Modeling and Rendering with eXpressive B-Spline Curves

Hock Soon Seah
Nanyang Technological University
ashsseah@ntu.edu.sg

Budianto Tandianus
Singapore Institute of Technology
budianto.tandianus@singaporetech.edu.sg

Yiliang Sui
Nanyang Technological University
yiliang.sui@ntu.edu.sg

Zhongke Wu
Beijing Normal University
zwu@bnu.edu.cn

Zhuyan Zhang
Nanyang Technological University
d190004@e.ntu.edu.sg

ABSTRACT

eXpressive B-Spline Curve (XBSC) is a resolution-independent and computationally efficient technique for vector-based stroke modeling and rendering with the flexibility in defining and adjusting the shape and other parameters of the stroke. It generalizes the existing Disk B-Spline Curve (DBSC) geometric representation, which itself is a generalization of the Disk Bézier curve. XBSC allows flexible shape and color manipulation and rendering of strokes with asymmetrical shape control and rich color management. These properties make XBSC suitable for modeling freeform stroke shapes and animation, specifically in squash and stretch, a common technique to bestow elasticity and flexibility in shape changes. During the squash and stretch animation computation, we constrain the shape of the XBSC stroke to conserve its area. To achieve this, we apply the simulated annealing algorithm to iteratively adjust the XBSC while maintaining its area. We show several drawings, rendering and deformation examples to demonstrate the robustness of XBSC.

Keywords

XBSC, DBSC, B-spline, Vector Graphics, Diffusion Curve, Deformation, Simulated Annealing, Computer Animation.

1. INTRODUCTION

A vector-based stroke means outlining a shape with some line thickness and color. Such stroke is resolution-independent and can be scaled without losing image quality. An example is the Disk B-Spline Curve (DBSC) [Sea05a, Wu21a], which enables varying thickness on a B-Spline curve by storing a radius parameter at each control point. For example, in Fig 1, each stroke in the Chinese calligraphy, painting, and portrait can be represented by a single DBSC. With conventional representations such as B-Spline and line segment, the strokes must be defined using several discrete lines or polygonal approximation to form the close regions. Modifying such a close region would not be as simple as modifying a DBSC stroke. However, DBSC is symmetrical about its skeleton and has only one stroke cap style (i.e., semi-circle). XBSC addresses these limitations by extending the DBSC representation in both shape modeling and color management.

Figure 1. Chinese calligraphy (left), Chinese painting (center), Portrait (right). Image from [Wu21a].

2. RELATED WORK

Several stroke representations, such as parametric curves [Sio90a, Pud94a], elliptical arcs [Com15a], line segments [Str86a], and raster image [Whi83a], have been proposed. These representations allow translation from physical to digital medium. However, the abovementioned representations have their respective disadvantages. Raster image representation is resolution-dependent, which results in large amount of data and is not editable.

Boundary-based method may cause mismatch between the upper and the lower boundary curves. Hsu's method requires complex calculation of the offset distances. Interval B-spline equation is only C^0 continuous causing difficulty in studying differential properties. To represent a non-zero-width stroke shape, typically existing solutions (including Poisson Vector Graphics [Hou18a] and Generalized Diffusion Curves [Orz13a, Jes16a] represent its shape with multiple curves that form the stroke region.

To address these shortcomings, DBSC was proposed. DBSC is a culmination of previous works in stroke drawing based on B-Spline representation. For example, stroke representation with a centerline and thickness [Hsu94a], using a B-spline as stroke skeleton [Pha91a], and interval B-spline as strokes [Su02a]. DBSC itself has also undergone development over the years. For example, intersections between DBSCs [Ao18a], brush shape modeling and in between drawing generation using DBSC [Sea05a, Sea05b]. Owing to its properties, DBSC has also been used to draw digital Chinese calligraphy [Wan16a, Fu16a].

