the removal process of control points for crown

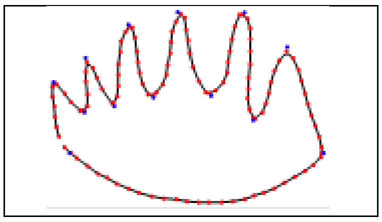

Figure 8. The created crown drawing

Figure 8 is obtained using the uniform knot vectors. In order to preserve the sharpness of the curve at the edges, we apply all the steps of knot repetition that we had learnt earlier. We test our algorithm created to detect the sharp edges as well as the algorithm to find the knot vectors (repeated knot vectors) to make repetitions at the position of the sharp points. Firstly, we obtain the sharp point and from there we get the knot vectors. We make a comparison of the increment in degree of two and four.

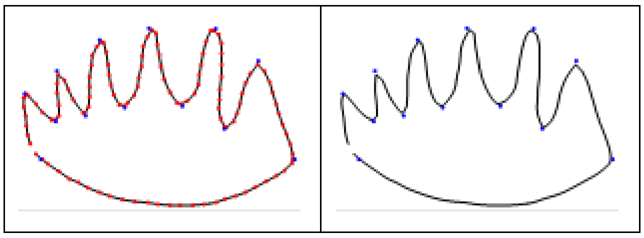

Figure 9. The created B-spline curve using degree 2

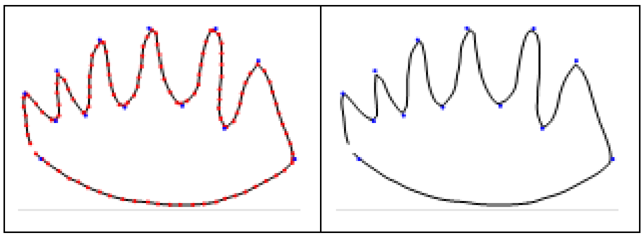

Figure 10. The created B-spline curve using degree 4

From the obtained results of 2D image, we could see that the sharpness is achieved and preserved at the edges. We apply the technique and method learnt earlier to produce a B-spline curve. Repetition of knot vectors at the sharp points is done to construct the sharp edges. The desired sharpness is well obtained.

6. Conclusion

We have proposed new algorithm to preserve the sharp edges of the curve by user-marking as well as to preserve the sharpness as the degree of curve increases. As we know, B-spline is one of the way to create a smooth curve. We generate the repeated knot vectors automatically to preserve sharp edges.

From this paper, some experiments have been done to test our application on the created 2D drawing such as letter K and the crown. Those drawings are created using the drawing tool application and thus involve many control points in order to get a pleasant or nice image. The number of sharp points depends on the number of edges or the corner of the curve.

Knots repetition at the detected sharp edges plays an important role as the degree of the curve increases. Repeated knots insertion sequentially at the sharp points makes stronger attraction towards the control points. Other than that, continuity of the resulting curve reduces the smoothness. The repetition of knot starts with the points where the sharp edges happen.

Finally, further studies are needed to gain a better understanding of the effect of repetition of knot vectors when it involves more complex data points such as non-uniform distribution of the point data and the issues related to sparseness of the data.

Acknowledgment

The authors would like to express their gratitude to the School of Mathematical Sciences, Universiti Sains Malaysia for the financial support from the Ministry of Higher Education Fundamental Research Grant Scheme (FRGS) grant no.203.PMATHS.6711433. The first author also gratefully acknowledges the generous financial support by the Ministry of Education Malaysia under MyMaster Scholarship.

References

[1] A. Rockwood, P. Chambers, "Interactive Curves and Surfaces", *A Mutimedia Tutorial on CAGD, Part 1,* 1996.

[2] B. T. Bertka, "An introduction to Bézier curves, B-Splines, and Tensor Product Surfaces with History and Applications", *University of California Santa Cruz,* May 30th, 2008.

[3] D. Salomon, "Curves and Surfaces for Computer Graphics", *New York, Springer*, 2006.

[4] M. Saito, S. Serikawa, "A Method for Automatic Generation of B-Spline Curve for the Representation of Handwritten Characters", *ISIS Proceedings Of The 8th Symposium On Advanced Intellingent Systems,* pp. 109–112, 2009.

[5] G. Farin, "Curves and Surfaces for Computer-Aided Geometric Design", *A Practical Guide,* Elsevier, 2014.

[6] T.W. Sederberg, J. Zheng, X. Song, "Knot Intervals and Multi-Degree Splines", *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 455-468, 2003.

[7] N. Hilman, "Effect of Knot Vector on B-Spline Curve and Surface", *MSc Dissertation*, Universiti Sains Malaysia, 2013.

[8] A. S. Awad, H. Man, "An edge detection technique in images", in *37th IEEE Applied Imagery Pattern Recognition Workshop,* pp. 1–5, 2008.

[9] H. Hassanpour, E. Nadernejad, H. Miar, "Image enhancement using diffusion equations", *Int. Symp. on Signal Processing and its Applications (ISSPA), Sharjah*, UAE, 2007.

[10] N. S. Daud, "Construction of sharp edges using B-Spline Curve", *MSc Dissertation,* Universiti Sains Malaysia, 2013.