World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:6, No:8, 2012

Beta-spline Surface Fitting to Multi-slice Images

Normi Abdul Hadi, Arsmah Ibrahim, Fatimah Yahya, Jamaludin Md. Ali

***Abstract*—**Beta-spline is built on G^2 continuity which guarantees smoothness of generated curves and surfaces using it. This curve is preferred to be used in object design rather than reconstruction. This study however, employs the Beta-spline in reconstructing a 3-dimensional G^2 image of the Stanford Rabbit. The original data consists of multi-slice binary images of the rabbit. The result is then compared with related works using other techniques.

***Keywords*—**Beta-spline, multi-slice image, rectangular surface, 3D reconstruction

I. Introduction

RECONSTRUCTION of a real object using computer software is of great importance as this enabled the storage of its data structure for future purposes. The reconstructed image can be used in studying the object's insides without breaking it. The advantages of 3D reconstruction algorithm over 3D reconstruction equipment like templates and mould are that the developed algorithm can be applied in any type of computer system and economize the cost of expenses, space and computer storage too. A lot of 3D reconstruction techniques have been developed, each having its own capability and limitation. Two major techniques are surface and volume rendering which focus on geometrical and interior side respectively. The chosen reconstruction technique depends on the type of image used.

Basically, a 3D image reconstruction is based on video, stereo and image itself. For image based reconstruction, the technique will be different between single type images and multi-image. Multi-image has two different categories, whether images are taken from different angles of the object such as axial, sagittal and coronal, or the object is scanned to produce multi-slice images as Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI) scanners.

Since a lot of techniques have been developed, user has to decide the best technique to be used. The performance of some of these techniques is evaluated based on several criteria. A main criterion is the smoothness of generated image besides the processing time and the accurateness. Image smoothness also shows the capability of the technique in handling noisy and large datasets. This criterion is highly dependent on the continuity of connection between images.

Continuity is a big issue in image reconstruction. Refinement of fitted curves and surfaces to fulfill the continuity requirement always delay the processing time. Continuity also makes complex image such as those containing branching contours, hard to handle. This is because in branching , new contour slice usually has to be inserted and this affects the distance between contour slices and the continuity.

N. A. Hadi, A. Ibrahim, F. Yahya are with the Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia (phone: 603-55435356, 5357, 5359; fax: 603-55435501; e-mail: normi, arsmah, fatima@tmsk.uitm.edu.my).

J. M. Ali is with Science University of Malaysia, 11800 Minden, Pulau Pinang, Malaysia (e-mail: jamaluma@cs.usm.my).

This study is supported by Excellent Fund, UiTM.

The required degree of continuity depends on the need of reconstruction. In designing a car for example, the continuity has to be until degree two to avoid sharp edges, holes and leakages. If the purpose is for visualization only, degree zero and one are sufficient. Lower degree is chosen because of the less work and time involved compared to one of higher degree although it is better.

Commonly used curves for reconstruction are Bezier and B-spline whether for triangular or rectangular mesh surface. Both curves have different ways in approximating the data points since the arrangement of the knots is different. Bezier curve consists of uniform knots and reconstruction using this curve is based on the corner points as the curve endpoints. Reconstructions using this curve can be referred in[1-2]. For B-spline, usually non-uniform knots are selected. The technique is called as squared distance minimization. The image is approximated by setting an initial polygon first. Then, the polygon is modified to approach the image by minimizing the error. Further reading of this technique can be found in[3-4].

Beta-spline is the family of B-spline that offers high degree of continuity. This curve is built on G^2 continuity properties which promise the reconstructed 2D and 3D images to have second degree of geometric continuity. Since the reconstruction using this curve requires a large number of curve segments, the approximation error is really small. Therefore, reconstruction process is simplified since no curve refinement is needed to satisfy the continuity and error threshold. This curve can give the same continuity although the distance between points and slices are changed. Therefore, 3D reconstruction using rhis curve may simplify some processes such as branching contour fitting and curve refinement.

This study uses Beta-spline in reconstructing a 3D image of the Stanford Bunny. This image is selected since it is widely used as an example in 3D reconstruction. The real object is a bunny statue which is scanned at Stanford University sometime in 1993 to 1994. The data then was publicized in the Stanford University website[5] and open for researchers to be used. Total image for Stanford bunny consists of 258 slices. This study uses only alternate slices to save on data storage and time. The data used also have been converted into binary form (black and white).

The methodology of the study is discussed in Section II which includes boundary tracing, corner detection, curve fitting and surface fitting. Section III shows and discusses the results obtained and make comparisons with other results from different studies. Finally, the conclusion is in Section IV.