Based on the DBSC formulation, a DBSC has symmetrical shape along its skeleton, which limits its capability in modeling a freeform shape. Thus, XBSC, which is a generalization of DBSC by enabling asymmetrical shape, was proposed [Sea22a]. Owing to its properties, XBSC allows flexible shape and color editing, compact stroke representation, asymmetrical shape control, and exciting color management. As a result, XBSC is suitable for drawing and representing freeform shapes. Fig. 2 demonstrates the differences in modeling and rendering between DBSC and XBSC. Using the same B-spline skeleton, XBSC can produce more wide-ranging shapes compared to DBSC. DBSC would require the artist to shape the skeleton in a complex manner to achieve the same drawing as XBSC.

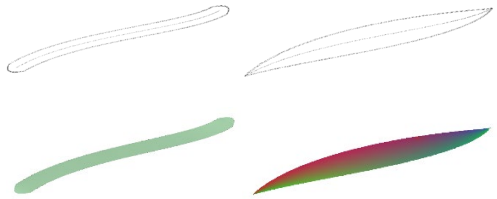

Figure 2. DBSC with symmetrical outline and uniform coloring (left) and XBSC with non-symmetrical outline and diffused coloring (right).

Shape deformation is an important topic in computer graphics [Gon13a]. Moreover, the conservation of area during deformation is necessary to simulate realistic elastic deformations. Shape deformation is typically an expensive calculation. We investigated different deformation techniques in both 3D [Por03a] and 2D [Che98a] and decided to advance a step further with area/volume conservation along with deformation. Diziol et al. [Diz11a] proposed a volume-conserving deformation method. However, it is not intuitive to use it in an interactive application as it is not possible to specify deformation constraints.

Compared to our preliminary paper on XBSC [Sea22a], the main contributions in this paper are the incorporation of color diffusion, stroke deformation with area preservation, and user interaction design to draw and manipulate an XBSC stroke.

3. PROPOSED SOLUTIONS: EXPRESSIVE B-SPLINE CURVES (XBSC)

Definition

The basic XBSC curve enables users to change the radii and color defined at each control point. It also enables user to change the shape of its end sections. The XBSC equation is defined in the following equation:

$$X(t) = \begin{cases} h_X^1(t), & t < k_0 \\ \sum_{i=0}^{n} N_{i,p}(t)\, \boldsymbol{P}_i, & k_0 \le t \le k_m \\ h_X^2(t), & t > k_m \end{cases} \qquad (1)$$

where $N_{i,p}$ is a B-spline basis function with degree p and $\boldsymbol{P}_i$ is the i-th control point. $\boldsymbol{P}_i = \langle \boldsymbol{x}_i; r_i^1; r_i^2; \boldsymbol{c}_i^1; \boldsymbol{c}_i^2; \varphi \rangle$ comprises $\boldsymbol{x}_i$ which are the spatial coordinates of the control point, r_i^1 and r_i^2 which are the radii of the two sides of the XBSC skeleton, $\boldsymbol{c}_i^1$ and $\boldsymbol{c}_i^2$ which are the colors of the two sides of the XBSC, and φ which is a color function used to control the transition between colors. Its default is linear and can be changed to radial. We define $h_X^1(t)$ and $h_X^2(t)$ to be the cap functions for both stroke ends (with the parameter t before and after the knot values k_0 and k_m).

Details

Referring to Eqn. 1, assume that $N_{i,p}(t)$ is the i-th item of the basis function of a B-spline curve whose degree is p, and knot vector $\boldsymbol{T} = \{t_0, t_1, \ldots, t_m\}$, then $m = n + p + 1$. When an XBSC is drawn with $n + 1$ control points with a specified degree p, the knot vector $\boldsymbol{T}$ is automatically computed. By default, $p = 3$. Fig. 3 shows an XBSC whose radii of the two sides of the XBSC skeleton are defined independently and two

different end caps. This allows wider flexibility in shape drawing compared to DBSC such as brush and ink pen drawings.

