Computer-Aided Design 46 (2014) 227–232

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Technical note

Self-overlapping curves: Analysis and applications

Uddipan Mukherjee

University of California, Irvine, USA

ARTICLE INFO

Keywords:
Self-overlapping curve
Immersion

ABSTRACT

When a disk in 2D is stretched arbitrarily with possible self-overlaps, without twisting it, its boundary forms a complex curve known as a self-overlapping curve. The mapping between the disk and its deformed self, also called an immersion of the disk, is useful in many applications like shape morphing and curve interpretation. Given a self-overlapping curve, an algorithm for computing its immersion is presented, which has an average time complexity quadratic in the number of points on the curve.

1. Introduction

Consider a circular disk in 2D, painted blue on one side and red on the other. The disk is placed in a plane such that only the blue side is visible. If the disk is now arbitrarily stretched and given a multiple number of self-overlaps, without twisting it at any time, still only the blue side is visible. The boundary of this disk is called a *self-overlapping curve* [1] (Fig. 1). Trivially, every simple closed-loop curve without self-intersections is a self-overlapping curve. The homeomorphism or one-to-one mapping between the disk and its deformed self is known as an *immersion* of the disk.

Self-overlapping curves occur in many physical systems. In graphite, which is one of the crystalline forms of carbon, hexagonal rings of connected carbon atoms form a self-overlapping polygon [2] (Fig. 2). A method for computing the disk immersion gives a way of interpreting the polygon from the boundary curve laid out by the carbon atoms.

In keyframe animations, objects between keyframes are often self-overlapping polygons. Morphing between the objects by considering their boundaries only does not produce intuitive morphs [3]. Computing immersions of the source and target shapes, and morphing one immersion to another produces aesthetically better and intuitive morphs (Fig. 3). Self-overlapping curves also appear as boundaries of lanes in a freeway. Computing an immersion in such cases clearly brings out the layout of the road pattern (Fig. 4).

An algorithm for computing the immersions of a self-overlapping curve is presented in this paper. The curve is segmented into a set of simple curves (Jordan curves) by non-trivial line segments (Fig. 5). The input to the algorithm is a sequence of points, in order, forming the curve. It has an average time complexity quadratic in the number of points on the curve.

Interestingly, there can be multiple immersions of a disk corresponding to a self-overlapping curve, i.e. a disk may be overlapped in multiple ways to produce the samecurve as the boundary (Fig. 6). The immersion algorithm presented here computes all possible immersions of a given self-overlapping curve.

2. Related work

Self-overlapping curves were studied in detail by Shor et al. [1], wherein an algorithm to detect whether a closed-loop self-intersecting curve is self-overlapping was proposed. This algorithm runs in $O(n^3)$ time, where n is the number of points on the curve. For curves in general positions (also known as self-overlapping polygons), where the intersection points do not coincide with any of the curve points, this algorithm also produces an arbitrary triangulation of the curve interior. For curves with multiple immersions, [1] proposes an algorithm to compute the number of immersions in $O(n^3 \log n)$ time. Applications like shape morphing require a 'good' Delaunay-like triangulation of self-overlapping polygons to produce aesthetically intuitive morphs, which cannot be obtained immediately from [1]. A method for obtaining such a triangulation from an arbitrary one, by a series of edge flips, for a self-overlapping polygon was proposed in [3]. However, the number of edge flips required for the same is unknown. Obtaining a suitable triangulation of the curve interior catering to the needs of the application concerned is thus challenging – even more so – because no known method of triangulating a simple polygon can be easily extended to self-overlapping polygons.

Mukherjee et al. [4] present an abstract representation of a self-overlapping curve which may be helpful in computing its immersions, but does not explicitly produce one. Guo et al. [2] investigate the different immersions produced by a self-overlapping curve generated by overlapping graphite atoms. Brinkmann et al. [5] and Graver [6] generalize the problem for (m, k) patches, $(m, k \geq 3)$ where regular m-gon tiles are attached to form the patches with k tiles meeting at an interior vertex and $k-1$ tiles meeting at boundary vertices.

Given a general self-intersecting curve, Titus [7] and Blank [8] examine the condition when it is self-overlapping. However, the exact time complexities of these methods are not mentioned.