II. Methodology

A. Boundary Tracing

In order to process and manipulate the image, the data of the image have to be extracted. Image data includes the boundary, and corner points of objects in the image.

Boundary tracing is a process to convert the embedded image from image format (.jpeg, .bmp) to a set of points which is the pixel coordinates of the boundary. Since the image is in binary form, the boundary is traced among the white pixels as shown in Fig.1 and Fig. 2.

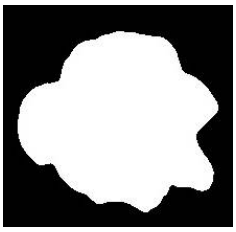

Fig. 1 An embedded image

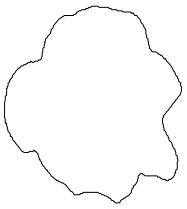

Fig. 2 A traced boundary

Fig. 1 and 2 shows the original embedded image in binary form, and the traced boundary respectively. Image in binary form can simplify the tracing process since it contains only two intensity values: black and white. Region of interest (ROI) is separated from another region by setting it to be white pixels region. The white pixel is traced as boundary point if at least one of its four neighbours (left, right, up, down) is black[6]. The coordinates of the pixel is also recorded. For gray-scale image, the image has to be converted into binary form before the tracing process by setting a threshold, T [7-8]. Let $L(x,y)$ as the original intensity of pixel with (x,y) coordinate, and $g(x,y)$ is the intensity after conversion.

$$g(x,y)=\begin{cases}1 \text{ for } L(x,y)\geq T\\ 0 \text{ for } L(x,y)<T\end{cases} \quad (1)$$

After the boundary has been traced, the boundary points are rearranged and the points are kept as a set of positive integers using chain code[9]. An image contour may contain hundreds of boundary points which require lots of computer storage, and delay processing time. Chain code is a simple way to store boundary points whilst minimizing space. An example of chain code is in Fig. 3 and 4.

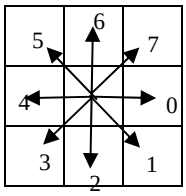

Fig. 3 A clockwise chain code

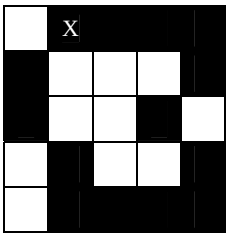

Fig. 4 An example of closed boundary

By considering the chain code in Fig. 3, chain code of the example edge in Fig. 4 is written as,

$$(0,0,0,2,3,1,2,4,4,4,6,5,6,7) \quad (2)$$

for the initial point marked with X.

B. Corner Detection

Important feature of an image is its corner points. These points have to be detected and preserved to maintain the accurateness of the image. Corner detectors are constructed based on the definition of corners itself. Generally a corner is defined as a point with highest curvature locally[10-11]. In curve fitting, corner is commonly set endpoint of a curve segment, and it is also the connection point between two curve segments[12-13].

Chord to point distance (CPD)[2] is one of the effective corner detectors as compared in[14]. The technique is based on the distance between a point to its related chord. The Euclidean distance (D_j) from each point (C_j) to the related chord between the two chord endpoints is calculated. Potential corners are assigned as the points with the highest distance for each chord. This set of potential corners is then filtered using range, *R*. However, this technique also has a tendency to detect false corners. False corners may appear because of the jagged boundary points caused by the noise in the image. This study has improved the technique by setting a threshold, *T* to filter the corners and eliminate false corners.

T is calculated as the average of highest distance, $D_{j,i}$ for all points, *n*.

$$T=\frac{\sum_{i}^{n} D_{j,i}}{n} \quad (3)$$

The improved CPD is shown in Fig. 5.

For each contour points C_i
  Determine $C_{k,i}$
End For

For each pair C_i and $C_{k,i}$
  For each C_j, $(j=i+1,i+2,...,k-1)$
    Calculate distance D_j
  End For
$D_{j,i} = \max\{D_j\}$
End For

For every $D_{j,i}$
  Corner = $\{C_i : \max\{D_{j,i}\} \geq T \in R\}$
End For

Fig. 5 Improved CPD algorithm

Parameter *k* in Fig. 5 is user-defined which gives the length of the chord. The detected corners using CPD and improved CPD is shown in Fig. 6 and 7.