In terms of stroke cap, XBSC allows other stroke cap shapes beyond the semi-circle cap as defined in DBSC. This enables simulating various drawing styles as shown in Fig. 4. As for the color, XBSC uses Diffusion Curve (DC) [Orz13a] which allows more expressive coloring compared to DBSC.

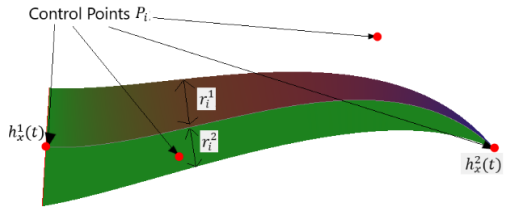

Figure 3. An XBSC with four control points P_i, $i = 0..3$, with radii r_i^1 and r_i^2 respectively, and end caps $h_X^1(t)$ and $h_X^2(t)$.

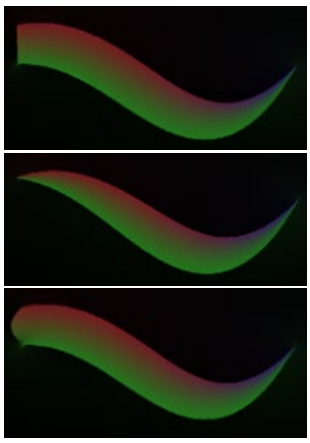

Figure 4. Different caps at the left side of a same stroke. Flat (top), Pointed (middle), Circular (bottom).

Computation of color dispersion in DC is based on a physics calculation technique used to calculate an electrical potential field. DC calculates color potential field instead of calculating electrical potential field by solving Poisson equations. In DC, thin curves serve as boundary values and the color for both sides of the curves are transformed to Laplacian values in the Poisson equations. In XBSC, the envelops are considered as DCs and color of both sides of DCs correspond to external (e.g., canvas) and internal (within stroke) colors.

Using the RGB format, we calculate the dispersion separately for each color channel by using diffusion. The RGB values are converted into vectors, and we represented the partial derivatives of the Poisson equation in matrix form:

$$\boldsymbol{Ax} = \boldsymbol{b} \quad (2)$$

with $\boldsymbol{b}$ being the divergence of the color gradient of each pixel in the XBSC image, $\boldsymbol{A}$ is a diagonal matrix representing the partial derivative coefficient, and $\boldsymbol{x}$ is the color of each pixel to be solved, which will give the color for every pixel throughout the canvas and stroke.

In Fig 4, the internal colors are the mix of red and green, with the black color as external color to match the canvas background. The internal colors determine the colors within the XBSC stroke. The external or outer colors are used to match colors outside of the XBSC stroke. The outer colors are used to match the canvas or background color, it can also be used when drawing multiple XBSC strokes next to each other to make sure the colors blend properly. Finally, the outer color is used when one draws XBSC stroke within another XBSC stroke.

DBSC can be applied to animation by keyframing its control point properties such as position and radius. Thus, in the user interface, the user adjusts the control points directly in each keyframe. An animation aspect that has not been tackled in DBSC is constraint-based deformation. In this paper, we apply deformation on the strokes with constraint on its area and it is generally known as squash and stretch. Other constraints, such as fixing a particular control point, is possible. To realize the deformation, we use simulated annealing [Kir83a] as the optimization algorithm. Given an initial stroke configuration, the user adjusts its skeleton and recompute the XBSC radii by using simulated annealing. The user may also constrain some radii. Hence, the constraint in the optimization is the skeleton shape and radii of some control points, and the outputs are the radii of the other control points that are not set as constraints.

User Interactions

To illustrate the ease of drawing and expressiveness of XBSC strokes, we designed the user interactions and develop a visualization tool, which we refer to as eXpressiveDrawing. Fig. 5 shows the step-by-step process of creating an XBSC stroke, which has a default degree of 3. The number of control points can be any value with a minimum of 4. The user first draws the required number of control points, followed by deciding the radii and colors to use for each control points. Finally, the caps at both ends of the stroke are decided by the user. Each end cap is an XBSC, and its shape can be either a point, flat line, semi-circle, or general shape. Some of these end caps are shown in Fig 6.

