Curvature Minimizing Depth Interpolation for Intuitive and Interactive Space Curve Sketching

Gautam Chaudhary Koel Das M. Gopi

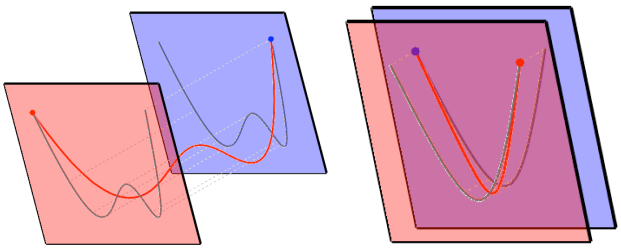
Computer Graphics Lab
University of California, Irvine
{gchaudha, kdas, gopi}@uci.edu

Abstract

Space curve sketching using 2D user interface is a challenging task and forms the foundation for almost all sketch based modeling systems. Since the inverse projection from 2D to 3D is a one to many function, there is no unique solution. In this paper, we propose to interpret the given 2D curve to be the projection of the 3D curve that has minimum curvature among all the candidates in 3D. We present an elegant algorithm to efficiently find a close approximation of this minimum curvature 3D space curve. We also compare with many other 2D to 3D curve generation techniques to show that the curve generated by our method is the one of the most "intuitive" and easy to compute. Fixing the space curve to be the minimum curvature curve does not restrict the user from getting high-curvature space curve, if desired. We analyze the complete behavior of our space curve generation algorithm in order to provide simple sketching rules to achieve the curve that the user wants, even the high-curvature space curve, with no change in our algorithm.

1. Introduction

Space curve sketching using 2D user interfaces is a challenging and active research problem in the field of interactive sketch based modeling. These non-planar 3D curves are fundamental component of surface generation [7, 20] for 3D models. Space curves are also used as input for specifying implicit surfaces and also for manipulating and modifying object deformations [16, 6]. The problem of finding the 3D space curve from a 2D sketch is mathematically indeterminable and has many possible solutions; it is well known that the fundamental difficulty is in choosing the appropriate one. Most of the sketch-based modeling tools try to avoid addressing this fundamental problem directly due to its complexity. These systems provide a fixed canvas to the user to draw the 2D curve, thus fixing the 3D curve to be on the canvas. They provide tools to change the shape of the canvas to enable the user to draw non-planar curves [8]. One of the clever uses of this canvas method was by Igarashi et al. [7] where the system provides a planar canvas first and the user starts with a planar closed curve. A rotund object is generated using that planar closed curve and from then on, the generated object becomes the canvas for all future curve drawings. Often, curves that are not drawn on the canvas are interpreted as gestural inputs for viewpoint or object manipulation. In spite of all the successes that have been achieved in the field of sketch based modeling, the fundamental problem of space curve determination still remains to be a challenging one.



(a) (b)

Figure 1. Given an orthographic projection of a 2D curve (gray curves on the planes) and the depth range (red and blue transparent planes), our algorithm computes the 3D space curve (red curve) that minimizes the maximum curvature and projects onto the given 2D curve. Though our method generates a low curvature curve, by reducing the range of depth values, space curves of arbitrarily high curvatures (b) can be generated.

In our paper, we attempt to provide an intuitive and mathematically elegant insight into this problem of 3D space curve determination from a 2D curve. We propose that the 3D curve corresponding to the given 2D curve to be the one that minimizes the maximum normal curvature in 3D. Curvature minimizing interpolation approaches have been previously used in the study of visual perception to create seamless color smoothing for overlapping multi-projector displays [12], indicating a relationship between curvature minimization and perception of desirability and intuitive-

ness. In this paper we take a similar approach to address the following problem: given a 2D curve, the projection parameters, and the depth range, find the 3D space curve within the given depth range that has the minimum curvature and matches with the 2D curve under the given projection.

Specifically, following are the main contributions of this paper:

1. We prove the existence of the curvature minimizing space curve for an orthographic 2D projection of a polynomial curve.

2. Given only a sequence of 2D curve points as input, we present an efficient algorithm to best approximate the curvature minimizing 3D space curve, from that viewpoint.

3. To the best of our knowledge, this is one of the earliest works on providing mathematical insight to the domain of generating intuitive 3D *free-form* space curves from 2D points. We would like to note that a lot of work has been done using learning approaches for interpreting 2D sketches for mechanical CAD drawings with straight line edges.

4. We use Laplacian filter effectively along with our approach to achieve the desired result of curvature minimizing depth interpolation.