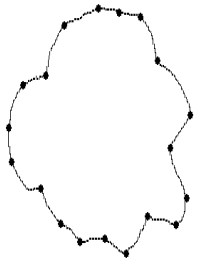

Fig. 6 Detected corners using CPD

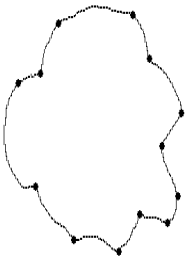

Fig. 7 Detected corners using improved CPD

Fig. 6 shows the 19 corners detected using CPD with $k = 20$. The detected corners then reduced to 13 corners as shown in Fig. 7. Some past studies have a special corners elimination process to delete false corners, for example using the corner angle[15], and collinear check[16]. The improved CPD has compiled the elimination process into the detection process, and save the processing time.

In curve fitting using Bezier and B-spline, the detected corners are used as the endpoints of curve segments corresponding to the definition of corner itself. In this study using Beta-spline, these corners are detected to be preserved while reducing other boundary points before the curve fitting.

C. Curve Fitting

In curve fitting using Beta-spline, no control points detection is needed since all boundary points are set as control points. Therefore, *m* boundary points generate *m* curves. This is the reason of boundary points reduction in Section B.

The purpose of curve fitting is to get the correct connection points for surface alignment.

Cubic Beta-spline equation on the parameter *u*, *R(u)*, given in matrix form is

$$R(u) = [U][M][P] \tag{4}$$

where,

$$[U] = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \tag{5}$$

$$[M] = \frac{1}{\delta}\begin{bmatrix} -2\beta_1^3 & 2(\beta_2+\beta_1^3+\beta_1^2+\beta_1) & -2(\beta_2+\beta_1^2+\beta_1+1) & 2 \\ 6\beta_1^3 & 3(\beta_2+2\beta_1^3+2\beta_1) & 3(\beta_2+2\beta_1^2) & 0 \\ -6\beta_1^3 & 6(\beta_1^3-\beta_1) & 6\beta_1 & 0 \\ 2\beta_1^3 & \beta_2+4(\beta_1^2+\beta_1) & 2 & 0 \end{bmatrix} \tag{6}$$

$$\delta = \beta_2 + 2\beta_1^3 + 4\beta_1^2 + 4\beta_1 + 2 \tag{7}$$

For each curve $R_i(u)$ the control point set is,

$$[P_i] = \begin{bmatrix} C_i & C_{i+1} & C_{i+2} & C_{i+3} \end{bmatrix} \tag{8}$$

Finally, a number of curve points (*NC*) of every fitted curve $R_i(u)$ then are extracted. The interval *h* between curve points is set as,

$$h = \frac{1}{NC} \tag{9}$$

and the curve point at *h* is $R(h)$.

A contour with *m* boundary points will has *m* curves. After curve points extraction, the total curve points *TNC* number of the contour is,

$$\text{total curve points} = (m \times NC) + m \tag{10}$$

For Fig. 2 case, it has 545 boundary points and 5995 curve points with $NC = 10$ which is sufficient to be considered for surface alignment before surface fitting.

D. Surface Fitting

Before fitting the Beta-spline surface, the contours are aligned. Contour alignment is to ensure the correct corresponding points between contours and to avoid generating a twisted surface. There are two ways of extracting the corresponding points of a contour which are either using arc length (AL) or angle from centroid (AC). The comparison is shown in Fig. 8 and 9.

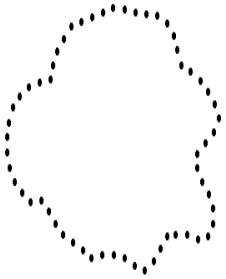

Fig. 8 Corresponding points using AL

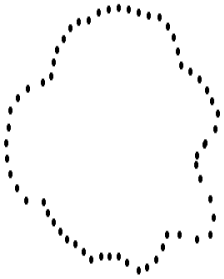

Fig. 9 Corresponding points using AC

Fig. 8 and 9 show the 60 corresponding points which are extracted using AL and AC respectively. Points in Fig. 8 are better uniformly distributed than points in Fig. 9. This is because AC cannot work well on complicated boundary especially with the flipped area. For the Stanford bunny image, examples of flipped areas are the feet, tail, and ears. Consequently, AL is the suitable method to be used.

Arc length of a parametric curve $R(u) = (x(u), y(u))$ is given as,

$$L(u) = \int_0^1 \sqrt{x'(u)^2 + y'(u)^2}\, du \tag{11}$$

Generally, formula in (11) does not work for all curves and time consuming too. Therefore, some modification is made as done in[1].

This study use Euclidean distance to calculate the length between curve points since each contour contains large number of curve points.