E-mail address: umukherj@ics.uci.edu.

0010-4485/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cad.2013.08.037

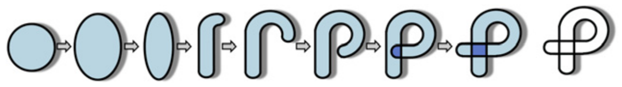


Fig. 1. A disk painted blue on the front and red on the back side is stretched and overlapped (left to right) without twisting such that only the blue side is always visible. The disk boundary is a self-overlapping curve (extreme right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The algorithm presented in this paper explicitly computes all the immersions from a given self-overlapping curve. It has an average time complexity of $O(n^2)$, where n is the number of points on the curve. The given self-overlapping curve is divided into a set of simple curves, each of which can be triangulated in any known way of triangulating simple polygons, thereby making the method suitable for applications like shape morphing.

Although not directly related to this paper, the work of Eppstein et al. [9] requires special mention where it was proved that extruding an immersion of a self-overlapping curve to a non self-intersecting surface in 3D (embedding) is NP complete.

3. Definitions

Self-overlapping curve: A self-intersecting closed-loop curve which can be divided into a set of simple curves by non-trivial line segments (Fig. 7). Traversing the curve in a particular direction (e.g. clockwise) from a starting point defines a natural interior of the curve. Every simple polygon is trivially a self-overlapping curve.

An edge of a curve is defined as the line segment joining two adjacent points on the curve. If the curve intersects in general positions only, i.e. the edges of the curves intersect only at non-terminal points, it is called a self-overlapping polygon. The formal definition of self-overlapping curves can be compared with an intuitive one introduced at the beginning of the paper.

Immersion: A continuous function $i : M \rightarrow T$ such that for any point $v \in M$ there exists a neighborhood $u(v)$ within which i restricts to a homomorphism from $u(v)$ to $i(u(v))$ [9].

A self-overlapping curve forms the boundary of a deformed disk. An immersion of a self-overlapping curve is a one-to-one mapping between the interior of the curve and the disk.

Crest point: A local extremum point with respect to the horizontal such that the curve takes a left turn at the crest point, assuming that the interior of the curve is to its right (Fig. 8). If the crest point is at a local maxima, it is called a *maximal crest point*, and if it is at a local minima, it is called a *minimal crest point*.

4. Computing immersions

Given a self-overlapping curve, horizontal rays directed towards the right (positive X axis) are drawn for each crest point P. If P has an ordinate value y, the ordinate value of the corresponding ray is $y - \epsilon$, if P is a maximal crest point, and $y + \epsilon$ if P is a minimal crest point, where ϵ is very small. These rays intersect the curve at various points including a pair of points in the local neighborhood

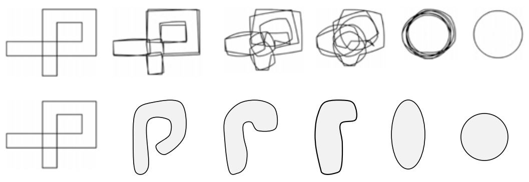


Fig. 3. Morphing boundaries only (top) and interior (bottom) of a self-overlapping polygon.

Fig. 4. An aerial view of a nested freeway. The boundaries of the lanes are self-overlapping curves (partly highlighted in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of each crest point. The ray–curve intersection points are defined as *cuts*. If the curve crosses a ray r, the cut is termed as positive and marked as r, or termed as negative and marked as r', according as the curve crosses the ray from right to left or left to right. The ray and its corresponding cuts are marked with the same symbol but it will be clear from the context which one is referred. The cuts in the local neighborhood of a crest point (having ordinate values $y \pm \epsilon$, where y is the ordinate value of the crest point) are called crest cuts. They are referred to as maximal or minimal crest cuts according as the corresponding crest point is a maximal or minimal crest point. Traveling along the curve, two consecutive cuts i and j are called adjacent cuts (Fig. 9).

A self-overlapping curve can be potentially segmented into two self-overlapping curves, by cutting it along a pair of positive and negative cuts from the same ray. If such a pair exists, it is called a *valid* pair of cuts. However, if by cutting along a pair of positive and negative cuts yields two curves, either of which is non self-overlapping, the cut pair is invalid (Fig. 10). A cut pair is defined to be left valid if the portion of the original curve to the left of the corresponding ray is self-overlapping, and right valid if the curve to the right of the ray is self-overlapping. Thus, a valid cut pair is both left valid and right valid.