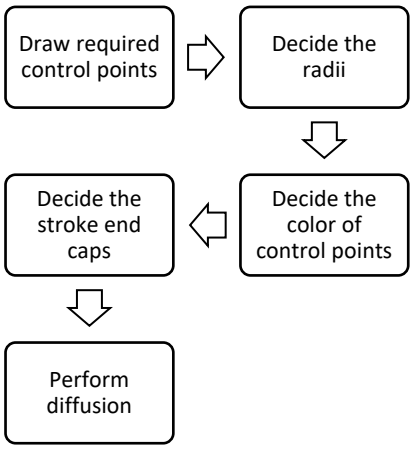

Figure 5. Steps to drawing an XBSC.

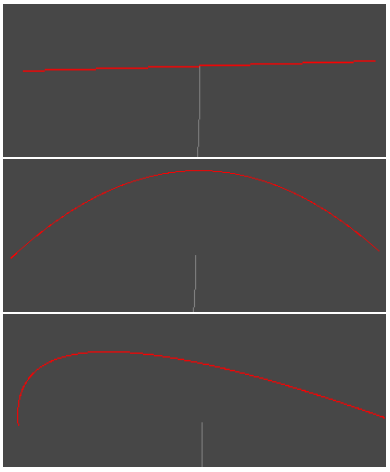

Figure 6. The red color lines are the end caps, the white lines are the XBSC skeleton. A flat end cap (top), a semi-circle cap (middle), a general shape cap (bottom).

As shown in Fig. 7, a window will pop up at the cursor location where a control point position is to be added. The window allows a user to change the key attributes of the control point. As the user changes the colors and radii, they will be visually reflected as colored circles, depicting intuitive changes in both colors and radii at the same time. If the user decides not to add the current control point to the current curve, he will click the "Cancel" button. Additionally, color and radius attributes are usually similar between neighboring control points. As a result, the design of the pop-up window in eXpressiveDrawing adheres to a scheme where the initial value of the attributes will be the same as the last confirmed control point.

At each control point, there will be two other colors which control the color of outer sides of an XBSC for diffusion computation. Since it is less frequently used, it is defaulted to null. If needed, a smaller color picker button is implemented to open another window for color selection. Keeping the narrative for the base case to a minimum lead to clean and simple operation of the user interface under normal circumstances.

In eXpressiveDrawing, a single right click will group all ungrouped control points in the canvas into a single curve. In Fig. 8, the outlines of a stroke are formed from four control points. The white color curve in the middle represents the central skeleton of the stroke. The order of the control point in the curve depends on the order that the user draws the control point. For example, the last control point created by the user right before the right click will be the end control point of the XBSC.

The two concentric circles around the control points represent the color and size for each envelop of the XBSC curve. The user can drag the circles to decide the width of the envelop and change the colors using the interface shown in Fig 7.

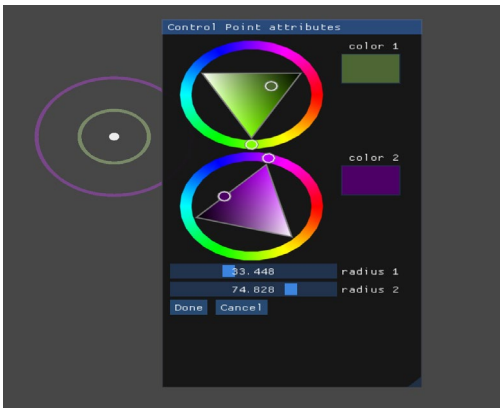

Figure 7. Interface for adding a control point in eXpresiveDrawing. At the left, the white dot depicts the position of the control point, the two other circles depict the colors and radii of the two sides of the XBSC. A pop-up window in the middle allows user to set the colors and the radii.