2. Related Work

The different endeavors in the field of 3D curves focus either on *curve manipulation methods* or on *2D to 3D curve generation methods*. The approach mentioned in this paper belongs to the latter category.

Existing space curves can be manipulated indirectly in many commercial packages like Maya, 3DStudioMax and most other CAD tools, by modifying the curve parameters such as spline control points or knot values. Spline curves can also be manipulated directly as described in [4, 5]. Modifying existing curves by over-sketching was initially presented by Baudel [1]. This approach has been adopted for curve modification and manipulation by several other researchers in this field [2, 3]. Cohen et al. [2] used the correlation between the curve and its shadow as a visual cue to help the user compute and modify the shape of the 3D curve. Karpenko et al. [9] take a multi-view sketching approach using epipolar lines as cues to deduce shape information. Their method uses perspective projection and can determine the 3D values only when drawn from two different view points. The epipolar geometry approach works for simple cases but fails to give desirable results for more complex objects due to the nature of their algorithm.

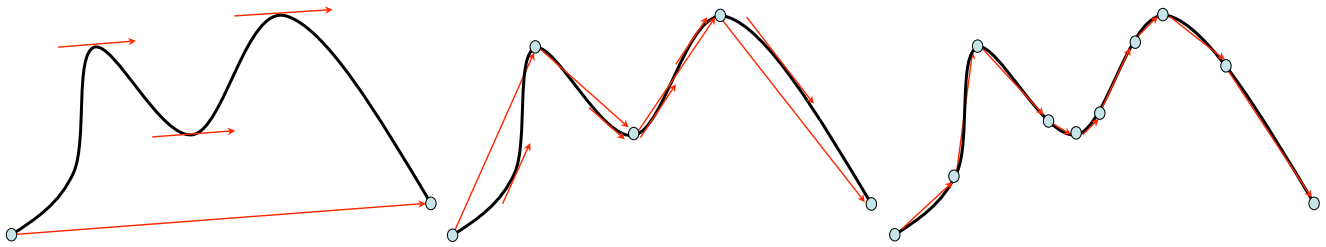
In the recent past, there has been some research devoted to extract depth information from 2D curves and strokes directly. Most of the existing literature on space curve generation relies on the domain specific knowledge. The edge-vertex graph method have been utilized to extract the depth information in systems best suited for architectural designs and CAD type applications in [10, 14]. Furthermore, [11] use an interesting domain specific learning based approach using geometric correlation to deduce the 3D model that is again more suitable for CAD tools. The underlying assumption behind this learning technique is that the input sketch comprises of straight lines and thus does not account for free-from curves. It also follows from the paper that the human ability to perceive the 3D information from 2D sketch is based on prior knowledge about the drawn object or scene and human perception fails to reconstruct random objects. Learning based methods are also non-interactive and are computationally expensive.

One of the earliest example of constructing free-form space curves is the 3-Draw system [13] that extracts the 3D coordinate information by using a tracker-based system. Ijiri et al. [8] utilize the input sketch to generate a curved canvas where the drawn curve essentially becomes an extruded surface from a given viewpoint providing the user the capability of drawing non-planar curves on that surface. Tolba et al. [18] use the approach of oriented projective representations of 2D points. This allows "3D-like" perspective viewing, object manipulation and rendering of the 2D points, but they do not generate actual 3D values from 2D points.

To the best of our knowledge, most of the existing approaches about space curve generation are based on heuristics and there is no literature on research that performs domain independent mathematical treatise to generate intuitive free form space curve. Our paper attempts to address this issue by using curvature minimizing depth interpolation. We present an interpolation method to generate artistic and expressive 3D curve from a *single* 2D curve input. It is obvious that one cannot provide the exact curve that user has in mind, just from a single 2D sketch. Hence, any curve *manipulation* technique can be used thereafter to modify our computed 3D curve. We believe that the need for such manipulation will be minimal since our user study shows that, in most cases, the initial guess of curvature minimizing curve is very close to user's expected result. The method discussed empowers the user by simplifying the user-interface, quickens the system learning time of the user and overall time to sketch the object.

3. Curvature Minimizing Curve

We repeat the problem statement for the sake of completion. *Given a 2D curve, the orthographic projection para-*



(a) (b) (c)

Figure 2. Illustration of our approximation algorithm to generate curvature minimizing space curve. At every iteration the critical points of the distance function from the line joining the first and last points of the curve segment is chosen for further curve subdivision. The recursion ends when the curve segment can be approximated well by the line joining its end points.

meters, and the depth range, the problem is to find from all the possible 3D space curves within the given depth range and that matches with the 2D curve under the given projection, the curve that minimizes the maximum curvature. The given 2D curve is assumed to be a polynomial curve. Since any polynomial basis can be converted to and from a Bernstein basis, we can convert the given polynomial curve into a 2D Bezier curve. We state, prove and use a theorem involving the control points of Bezier curves to come up with a solution for the stated problem.