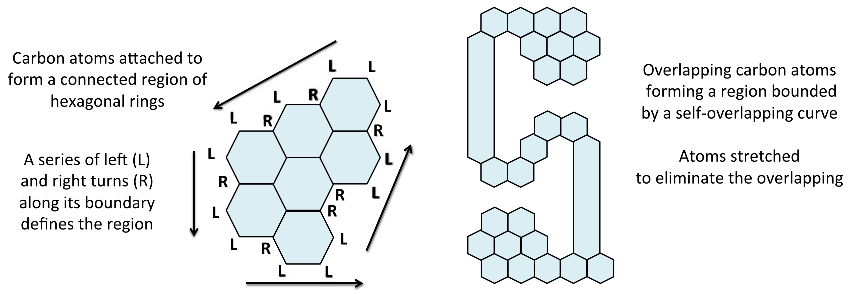

Carbon atoms attached to form a connected region of hexagonal rings
A series of left (L) and right turns (R) along its boundary defines the region
Overlapping carbon atoms forming a region bounded by a self-overlapping curve
Atoms stretched to eliminate the overlapping

Fig. 2. Self-overlapping polygon in graphite atom.

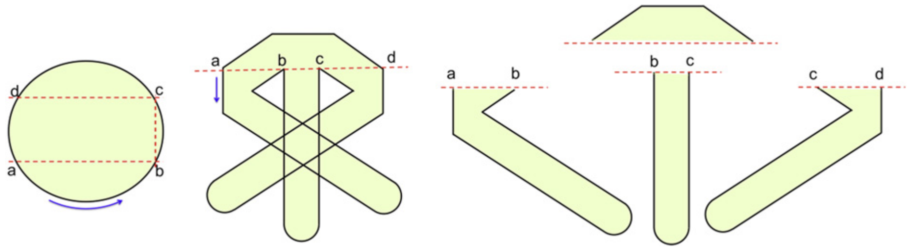


Fig. 5. A self-overlapping curve (center) is segmented into a set of simple curves by non-trivial line segments (dotted). These line segments are chords on the disk dividing it into mutually exclusive segments.

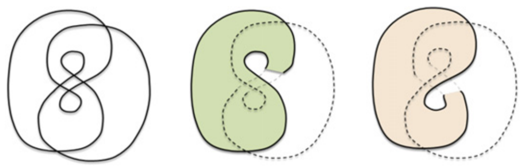


Fig. 6. Two immersions (center and right) of the Milnor's doodle (left).

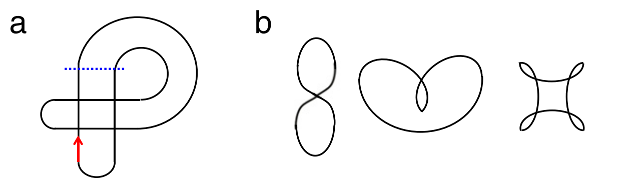


Fig. 7. (a) Self-overlapping and (b) non self-overlapping curves. Traversing (a) along the red arrow, has its interior to its right.

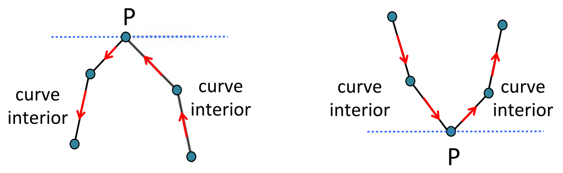


Fig. 8. Maximal(left) and minimal(right) crest points on a self-overlapping curve. The curve interior is on its right.

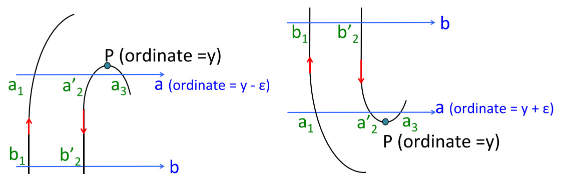


Fig. 9. Rays drawn corresponding to the crest points of a self-overlapping curve and the cuts produced. a_1, a_3 are positive cuts and a_2' is a negative cut. a_2' and a_3 are crest cuts—maximal (left) and minimal (right).