Fig. 8 shows the results of four control points with different concentric circles forming an XBSC accordingly Furthermore, eXpressiveDrawing is designed to be interactive. When the user changes the properties of control points like the color and radius, the changes will be reflected instantly on the outline of the stroke.

Many artists use existing images as reference or inspiration when creating their own drawings or paintings. This can help them to accurately depict certain elements or details, or to capture a certain mood or atmosphere. In the window menu shown in Fig. 9, eXpressiveDrawing provides users with

the functionality of loading an image file as a background. The user can then freely create a stroke outline on top of the background image.

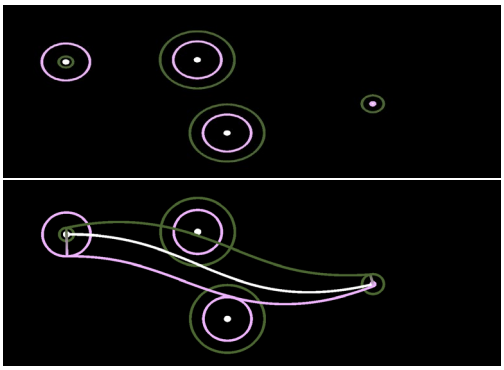

Figure 8. Four control points (top) and an XBSC curve formed from the control points (bottom). The white line depicts the skeleton, and the green and purple lines depict the envelops of the XBSC.

Figure 9. Tracing the outline of an eye based on background image, which was taken from [Orz13a].

4. EXPERIMENTAL RESULTS

Shape Modeling and Rendering

With XBSC, we can create a variety of shapes using the two radii on both sides of the skeletal B-spline curve. The color also follows the same principle and formula as the radii resulting in a gradual change and flow of color. The XBSC allows colors to diffuse along the curve. Adding diffusion properties allows the color to blend completely without the clear divide in the middle.

We recreated the eye example in Fig. 9 from [Orz13a] to illustrate the ease of drawing with XBSC. The skeletons of the XBSC lines are shown in Fig. 10 (top). The eyelid, the eyeball and the white of the eye are each created using an XBSC stroke. The color is defined on each control point of the skeleton, after diffusion, the image of the eye is rendered as shown in Fig. 10 (bottom).

Fig. 11 shows an example of a drawing with 5 XBSCs to form an apple.

Shape Deformation

The extended capabilities of the XBSC to control the radii of the shapes drawn allows us to perform Shape Deformation. By altering the radii and moving the control points, the size and shape of any drawings can be quickly altered from one form to another without any constraints. However, to create realistic deformation, a user may want to constrain the shape of the deformation to preserve the area of the shapes during the deformation process. Since this shape deformation is done by changing the radii of the control points, it can only deform a single XBSC stroke to preserve the area. As the area is computed for a single XBSC stroke, multiple XBSC strokes need to be deformed separately.

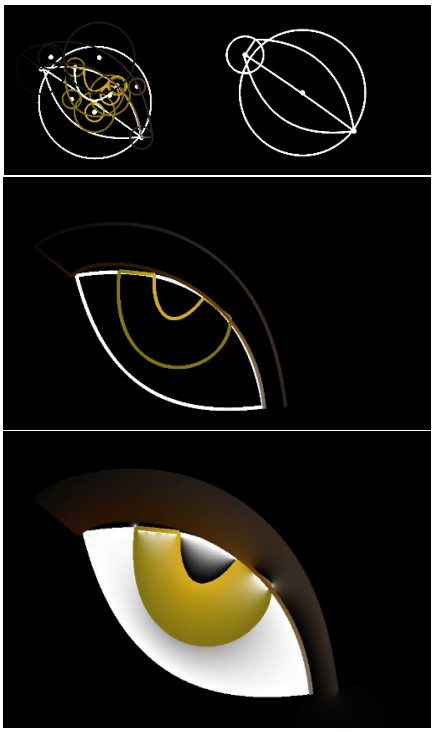

Figure 10. Zoomed-in view of XBSC skeletons of an eye drawing with control points (top), the drawing before diffusion (middle) and the final rendered result (bottom).