Euclidean distance L_i between curve points is calculated as,

$$L_i = \left\| C_i - C_{i+1} \right\| \tag{12}$$

The corresponding points P_i for each contour then calculated based on cumulative arc length and interval, *h*.

$$h = \frac{\sum_i L_i}{Q} \tag{13}$$

where Q is the required number of corresponding points.

Cumulative arc length CL_i at point *i* is the total arc length from the first point to the point *i*. The corresponding point P_i is calculated as shown in Fig. 10.

For each interval *h*
  $h_i = i \times h$, $i = 0,1,2,..., Q-1$
End For

For each h_i
  For each curve point *j*
    If $h_i - CL_j \le 0$
      Set *j* as corresponding point P_j
  End For
End For

Fig. 10 Corresponding points calculation algorithm

All contours used in this study are closed. Initial point is set as a point with zero (0) angle degree from centroid for all slices. This is to ensure a uniform initial point for all contours. Centroid of a contour is calculated as,

$$\text{Centroid} = \frac{\sum_i^{TNC} C_i}{TNC} \tag{14}$$

The angle from centroid for each curve point is,

$$\theta_i = \tan^{-1}\left[\frac{C_i(y) - \text{Centroid}(y)}{C_i(x) - \text{Centroid}(x)} \right] \tag{15}$$

Example of a contour with its centroid and initial point is in Fig. 11.

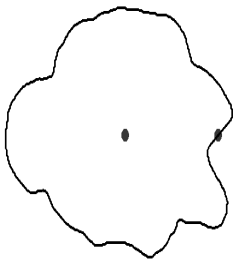

Fig. 11 A contour with its centroid and initial point

In Fig. 11, the middle point of the image is the centroid. The point on the image boundary is the calculated initial point. The rest of the corresponding points are as shown in Fig. 8.

After the corresponding points for all contours have been obtained, the Beta-spline surface is fitted. Based on (4), the equation of Beta-spline surface is,

$$R(u,w) = [U][M][P][M]^T[W]^T \tag{16}$$

With $[W] = \lfloor w^3 \quad w^2 \quad w \quad 1 \rfloor$, and *P* is the 3×3 control points.

Some contours are branching contours, as shown in Fig. 12 and 13.

Fig. 12 A based contour

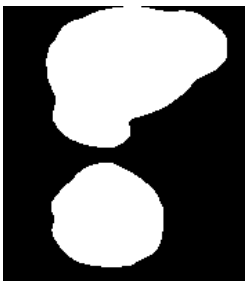

Fig. 13 A branched contour

For case in Fig. 12 and 13, it is called as one-to-many branching problem. Fig. 12 contains only a contour, and has to be connected with Fig. 13 which contains two-subcontours. This is solved by inserting a composite contour between both contours[17].

Corresponding points for the three contours are extracted and modified based on its based or branched respectively. The modified based and branched then are compiled as a new contour called composite contour. The composite contour then is inserted at a new level z_{com} .

$$z_{com} = \frac{z_{based} + z_{branched}}{2} \tag{17}$$

Based and branched contours are connected to this composite contour correspondingly as shown in Fig. 14.

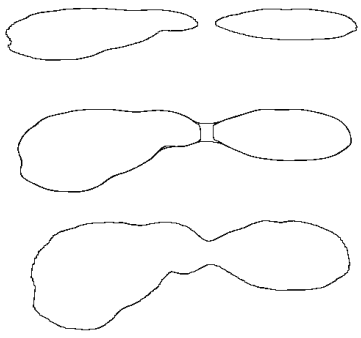

Fig. 14 The based, branched and composite contours

III. Result and Conclusion

With minimum number of surface points, Beta-spline can approximate the original image with high accuracy which is evaluated based on visual 3D image generated compared to the original image.

The reconstructed Stanford bunny using Beta-spline is shown in Fig. 15 and 16.

Fig. 15 Stanford bunny using Beta-spline from left view

Fig. 16 Stanford bunny using Beta-spline from front view

Result in Fig. 15 an 16 used only 60 corresponding points for each contour of the reconstruction. Higher number of corresponding points will increase the accuracy of the reconstructed image, but it is not effective in terms of time usage. Original Stanford bunny is shown in Fig. 17 and continued by other results using different techniques are shown in Fig. 18-20.

Fig. 17 The original Stanford bunny[5]

Fig. 18 3D Stanford bunny using Level Set method[18]

Fig. 19 3D Stanford bunny using Parametric Triangular Bezier[19]

Fig. 20 3D Stanford bunny using Radial Basis Function[20]

Comparison of generated bunny using Beta-spline in Fig. 15 and 16 with the original and other techniques shows the capability of Beta-spline in producing a good reconstructed image. However, since a minimum number of data points is used in this reconstruction, result in Fig. 15 and 16 seems missed some features of the original surface.