A cut pair (a_1, a_2') is left valid if either of the following conditions are satisfied (Fig. 11).

(i) Positive cut a_1 is the previous adjacent cut of negative cut a_2', both lying on the ray a.
(ii) b_1 and b_2', on ray b, are next adjacent cut of a_1 and previous adjacent cut of a_2' respectively, and (b_1, b_2') is a left valid cut pair.
(iii) The two minimal crest cuts a_3' and a_4, corresponding to ray a, lie between a_1 and a_2', and (a_1, a_3') and (a_4, a_2') are left valid cut pairs.
(iv) a_2' is a maximal crest cut, a_3 is the other maximal crest cut, and if a_p' is a cut lying to the right of a_3, (a_1, a_p') is a left valid cut pair and (a_3, a_p') is a right valid cut pair.
(v) a_1 is a maximal crest cut, a_3' is the other maximal crest cut, and if a_p is a cut lying to the left of a_3', (a_p, a_3') is a right valid cut pair and (a_p, a_2') is a left valid cut pair.

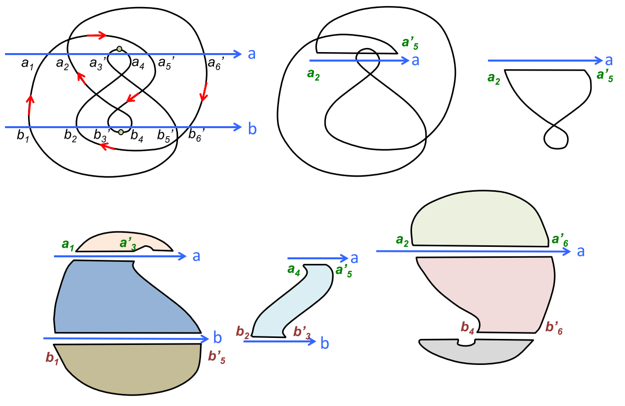


Fig. 10. Segmenting a self-overlapping curve (top left) into two curves along cut pairs. Top right: cutting along the pair a_2, a_5' produces two curves each of which is non self-overlapping. Bottom: cutting along the designated valid cut pairs (shown alongside) produces a set of simple curves (interiors are colored) which are trivially self-overlapping.

The conditions for the cut pair to be right valid are similar except that the part of the curve from a_2' to a_1 is considered and the interpretations concerning the minimal and maximal crest points are interchanged.

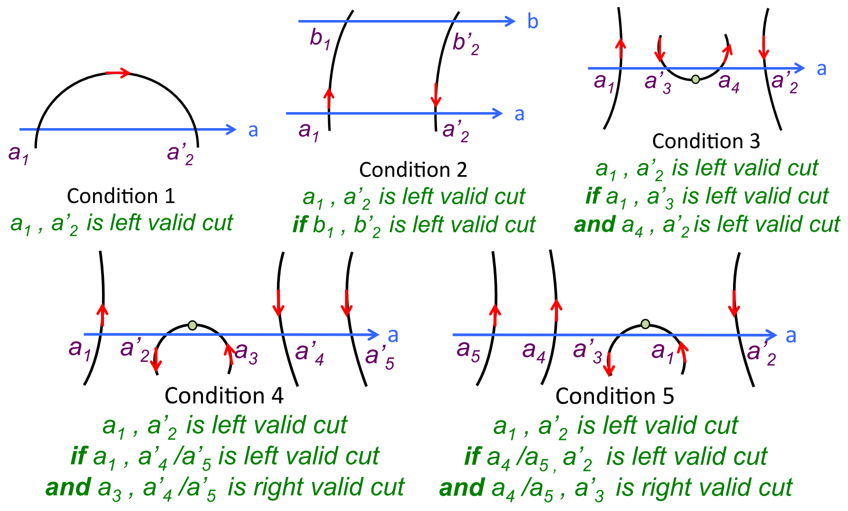
A cut pair (a_1, a_2') is left invalid in either of the following cases (Fig. 12).