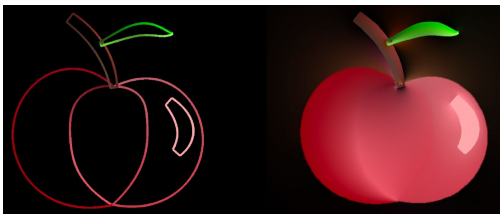

Figure 11. Five XBSCs form an apple outline (left) and rendered (right).

To calculate the surface area, we tessellate the XBSC into a triangle mesh, see Fig. 12. We compute the area of each triangle using cross product of two of its edge vectors. The area of the triangle mesh is obtained by summing the areas of all the triangles in the mesh. Using the same method, we also calculated the areas of the end caps of the XBSC.

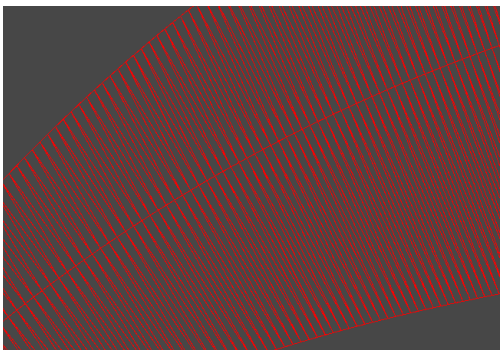

Figure 12. A section of a triangulated XBSC. Top and bottom curves are the envelops and central red curve is the skeleton of the XBSC.

With the area calculated, by using a simulated annealing optimization formula, we iteratively enable the XBSC to change the radii to obtain a new area to approximate the previous area. The previous area and the new area are saved, and the user can decide which control points should not be changed through the entire iteration of the formula. This provides user with some control over the resulting shapes. The users can decide the number of iterations, the rate of change for the radii and which radii to remain unchanged. As it is impossible for the user to determine the exact changes for each radius, a random value that ranges from 0.0 to 1.0 will be chosen and will multiply the rate of change previously determined by the user. These values will be added or deducted from the radii. The area will be recalculated using these new radii and compared to its previous area. Using the acceptance probabilities of simulated annealing and depending on how close this new area is compared to the previous area, the program will either keep these radii or revert to the previous radii values. With each iteration, the changes in the radii will alter the area of the current shape to closer to the previously decided saved area. How close the area value depends on the number of iterations and the difference between the previous area and the new area.

Furthermore, users may draw XBSC markings in the inside of an XBSC object or shape. When the object is deforming, the deformation will also affect the markings accordingly. The markings' control points are tied to the closest point on the deforming XBSC (i.e., either side of its envelop or its skeleton). Hence, when the main XBSC object deforms, their inner marking control points change along with it. Fig. 13 shows an example of such a deformation. Having reduced its height, the vase deforms while maintaining the same area and its markings change accordingly.

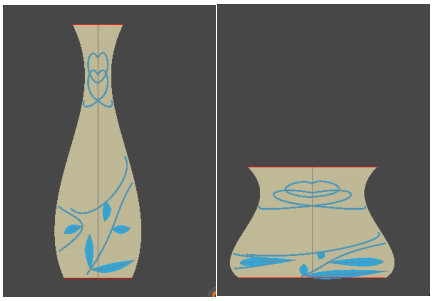

Figure 13. A vase object being deformed (left to right). The blue markings on the vase are constrained with the same deformation.

Fig. 14 shows another example of the vase being stretched and squashed to illustrate the ease with which constant area (but not the same shape) can be maintained in an animation.