Theorem 1 Given a 2D Bezier curve with a known depth range, the depth of the control points of curvature minimizing 3D Bezier curve is monotonically and equally spaced within the given depth range.

Proof: (Sketch) For the sake of simplicity of mathematical proof, we assume that the given plane is parallel to the XY plane and the viewing direction is the Z axis. Let us also assume that the range denotes the depth value of the first and the last control point of the Bezier curve.

Let the control points of the 2D Bezier curve be denoted by $P_i = (x_i, y_i)$, $0 \leq i \leq n$, where n is the order of the curve. The Bezier curves are orthographic projection invariant. Hence the 3D curve will have the same number of control points as its 2D projection, and the control points of the 2D curve are actually the projection of the 3D curve control points. Since it is an orthographic projection, the control points of all the candidate 3D space curves will have the same x and y coordinates. Let the 3D control points be $Q_i = (x_i, y_i, z_i)$. The curve is,

$$Q(u) = \Sigma_{i=0}^{n} \frac{n!}{i!(n-i)!} Q_i u^i (1-u)^{n-i}$$

Differentiating, we get

$$Q'(u) = n\Sigma_{i=0}^{n-1} \frac{(n-1)!}{i!(n-1-i)!} [Q_{i+1} - Q_i] u^i (1-u)^{(n-1-i)}$$

and the second derivative is given by,

$$Q''(u) = (n)(n-1)\Sigma_{i=0}^{n-2} \frac{(n-2)!}{i!(n-2-i)!} [Q_{i+2} - 2Q_{i+1} + Q_i] u^i (1-u)^{(n-2-i)}$$

The normal curvature κ is given by $\frac{|Q' \times Q''|}{|Q'|^3}$. Intuitively, to minimize the maximum curvature, we need to minimize the maximum value of Q'' and maximize the minimum value of Q'.

With respect to Q', larger the differences in depth values of the control points, larger the denominator and hence the minimum value has no maxima. Hence, minimizing curvature depends only on the magnitude of Q''. This can also be seen from the fact that since curvature is an implicit characteristic of a curve, for an equivalent arc-length parameterization, the curvature is controlled by its second derivative. Based on the above argument, our goal is to minimize the magnitude of the 3D vectors $(Q_{i+2} - 2Q_{i+1} + Q_i)$ for all i. Since the x and y coordinates of Q_i are fixed by the input 2D Bezier control points, and the z coordinates of the first and last control points are fixed by the given depth range, the only free parameters are the z coordinates of the intermediate control points Q_i, $1 \leq i \leq (n-1)$. Minimizing $|z_{i+2} - 2z_{i+1} + z_i|$ would minimize the magnitude of the required vector. Since the z_0 and z_n are known, setting $z_{i+2} - 2z_{i+1} + z_i = 0$ for all i provides a set of recurrence equations. The solution for this set of equations is

$$z_i = z_0 + \frac{z_n - z_0}{n} i.$$

□

If we use this theorem to construct the 3D curve from the 2D sketch, it will have seemingly two drawbacks: it can produce only monotonic curves and low curvature curves. In the problem of going from 2D to 3D, it is obvious that we have to constrain the problem to fix one solution from infinite number of solutions. Monotonicity from a given view point is a natural constraint that is required by a mathematically convincing choice of the solution – minimum curvature curve. At the same time, the generated curve can be non-monotonic in every other viewing direction and hence is not a limitation. Second, the curvature is minimized within certain constraints - the given depth range and the 2D projection. Smaller depth ranges provide less space to reduce the curvature. Hence arbitrarily high curvature curves

can be created by using appropriately small depth ranges (Figure 1(b)).

4. Our Algorithm

The 2D curves drawn in sketching applications are not usually Bezier curves, but a sequence of edge connected points. Hence, we have to adapt our stated theorem for the depth extraction for a generic 2D curve. One way to adapt our result on curvature minimizing depth interpolation is to fit a Bezier curve to the input point sequence and interpolate the depth of this 2D Bezier curve. This is a computationally expensive operation and most importantly, the error in curve fitting will provide a distracting and unexpected feedback to the user.