(i) a_1 has a next adjacent cut a_3' lying between a_1 and a_2' on the ray a.
(ii) a_2' has a previous adjacent cut a_3 lying between a_1 and a_2' on the ray a.
(iii) The segment of the curve from a_1 to its next adjacent cut intersects the segment of the curve from a_2' to its previous adjacent cut.

Assuming a triangulation of the curve interior, a valid cut complies with an internal edge of that triangulation and an invalid cut is incompatible with any internal edge of any possible triangulation. With this framework, the immersion algorithm identifies the set of valid cut pairs which segment a given self-overlapping curve into simple curves. The number of rays drawn are sufficient for this purpose. This can be proved by contradiction in the following way. Suppose a curve segment produced by the algorithm does not bound a simple polygon. In that case the polygon loops over to intersect itself and there is at least one crest point on the polygon from which a ray can be drawn. The valid cut pairs on this ray further divide the curve.

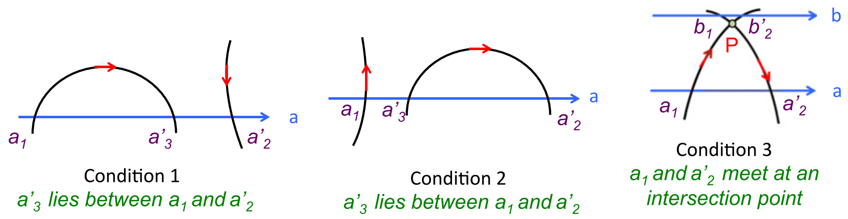
4.1. Immersion algorithm

Given a self-overlapping curve, the crest points are identified and the cuts corresponding to rays are computed. A *validity tree* is generated for the curve with nodes signifying left or right validity of cut pairs. For example, if a node P represents the left validity of a cut pair C_1, and the latter is left valid only if either cut pair C_2 or C_3 is left valid, then P has two children nodes signifying the left



a₁
a'₂
a
Condition 1
a₁ , a'₂ is left valid cut
b₁
b'₂
b
a₁
a'₂
a
Condition 2
a₁ , a'₂ is left valid cut
if b₁ , b'₂ is left valid cut
a₁
a'₃
a₄
a'₂
a
Condition 3
a₁ , a'₂ is left valid cut
if a₁ , a'₃ is left valid cut
and a₄ , a'₂ is left valid cut
a₁
a'₂
a₃
a'₄
a'₅
a
Condition 4
a₁ , a'₂ is left valid cut
if a₁ , a'₄ /a'₅ is left valid cut
and a₃ , a'₄ /a'₅ is right valid cut
a₅
a₄
a'₃
a₁
a'₂
a
Condition 5
a₁ , a'₂ is left valid cut
if a₄ /a₅ , a'₂ is left valid cut
and a₄ /a₅ , a'₃ is right valid cut

Fig. 11. The five conditions (counter clockwise from top left as listed in the text) for a cut pair (a_1, a'_2) to be left valid.



a₁
a'₃
a'₂
a
Condition 1
a'₃ lies between a₁ and a'₂
a₁
a'₃
a'₂
a
Condition 2
a'₃ lies between a₁ and a'₂
b₁
b'₂
b
P
a₁
a'₂
a
Condition 3
a₁ and a'₂ meet at an intersection point

Fig. 12. The three conditions (from left to right as listed in the text) for a cut pair (a_1, a'_2) to be invalid.