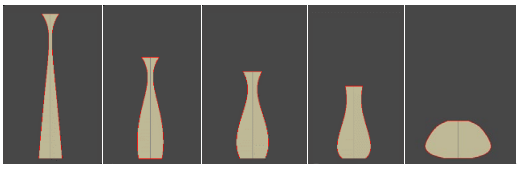

Figure 14. Vase being stretched and squashed while maintaining the same area. The original vase (middle), the stretched vases (left), the squashed vases (right).

Computational Cost

The computational cost of drawing an XBSC without shape deformation and color diffusion is only marginally more than drawing a B-spline curve with similar quality. The shape deformation computes the change of the radii of the XBSC. It takes around 1 second for 100 iterations, which is not too high. Using the biconjugate stabilization method to solve Eqn. 1, the diffusion computation takes around 2 seconds to complete a 700x700 image. This is the most compute intensive task.

5. CONCLUSION

We have discussed the properties of XBSC and its use in modeling and rendering shape and deformation. From our investigation we show that by using XBSC artists have more freedom of drawing expression not only in terms of shape and color, but also in terms of shape deformation. As

mentioned before, deformation is currently limited to a single stroke, which can be improved to enable multiple stroke deformation. XBSC has potentials in the future and there are rooms for improvement. For instance, applying GPU-based Poisson equation solver to accelerate the diffusion curve computation, keyframing the deformation to produce coherent and smooth deformation animation, and applying XBSC in 3D space.

6. ACKNOWLEDGMENTS

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG 22/20). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

7. REFERENCES

[Ao18a] Ao, X., Fu, Q., Wu, Z., Wang, X., Zhou, M., Chen, Q., Seah, H.S. An intersection algorithm for disk B-spline curves. Computers & Graphics. 70, 99–107, 2018. https://doi.org/10.1016/j.cag.2017.07.021.

[Che98a] Cheng, S. W., Edelsbrunner, H., Fu, P., & Lam, K. P. Design and analysis of planar shape deformation. In Proceedings of the Fourteenth Annual Symposium on Computational geometry, pp. 29-38, 1998.

[Com15a] Company, P., Plumed, R., Varley, P.A.C. A fast approach for perceptually-based fitting strokes into elliptical arcs. Vis Comput. 31, 775–785, 2015. https://doi.org/10.1007/s00371-015-1099-6.

[Diz11a] Diziol, R., Bender, J., Bayer, D. Robust real-time deformation of incompressible surface meshes. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. pp. 237–246. Association for Computing Machinery, New York, NY, USA, 2011. https://doi.org/10.1145/2019406.2019438.

[Fu16a] Fu, Q., Wu, Z., Ying, X., Wang, M., Zheng, X., Zhou, M. Generating chinese calligraphy on freeform shapes. In: Gavrilova, M.L., Tan, C.J.K., and Sourin, A. (eds.) Transactions on Computational Science XXVIII: Special Issue on Cyberworlds and Cybersecurity. pp. 69–87. Springer, Berlin, Heidelberg, 2016. https://doi.org/10.1007/978-3-662-53090-0_4.

[Gon13a] González Hidalgo, M., Torres, A.M., Gómez, J.V. (eds.) Deformation models: tracking, animation and applications. Lecture Notes in Computer Vision and Biomechanics, vol. 7. Springer, New York, 2013.

[Hou18a] Hou, Fei, Qian Sun, Zheng Fang, Yong-Jin Liu, Shi-Min Hu, Hong Qin, Aimin Hao, and Ying He. Poisson vector graphics (pvg). IEEE Transactions on Visualization and Computer Graphics 26, no. 2, 1361-1371, 2018.

[Hsu94a] Hsu, S.C., Lee, I.H.H. Drawing and animation using skeletal strokes. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. pp. 109–118. ACM, New York, NY, USA, 1994. https://doi.org/10.1145/192161.192186.

[Jes16a] Jeschke S. Generalized diffusion curves: An improved vector representation for smooth-shaded images Comput. Graph. Forum, vol. 35, no. 2, pp. 71–79, 2016.