In this section we provide an alternative method for finding the depth of the sketched 2D curve points that approximates the minimum curvature 3D curve. We use the following fundamental observation of the 3D Bezier curve chosen in the previous section (but not true in a generic Bezier curve): since the depth values of the control points are equally spaced, the closest point on the curve from any control point has the same depth value as the control point. Hence if we identify these *key points* on the curve that are closest to the control points and space their depth values equally, we will have the first approximation of the space curve. Since we do not have these 3D Bezier curve control points and hence of its 2D projection, identifying even the correct *number of key points* is difficult, let alone the actual key points. Every lower degree polynomial curve can be represented using a function of higher degree with more than required control points. In the limit every point on the curve can be used as a control point and our interpolation scheme will boil down to an arc-length based linear interpolation scheme. Hence there is no upper bound on the number of control points that represents a polynomial curve. On the other hand, we can use the following theorem to find the lower bound on the number of control points (key points).

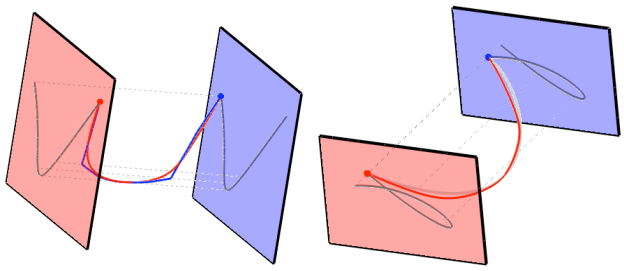
Theorem 2 Every non-degenerate critical point of the distance function from a Bezier curve to the line joining the first and the last control points of the curve corresponds to a unique control point of the curve.

In other words, even though we do not have explicit Bezier representation of the 2D sketched curve, the number of critical points in the curve is the lower bound on the number of control points of the hypothetical Bezier representation of the same curve. Proof of this theorem is given in the appendix. The critical point need not be the closest curve point (key point) to the corresponding control point. However, these critical points are in fact very close to the key points as shown by our empirical studies. The fundamental source of *approximation* in our algorithm for curvature minimizing depth interpolation comes from the fact that we assume that these critical points are key points.

The second source of approximation comes from the fact that, in general, the converse of the above theorem is not true: there need not be a critical point corresponding to every control point. Hence all the required control points cannot be found by just one sequence of critical points. We solve this issue by subdividing the original 2D curve at the initial critical point (*key point*) locations. If we perform a deCasteljau subdivision of the Bezier curve at the *key points*, the above theorem still holds good for each of the subdivided curve segments. In this process, we increase the total number of control points (sum of the control points of the subdivided curve segments) and hence the possibility of more critical points. Using this observation, we repeat the process of finding the critical points recursively within each 2D segment between the initial sequence of critical points up till a threshold in terms of linearity of the curve segment. We show empirically that we can extract the required number of critical points for faithful representation of the sketched curve. Since we assume that all these critical points are representatives of *key points*, we assign equally spaced depth values within the range to the critical points. The depth of all other points on the 2D curve is computed by linearly interpolating the depth between the critical points based on the curve length parameterization.

The resulting 3D space curve, though a very close approximation of the curvature minimizing curve, has non-smooth, high frequency transitions of the depth values across the critical points. Laplacian filtering is used in image processing applications to remove high frequency components for edge detection, morphological smoothening and contrast enhancement [15] and for smooth surface generation by removing the high curvature regions from noisy meshes in geometric modeling [17, 19]. There are other smoothing techniques like subdivision approach and Lagrange interpolation. We chose the Laplacian low-pass filter as it is extremely fast and simple as required by interactive sketching applications. But it is a known fact that this filter produces shrinkage effect for geometric primitives. In our algorithm, the shrinking of curves is prevented by keeping the depth at the curve end points constant. We apply a few iterations of this smoothening filter *only* to the interpolated depth values of all the interior points on the curve, as $z_i\prime = z_i + \lambda(\Delta z_i)$ where $\Delta z_i = ((z_{i-1} + z_{i+1})/2) - z_i$ and $\lambda < 1.0$ for a low-pass filter, . This produces an excellent approximation of the curvature minimizing curve. In Figure 3 we show a few examples of actual curvature minimizing Bezier curves and the corresponding space curves produced by our algorithm.

The summary of the algorithm for finding the curvature minimizing curve is given as pseudo-codes in Algorithms 1



(a) (b)

Figure 3. (a) Effect of filtering. Blue curve shows the interpolation before filtering and the red curve shows the curve after filtering. (b) shows reconstruction of space curves using our approximate method for different Bezier curves. The actual curvature minimizing Bezier curve as suggested in Theorem 1 is shown in gray. Our approximate curve follows the actual curve closely. Notice in (b) that even from a viewpoint where projections are self intersecting in 2D, the reconstructed curvature minimizing curve follows the actual Bezier curve.

and 2. An illustration of this algorithm is shown in Figure 2.