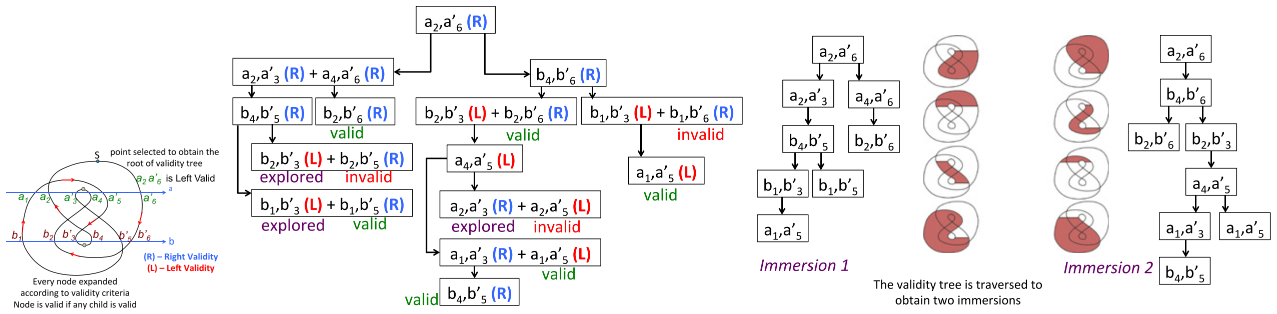


Fig. 13. Immersion algorithm applied to the self-overlapping curve (left). The validity tree with valid, invalid and explored nodes is shown in the center. The right figure shows a complete path of valid cut pairs from root to leaf nodes along with the segmented simple curves with colored interiors.

validities of C_2 and C_3. A node signifying a valid cut pair by default, e.g. if the cut pair is adjacent, has no children.

An arbitrary point on the curve to the left of the topmost ray is selected. The adjacent cuts corresponding to this point are left valid by construction. The root of the validity tree is selected as the node signifying the right validity of this cut pair (Fig. 13). Starting from the root, each node is expanded by adding its children according to the correct validity criteria, and a node already expanded earlier is not expanded further. Thus the validity tree has a finite number of nodes. The validity of the root follows a series of its valid descendants upto a leaf node in the tree, cutting the curve along which produces a set of simple curves.

The correctness of the algorithm can be proved as follows. Assume that each node of the tree is expanded based on its validity criteria irrespective of whether the node has been expanded earlier. Imagine that the cut points are also part of the curve. Then the line joining a valid cut pair can be thought of as an edge in a valid triangulation of the curve interior. Since the triangulation of a disk is a tree, the path of valid cut pairs starting from the root must terminate in a child node in the validity tree. The fact that a node is not expanded more than once may break the continuity of the path of valid cut pairs but does not affect the termination or correctness of the algorithm. By traversing the validity tree along nodes signifying valid cut pairs, all the valid cut pairs can be obtained which divide the original curve into a set of simple curves. The time complexity of the algorithm is thus proportional to the number of nodes in the validity tree.

4.2. Complexity analysis

If there are n points on the curve, there can be at most n crest points, and hence n rays in the worst case. Each ray can have at most n intersections with the curve. Hence the number of cut pairs on a ray is n^2, and the total number of cut pairs is n^3. Hence, the running time of the immersion algorithm, which is proportional to the number of cut pairs, is $O(n^3)$ in the worst case.

If there are m rays, R_1 to R_m, they divide the plane into $m + 1$ smaller blocks, S_0 to S_m each having a finite number of curve points (Fig. 14). On an average, each block has the same number of points, say p from the curve.

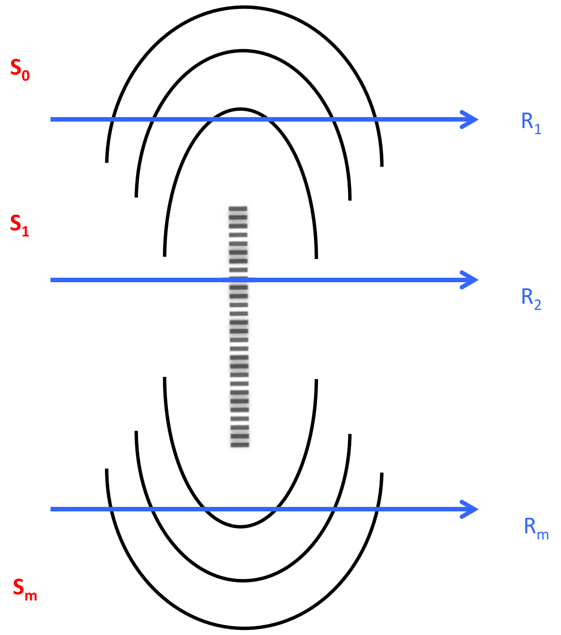


Fig. 14. Rays drawn on the self-overlapping curve divide the plane into blocks, each having a finite number of curve points.