[Kir83a] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. Optimization by simulated annealing. Science. 220, 671–680, 1983. https://doi.org/10.1126/science.220.4598.671.

[Orz13a] Orzan, A., Bousseau, A., Barla, P., Winnemöller, H., Thollot, J., Salesin, D. Diffusion curves: a vector representation for smooth-shaded images. Commun. ACM. 56, 101–108, 2013. https://doi.org/10.1145/2483852.2483873.

[Pha91a] Pham, B. Expressive brush strokes. CVGIP: Graphical Models and Image Processing. 53, 1–6, 1991. https://doi.org/10.1016/1049-9652(91)90013-A.

[Por03a] Portells, M.M., Mir, A., Perales, F. Shape deformation models using non-uniform objects in multimedia applications. In: Perales, F.J., Campilho, A.J.C., de la Blanca, N.P., Sanfeliu, A. (eds) Pattern Recognition and Image Analysis. IbPRIA 2003. Lecture Notes in Computer Science, vol 2652. Springer, Berlin, Heidelberg, 2003. https://doi.org/10.1007/978-3-540-44871-6_61

[Pud94a] Pudet, T. Real time fitting of hand-sketched pressure brushstrokes. Computer Graphics Forum. 13, 205–220, 1994. https://doi.org/10.1111/1467-8659.1330205.

[Sea05a] Seah, H.S., Wu, Z., Tian, F., Xiao, X., Xie, B. Artistic brushstroke representation and animation with disk b-spline curve. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. pp. 88–93. ACM, New York, NY, USA, 2005. https://doi.org/10.1145/1178477.1178489.

[Sea05b] Seah, H.S., Wu, Z., Tian, F., Xiao, X., Xie, B. Interactive free-hand drawing and In-between generation with DBSC. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. pp. 385–386. ACM, New York, NY, USA, 2005. https://doi.org/10.1145/1178477.1178561.

[Sea22a] Seah, H.S., Tandianus, B., Sui, Y., Wu, Z. Expressive B-spline curves: A pilot study on a flexible shape representation. In: Muramatsu, S., Nakajima, M., Kim, J.-G., Guo, J.-M., and Kemao, Q. (eds.) International Workshop on Advanced Imaging Technology (IWAIT) 2022. p. 87. SPIE, Hong Kong, China, 2022. https://doi.org/10.1117/12.2626063.

[Sio90a] Siong Chua, Y. Bézier brushstrokes. Computer-Aided Design. 22, 550–555, 1990. https://doi.org/10.1016/0010-4485(90)90040-J.

[Str86a] Strassmann, S. Hairy brushes. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. pp. 225–232. ACM, New York, NY, USA, 1986. https://doi.org/10.1145/15922.15911.

[Su02a] Su, S.L., Xu, Y.-Q., Shum, H.-Y., Chen, F. Simulating artistic brushstrokes using interval splines. In: Proceedings of the 5th IASTED International Conference on Computer Graphics and Imaging. pp. 85–90. Citeseer, 2002.

[Wan16a] Wang, M., Fu, Q., Wang, X., Wu, Z., Zhou, M. Evaluation of Chinese calligraphy by using DBSC vectorization and ICP algorithm. Mathematical Problems in Engineering. 2016, 1–11, 2016. https://doi.org/10.1155/2016/4845092.

[Whi83a] Whitted, T. Anti-aliased line drawing using brush extrusion. In: Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques. pp. 151–156. ACM, New York, NY, USA, 1983. https://doi.org/10.1145/800059.801144.

[Wu21a] Wu, Z., Wang, X., Liu, S., Chen, Q., Seah, H.S., Tian, F. Skeleton-based parametric 2-D region representation: Disk B-spline curves. IEEE Computer Graphics and Applications. 41, 59–70, 2021. https://doi.org/10.1109/MCG.2021.3069847