Algorithm 1 ReturnCriticalPoints(Curve C)

{Function: Find the Critical Points of the 2D curve C.}
{Given: Sample points of a 2D curve C}
L = Line between the first and the last sample points of C.
CS = Sequence of all the critical points of the distance function of C from L.
R = Subset of CS, that is within a user-defined threshold distance from L.
$CS = CS - R$.
if $CS == \phi$ **then**
  return; {Terminating condition.}
end if
$CriticalList = CS$
$CheckList$ = {First point of C} ∪ CS ∪ {Last point of C}
for every curve segment C_i between consecutive points in $CheckList$ **do**
  CriticalList = CriticalList ∪ ReturnCriticalPoints(C_i) { Insert into CriticalList, the critical points of C_i in order of their appearance in C.}
end for
return CriticalList;

4.1. Implementation

Here we elaborate on the implemetation aspect of our approach that are mentioned in Algorithms 1 and 2. The input to the algorithm is a sequence of points that represents a 2D curve, the values of which are discrete since they are generated from the pixel values. This would create a lot of local-

Algorithm 2 CurvatureMinimizingCurve(Curve C, Depth z0, Depth z1)

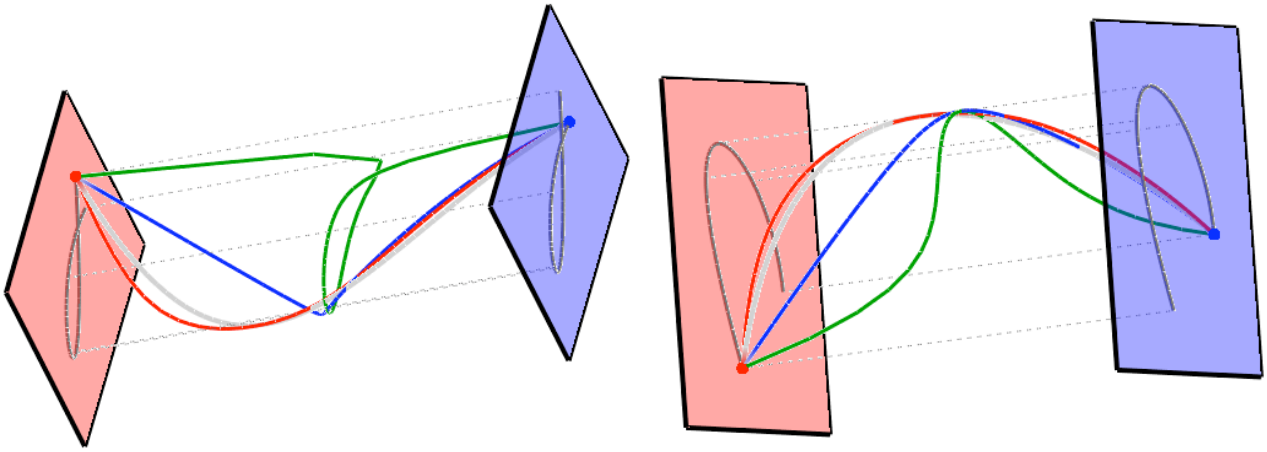
{Function: Curvature Minimizing Interpolation of Depth}
{Given: Sample points of a 2D curve C; depth values of the first and the last points of C, z0 and z1. }
$CriticalList$ = {First point of C} ∪ ReturnCriticalPoints(C) ∪ {Last point of C}
Let DepthInterpC be the points in C with unset depth values.
Interpolate the depth value linearly between z0 and z1 among all points of $CriticalList$ and set their values in DepthInterpC.
for every curve segment C_i between consecutive points in $CriticalList$ **do**
  Interpolate the depth value based on curve length parameterization and set their values in DepthInterpC.
end for
CurvMinC = LaplacianFilter(DepthInterpC)
return CurvMinC

ized extremas while computing the critical points. Hence, in order to make the algorithm more efficient and robust to the nature of the input, we choose only one extrema that is the maximum for that iteration and use it as the critical point. This process is repeated until a terminating condition is reached. Here it needs to be mentioned that the terminating condition of the algorithm is one of the open parameters that is a user-defined maximum allowable deviation of the 2D curve from the line joining the end-points of the curve. For implementation purposes, iterations continue till the critical point's deviation is less than this user-defined threshold that is given in number of pixels. For all the results presented in this paper, we used a 3-pixel threshold.