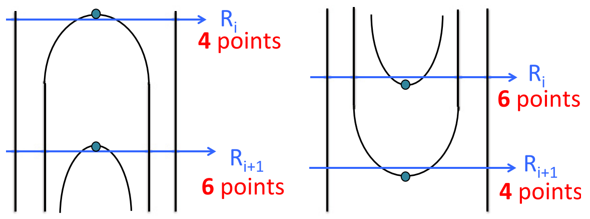


Fig. 15. The number of cuts c_i and c_{i+1} for rays R_i and R_{i+1} are related as $c_i = c_{i+1}-2$ if the crest points are both maximal, and $c_i = c_{i+1} + 2$ if both are minimal.

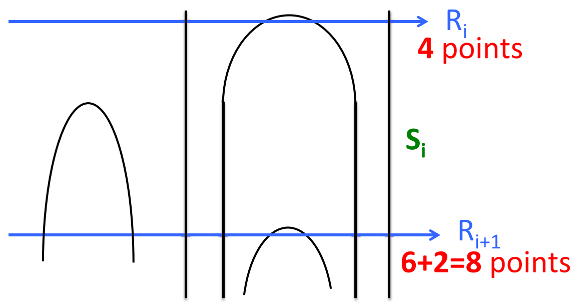


Fig. 16. The number of cuts on R_{i+1} is four more than that of R_i as both crest points are maximal and the curve has a loop in seg. S_i.

The ray R_1 has p cuts on it as the segment S_0 has p points on an average. Assume that all the cuts on a ray have no adjacent cuts on the same ray except for the terminal rays R_1 and R_m. Under this condition, if the number of cuts on any ray R_i is c_i and that on the adjacent ray R_{i+1} is c_{i+1}, then $c_i = c_{i+1}$, or $c_i = c_{i+1} - 2$ or $c_i = c_{i+1}+2$ according as the corresponding crest points are different in nature, both maximal or both minimal (Fig. 15). On an average, the curve has almost the same number of maximal and minimal cuts and the sum of all the cuts c_i is bounded by n, i.e. $\sum_{i=0}^{m} c_i = O(n)$.

If now the condition is relaxed, i.e. a cut is allowed to have an adjacent cut on the same ray R_{i+1} so that the curve loops around

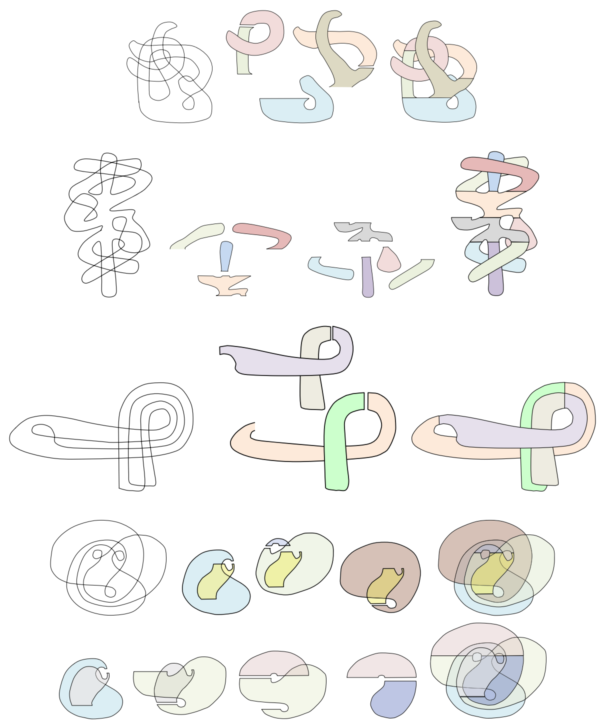


Fig. 17. Immersion of self-overlapping curves. In each row (except the last), the leftmost figure is the given curve and the rightmost figure shows the assimilation of all the simple curves (with color coded interiors) to form the overall curve. The last two rows show two different immersions (out of five) of a curve called Bennequin's disk.

in the segment S_i, the ray R_{i+1} has at least two more cuts than R_i (Fig. 16). So, if the number of cuts on R_i is c_i, then that on R_{i+1} is $d_i \pm 2l$, where $d_i = c_i$ or $d_i = c_i \pm 2$, and l is the number of loops of the curve in the concerned segment S_i.