The original Stanford bunny has a rough skin due to its fur, but this feature is smoothed and vanished in some results. In this study, it is because of some data contours are skipped and not considered in the reconstruction. However, the result may be improved if the number of data points used is increased.

Smoothing process is one of other reasons of inaccurate resulting image. Too high degree of smoothing will eliminate the important points of the data. Nevertheless, smoothing process is important in removing the jagged points.

Beside the accuracy of resulting image, another criterion to measure the technique performance is the total processing time. Since results in Fig. 18-20 are generated by different researchers with different software and technology being used, a meaningful and satisfying comparison and conclusion cannot be made about time efficiency of the techniques.

Acknowledgment

Author thanks to Shay Kels from School of Mathematical Sciences, Tel Aviv University, Israel for the Stanford bunny data.

References

[1] F. Yahya, "Automatic G1 parametric fitting of curves and surfaces to outlines of images," PhD, School of Mathematics, Science University of Malaysia Penang, Malaysia, 2009.

[2] A. Masood and M. Sarfraz, "Capturing outlines of 2D objects with Bézier cubic approximation," *Image and Vision Computing*, vol. 27, pp. 704-712, 2009.

[3] X. D. Chen, *et al.*, "Computing the minimum distance between a point and a clamped B-spline surface," *Graphical Models*, vol. 71, pp. 107-112, 2009.

[4] W. Wenping, *et al.*, "Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization," University of Hong Kong2004.

[5] (2011, July 9th). *The Stanford 3D Scanning Repository*. Available: http://graphics.stanford.edu/data/3Dscanrep/

[6] M. Sarfraz and A. Masood, "Capturing outlines of planar images using Bézier cubics," *Computers & Graphics*, vol. 31, pp. 719-729, 2007.

[7] S. A. Halim, *et al.*, "The geometrical feature of weld defect in assessing digital radiographic image," 2011, pp. 189-193.

[8] N. A. Hadi, *et al.*, "3-Dimensional Beta-spline wireframe of human face contours," 2012, pp. 110-114.

[9] H. Haron, *et al.*, "Chain code algorithm in deriving T-junction and region of a freehand sketch," *Jurnal Teknologi*, vol. 40, pp. 25-36, 2004.

[10] D. M. Tsai, *et al.*, "Boundary-based corner detection using eigenvalues of covariance matrices," *Pattern Recognition Letters*, vol. 20, pp. 31-40, 1999.

[11] A. Masood and M. Sarfraz, "Corner detection by sliding rectangles along planar curves," *Computers & Graphics*, vol. 31, pp. 440-448, 2007.

[12] Q. Zhu, *et al.*, "Auto-Corner Detection Based on the Eigenvalues Product of Covariance Matrices over Multi-Regions of Support," *Journal of Software*, vol. 5, pp. 907-917, 2010.

[13] F. Yahya, *et al.*, "An Automatic Generations of G^1 Curve Fitting of Arabic Characters," in *International Conference on Computer Graphics, Imaging and Visualisation*, 2006.

[14] A. H. Normi, *et al.*, "Competent Corner Detectors for Outline Image," in *The Third International Conferencen on Computer Engineering and Technology*, Kuala Lumpur, Malaysia, 2011, pp. 319-324.

[15] X. C. He and N. H. C. Yung, "Corner detector based on global and local curvature properties," *Optical Engineering*, vol. 47, p. 057008, 2008.

[16] Y. Xiong and J. Joseph J. LaViola, "Revisiting ShortStraw: improving corner finding in sketch-based interfaces," presented at the Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling, New Orleans, Louisiana, 2009.

[17] J. H. Ryu, *et al.*, "Contour-Based Algorithms for Generating 3D CAD Models from Medical Images," *International Journal of Advanced Manufacturing Technology*, vol. 24, pp. 112-119, 2004.

[18] P. X. D. Y. C. Zhang, "3D Surface Reconstruction Using Level Set Method," ed, 2010.

[19] Y. Liu and S. Mann, "Parametric triangular B\ézier surface interpolation with approximate continuity," presented at the Proceedings

of the 2008 ACM symposium on Solid and physical modeling, Stony Brook, New York, 2008.

[20] B. S. Morse, *et al.*, "Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions," presented at the ACM SIGGRAPH 2005 Courses, Los Angeles, California, 2005.