Once all the key-points are identified, they are linearly interpolated within the z-range between the start and end points of the curve. In order to achieve a smoother curve, we apply a low-pass Laplacian filter. In our implementation, we used 25 iterations for filtering with $\lambda = 0.25$. The resulting space curves before (blue curve) and after (red curve) filtering is shown in Figure 3.

5. Evaluation

In the previous section, we proposed a curvature minimizing approach to generate a feasible 3D space curve from a 2D curve. In this section we evaluate our proposed solution to the problem from various perspectives. First, we analyze the effectiveness of our algorithm in approximating the curvature minimizing space curve using error analysis. Second, we evaluate the performance of our approximating algorithm in terms of its efficiency and appropriateness of its



(a) (b)

Figure 4. The actual curvature minimizing Bezier curves are shown in gray. Our approximation method for curvature minimizing curve is shown in red, depth interpolation based on curve length is shown in blue, and the space curve produced by the distance based depth interpolation is shown in green.

use in interactive sketching applications using standard performance analysis. Third, we validate our hypothesis that the curvature minimization is an intuitive measure for depth interpolation using user studies.

5.1. Analysis of Error in Approximation

We have proposed an approximate method to generate a curvature minimizing 3D space curve. Hence, it is only logical that in order to evaluate the error in the approximation, we compare our reconstructed curve with the actual curvature minimizing curve. The inputs to our error analysis are the 3D Bezier curve whose control points are equally spaced in depth from the given view point (as proved in Section 3), and the 3D curve generated using our interpolation algorithm. The distance between these curves is the measure of error in approximation.

As seen from Figures 3 and 4, the reconstructed curve follows the actual Bezier curve closely which *visually* verifies our claim that our approach produces a curvature minimizing curve in 3D. In order to *empirically* verify our claim, we computed the maximum relative distance between the points in our reconstructed curve from the actual curve to the depth range of the entire curve. In other words, we computed the following function as the measure of error:

$$Error = \max \frac{d(u,B)}{z1 - z0}$$

where $d(u,B)$ is the closest distance between a point u in the reconstructed curve to the given Bezier curve B, and $(z1 - z0)$ is the given depth range of the entire curve.

The error analysis was carried out with 50 different curves. The proposed method consistently gave an error of less than 0.1%. The shape of the curve is controlled by the number of critical points and the smoothness of the curve is determined by the density of sample points constituting the input curve.

The number of critical points usually stabilize after a certain density of sample points and hence the error rate is not dependent on the number of sample points in the projected curve. Under exactly the same experimental conditions, distance based interpolation scheme exhibited an error of about 30% and curve-length based interpolation had more than an order of magnitude error than our method.

5.2. Timing Analysis

The algorithm should be time efficient in order to be useful for interactive applications like sketch based modeling. The algorithm takes less than 30 ms to compute the space curve from a 2D curve consisting of around 200 sample points. Our method has linear time complexity in terms of the number of sample points, as shown in Figure 5. The experiments were conducted using a variety of curve configurations on a 2.4GHz Pentium platform with 512MB memory.

5.3. User study

We hypothesize that the depth interpolation that minimizes the maximum curvature generates one of the most intuitive space curves. In order to validate this claim, we conducted a preliminary user study in which, from the 2D curve chosen by the user, space curves were reconstructed using different interpolation techniques, including the one described in this paper. Most of the existing space curve generation techniques use either a planar canvas or a pre-defined canvas.

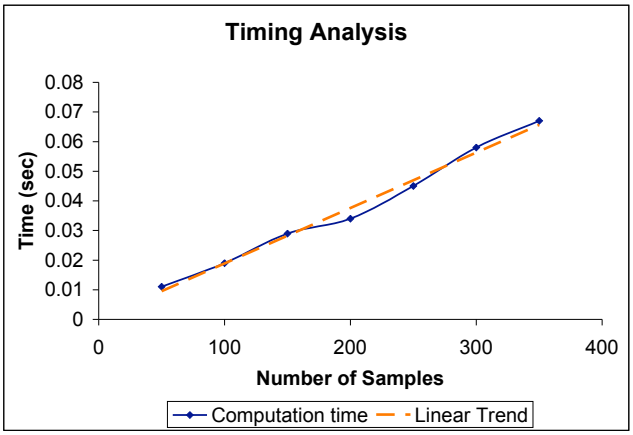


Figure 5. The graph shows the execution time for depth interpolation for input curves of varying sizes.