The total number of cuts, N, is then given by $N = \sum_{i=0}^{m}(c_i+2l_i)$, where $c_{i+1} = c_i \pm k$, $k \in [0, 2]$ and $c_0 = p$. Here c_i is the number of cuts in S_i and l_i is the number of times the curve loops around in S_i. Simplifying,

$$N = \sum_{i=0}^{m} c_i + \sum_{i=0}^{m} 2l_i.$$

The sum of all the l_is is of $O(n)$ as there are n points on the curve. Hence, the total number of cuts, N, is of $O(n)$ and the total number of cut pairs is of $O(n^2)$ on an average. Thus, the average run time of the immersion algorithm is also $O(n^2)$.

4.3. Results and discussion

Fig. 17 shows the result of applying the immersion algorithm on a few self-overlapping curves.

The rays drawn need not be horizontal or even parallel to one other as long as they do not intersect one another. The direction of the rays can also be completely arbitrary as long as the cuts are properly assigned positive or negative polarities.

The immersion algorithm is also capable of detecting non self-overlapping curves in which case the validity tree will not have a continuous path of valid cut pairs from the root to a leaf node.

The number of crest points to be considered may be limited to alternate maximal and minimal crest points. If two consecutive crest points are both maximal or minimal either one can be considered in the immersion algorithm (Fig. 18).

The crest points are defined in terms of the horizontal vector. However, an equivalent definition with respect to any other vector

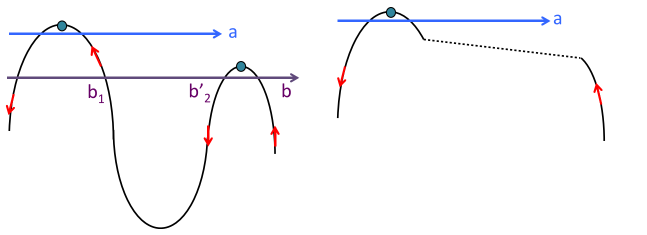


Fig. 18. Reduction of the number of crest points. By altering the left curve to resemble the right one, the crest point and hence the ray b can be eliminated without affecting the immersion algorithm.

is possible. Also, even though the analysis of the algorithm and the figures shown are self-overlapping polygons, the algorithm works for any general self-overlapping curves.

Applications like shape morphing require a constrained Delaunay triangulation of the interior of a self-overlapping curve, a requirement not guaranteed fulfilled by prior art. The immersion algorithm presented in this paper divides a self-overlapping curve into a set of simple ones each of which can be triangulated using any known techniques of simple polygon triangulation, and the triangulations of each can be glued to obtain an overall triangulation of the curve interior, thereby making it suitable for such applications.

Acknowledgments

I would like to thank Professor M. Gopi for introducing me to this problem and his continued support. I would also like to thank Shan Jiang for helping me prepare this paper.

References

[1] Shor PeterW, Van Wyk ChristopherJ. Detecting and decomposing self-overlapping curves. Computational Geometry 1992;2(1):31–50.
[2] Guo Xiaofeng, Hansen Pierre, Zheng Maolin. Boundary uniqueness of fusenes. Discrete Applied Mathematics 2002;118(3):209–22.
[3] Mukherjee Uddipan, Gopi M. Tweening boundary curves of non-simple immersions of a disk. In: Proceedings of the eighth Indian conference on computer vision, graphics and image processing. New York (NY, USA): ACM; 2012. 36:1–8.
[4] Mukherjee Uddipan, Gopi M, Rossignac Jarek. Immersion and embedding of self-crossing loops. In: Proc. ACM/eurographics symp. on sketch-based interfaces and modeling. 2011. p. 31–8.
[5] Brinkmann Gunnar, Delgado-Friedrichs O, Von Nathusius U. Numbers of faces and boundary encodings of patches. Graphs and Discovery 2005;27–38.
[6] Graver JE. The (m, k)-patch boundary code problem. MATCH Communications in Mathematical and in Computer Chemistry 2003;189–96.
[7] Titus CharlesJ. The combinatorial topology of analytic functions of the boundary of a disk. Acta Mathematica 1961;106(1–2):45–64 (English).
[8] Blank SJ. Extending immersions of the circle, Ph.D. dissertation. Brandeis University; 1967.
[9] Eppstein David, Mumford Elena. Self-overlapping curves revisited. In: ACM–SIAM symposium on discrete algorithms. 2009. p. 160–9.