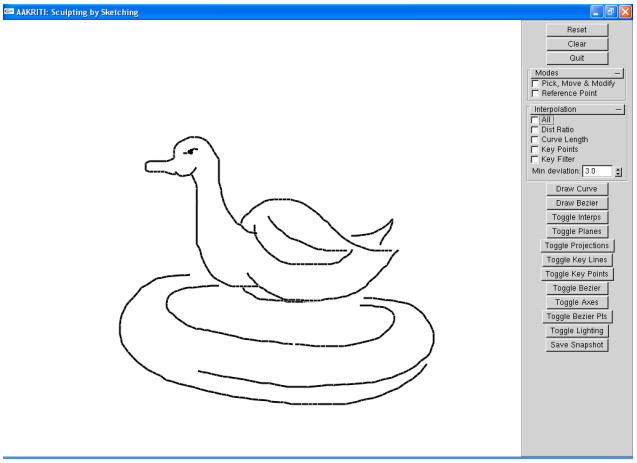


Figure 6. A snapshot of our preliminary sketch-based free-form modeling system.

There has been no significant research for generating a free-form space curve directly from a single viewpoint. Thus, in our experiments, we compare our approach with two common interpolation techniques based on *curve-length parametrization* and *distance ratios*. Given the depth values at the end points of the curve,

1. *Curve Length Parametrization* technique linearly interpolates and computes the depth of a point based on the ratio between the curve length up till the point and the total curve length;

2. *Distance Ratio* technique linearly interpolates and computes the depth of a point based on the ratio between the distances of that point from the end points of the curve.

The space curves generated by the above two interpolation techniques and the proposed technique were presented to the user. The user was allowed to view these space curves from different view points in order to get acquainted with the system and also familiarize with the 3D space in which the curves exist. The chosen 2D curve was also shown on transformed projection planes in order to help the user to orient him/herself using familiar markers. Then the users were asked to choose the curve that was most intuitive and that best conveyed their expected interpolation. All the users most emphatically (91%) chose the curve generated by our method as the most intuitive curve. The user study was carried out with more than a dozen users. Around 100 different curves were generated and every user was allowed to experiment with at least 6 different curves.

6. Conclusion

In this paper we have addressed the difficult problem of generating an intuitive and aesthetic space curve from a single view of a 2D curve without using any additional cues. We believe that the mathematical rigor used to explain the intuitiveness would find its use in perceptual studies. Further, the results of this work can be used to effectively quantify the quality of curve and surface generation in free-form modeling and used extensively for a better surface generation for 3D models in sketch based modeling applications. The proposed method in this paper for generating intuitive 3D curves from 2D input is being utilized in a sketch-based free-form modeling system that we are currently developing. A snap-shot of our preliminary free-form sketching system is shown in Figure 6.

References

[1] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of 7th annual ACM Symposium on User Interface Software and Technology*, pages 185–192. ACM Press, 1994.

[2] J. M. Cohen, L. Markosian, R. C. Zeleznik, J. F. Hughes, and R. Barzel. An interface for sketching 3d curves. In *Proceedings of the Symposium on Interactive 3D graphics*, pages 17–21. ACM Press, 1999.

[3] T. Fleisch, F. Rechel, P. Santos, and A. Stork. Constraint stroke-based oversketching for 3d curves. In *Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM-04)*, pages 161–166. Eurographics Association, August 2004.

[4] B. Fowler and R. Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, 1993.

[5] C. Grimm and M. Ayers. A framework for synchronized editing of multiple curve representations. In *Proceedings of the EUROGRAPHICS*, volume 17, pages C31–C40, 1998.

[6] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *Proceedings of the Symposium*

on Interactive 3D graphics, pages 139–142. ACM Press, 2003.

[7] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH*, pages 409–416. ACM Press, August 1999.

[8] T. Ijiri, T. Igarashi, S. Takahashi, and E. Shibayama. Sketch interface for 3d modeling of flowers. Technical Sketch at ACM SIGGRAPH, August 2004.

[9] O. Karpenko, J. F. Hughes, and R. Raskar. Epipolar methods for multi-view sketching. In *Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM-04)*, pages 167–174. Eurographics Association, August 2004.

[10] H. Lipson and M. Shpitalni. Identification of faces in a 2d line drawing projection of a wireframe object. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(10):1000–1012, 1996.

[11] H. Lipson and M. Shpitalni. Correlation-based reconstruction of a 3d object from a single freehand sketch. In *AAAI Spring Symposium on Sketch Understanding*, pages 99–104, 2002.

[12] A. Majumder and R. Stevens. Perceptual photometric seamlessness in projection-based tiled displays. *ACM Transactions on Graphics*, 24(1):118–139, 2005.

[13] E. Sachs, A. Roberts, and D. Stoops. 3-draw: A tool for designing 3d shapes. *IEEE Computer Graphics and Applications*, 11(6):18–26, 1991.

[14] A. Shesh and B. Chen. Smartpaper–an interactive and easy-to-use sketching system. In *Proceedings of Eurographics*, pages 409–416, August 30-September 2 2004.

[15] P. G. Sibley and G. Taubin. Atlas-aware laplacian smoothing. IEEE Visualization Poster, August 2004.

[16] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of ACM SIGGRAPH*, pages 405–414. ACM Press, 1998.

[17] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH*, pages 351–358. ACM Press, 1995.

[18] O. Tolba, J. Dorsey, and L. McMillan. Sketching with projective 2d strokes. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 149–157. ACM Press, 1999.

[19] J. Vollmer, R. Mencl, and H. Muller. Improved laplacian smoothing of noisy surface meshes. In *Proceedings of Eurographics*. Eurographics Association, September 1999.

[20] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3d scenes. In *Proceedings of ACM SIGGRAPH*, pages 163–170. ACM Press, August 1996.

Appendix: Proof of Theorem 2

Theorem 2: Every non-degenerate critical point of the distance function from a Bezier curve to the line joining the first and the last control points of the curve corresponds to a unique control point of the curve.

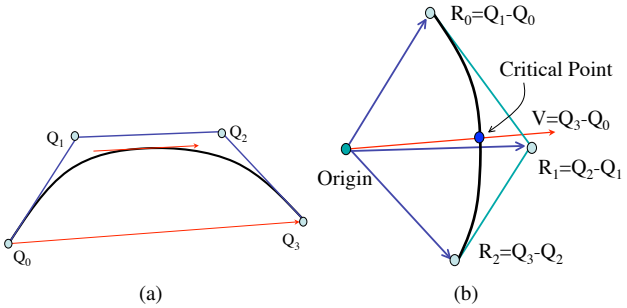


Figure 7. Illustration of the proof for Theorem 2. (a) Bezier curve $Q(u)$ with four control points. (b) Bezier curve $Q'(u)$ which represents tangent vector at each point of $Q(u)$. The intersection point corresponds the parameter value of the critical point in the original curve.

Proof: For a Bezier curve $Q(u)$ in Figure 7, let $\vec{L}$ denote the vector between the first and the last control points $Q(0)$ and $Q(n)$. Let us find the ranges of the parametric values within which the curve reaches the critical distances (maximas and minimas) from the line L. The extremas will have their tangent vectors parallel to $\vec{L}$. The locus of the tangent vectors at every point on $Q(u)$ is also a Bezier curve as shown below.

$$\begin{aligned} Q'(u) &= n\Sigma_{i=0}^{n-1} \frac{(n-1)!}{i!(n-1-i)!}[Q_{i+1}-Q_i]u^i(1-u)^{(n-1-i)} \\ &= (k+1)\Sigma_{i=0}^{k} \frac{k!}{i!(k-i)!}R_i u^i(1-u)^{k-i} \end{aligned}$$

where $k = n-1$ and $Q_{i+1} - Q_i = R_i$

Computing the critical points on $Q(u)$ at which the tangent vector is parallel to $\vec{L}$ is same is computing the intersection points of $Q'(u)$ with $\vec{L}$. The number of intersection points of $\vec{L}$ and $Q'(u)$ will be no more than the number of intersections of $\vec{L}$ with the control polyline of $Q'(u)$, due to the variation diminishing property of the 2D Bezier curves. It can be shown that if the line L intersects $Q'(u)$ at $u = u_t$, where $\frac{i}{n-1} \leq u_t \leq \frac{i+1}{n-1}$ for some $0 \leq i \leq (n-1)$, then it intersects the line R_iR_{i+1} of the control polygon. Since two line segments, L and R_iR_{i+1}, can intersect at most once, there can be at most one intersection point u_t within the given range due to the variation diminishing property. It can be observed that $\frac{i}{n-1} < (u_i = \frac{i}{n}) < \frac{i+1}{n-1}$, for $0 < i < n$, where u_i is the parametric value at which the control point Q_i has maximum control over the curve and hence the closest point on the curve. Since there is at most one critical point and exactly one control point of the original curve corresponding to each range, we correspond the critical point to the unique control point that share the same range of parametric values. □