Computers & Graphics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers & Graphics

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

SMI 2011: Full paper

Efficient convex hull computation for planar freeform curves

Yong-Joon Kim [a], Jieun Lee [b], Myung-Soo Kim [a,*], Gershon Elber [c]

[a] School of Computer Science and Engineering, Seoul National University, Seoul 151-744, Republic of Korea
[b] School of Computer Engineering, Chosun University, Gwangju 501-759, Republic of Korea
[c] Computer Science Department, Technion, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 10 December 2010
Received in revised form
13 March 2011
Accepted 15 March 2011

Keywords:
Convex hull
Planar freeform curves
Biarcs
Efficient algorithm
Culling
Upper envelope

ABSTRACT

We present an efficient real-time algorithm for computing the precise convex hulls of planar freeform curves. For this purpose, the planar curves are initially approximated with G^1-biarcs within a given error bound ε in a preprocessing step. The algorithm is based on an efficient construction of approximate convex hulls using circular arcs. The majority of redundant curve segments can be eliminated using simple geometric tests on circular arcs. In several experimental results, we demonstrate the effectiveness of the proposed approach, which shows the performance improvement in the range of 200–300 times speed up compared with the previous results (Elber et al., 2001) [8].

1. Introduction

Geometric algorithms for convex objects are more efficient than those for non-convex ones [5,9,19]. Thus convex hulls can be used to get a good initial solution for the general problem with non-convex objects. For example, in computing the minimum distance between two objects under continuous motion, their convex hulls can play an important role in making the algorithm efficient.

The minimum distance between two objects is often realized between their convex hulls. As long as the min-distance points between the convex hulls stay on the boundary curves of the two objects (but not on the lids of concavities), the distance is guaranteed to be the minimum between the two original objects (Fig. 1(a)). Even if one min-distance point falls onto the lid of a pocket, the location provides a good initial solution for the problem.

In Fig. 1(b), the upper object has its min-distance point inside the concavity bounded by a lid, i.e., a supporting bitangent line. Note that the min-distance between the convex hulls is realized on the lid and its end points $\mathbf{p}_l$ and $\mathbf{p}_r$ provide good initial solutions for starting the min-distance search inside the concavity. The footpoint $\mathbf{p}_0$ on the lid is closer to the left end point $\mathbf{p}_l$; thus we may start the search from $\mathbf{p}_l$ to the right. At the same time, on the boundary of the lower object, we start the search from the min-distance point $\mathbf{q}_0$ of the convex hull and proceed to the left. The first pair of boundary points that share the same normal line produces the min-distance points $\mathbf{p}_1$ and $\mathbf{q}_1$ in this example. This is not always the case. Nevertheless, their distance provides a tight upper bound for the minimum distance, which can greatly accelerate the search for other candidates.

An efficient and robust computation of the convex hulls for freeform curves and surfaces is still a challenging task in geometric modeling. There are a few implemented convex hull algorithms for freeform curves and surfaces [8,21]. They are based on solving a system of multivariate equations [7,14]. Thus it was difficult to achieve a real-time performance using these approaches. In this paper, we present a considerably more efficient and robust algorithm, which is based on the construction of an approximate convex hull boundary with circular arcs and line segments. For this purpose, we assume that the given planar freeform curves are initially approximated with tightly fitting G^1-biarcs within a given error bound ε in a preprocessing stage, which greatly simplifies the elimination of many redundant curve segments from the convex hull computation.

There is a certain price one has to pay for the extra space for storing the biarcs. But almost all real-time algorithms employ certain pre-built data structures, such as BVH, which can be quite large in general [2]. As long as we have a variety of useful geometric algorithms employing the G^1-biarc approximation, the cost can be amortized among them and the usage of extra space can be justified. Though it is only a recent development,

* Corresponding author. Tel.: +82 2 880 1838.
E-mail address: mskim@snu.ac.kr (M.-S. Kim).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.03.028

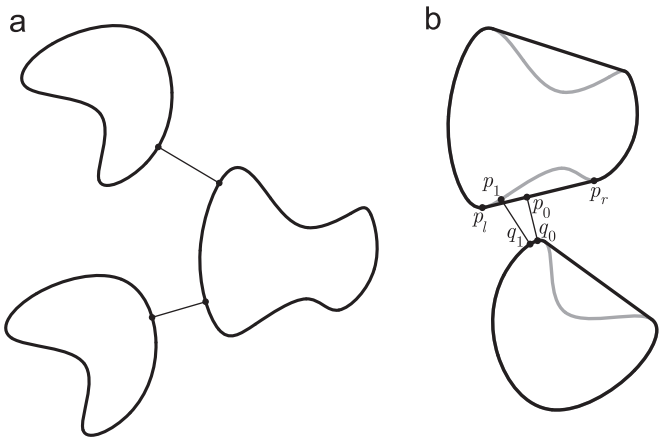

Fig. 1. Minimum distance computation: (a) the minimum distance between the convex hulls is identical to the distance between the two objects and (b) the min-distance points between the convex hulls provide a good initial solution for searching the min-distance points.

Aichholzer et al. [1] greatly simplified the medial axis transformation of planar freeform curves using the G^1-biarc approximation. Kim et al. [16] employed the G^1-biarcs for a real-time Hausdorff distance computation for planar freeform curves. There is potentially a huge number of important geometric applications that can benefit from adapting this approach.

The main contribution of this work can be summarized as follows:

- A simple technique is introduced that can eliminate the majority of redundant planar curves from the convex hull computation using simple geometric tests for circular arcs.
- Measuring the distance from an approximate convex hull boundary (consisting of circular arcs and line segments), we can trim off all curve segments which are more than distance ε away from the boundary. The distance test can be replaced by a simpler and sufficient condition test of checking whether the biarcs are more than distance 2ε away from the boundary.
- The implemented algorithm can compute the convex hull of planar freeform curves more than 200–300 times faster than the conventional algorithms [8].

The rest of this paper is organized as follows. In Section 2, we briefly review the previous work on the convex hull computation. Section 3 presents the basic idea of our approach by developing a precise convex hull algorithm for planar freeform curves. Section 4 shows how the performance improvement has been made over the previous algorithms by using experimental results. Finally, in Section 5, we conclude this paper with discussions on future work.

2. Related work

We briefly review related work on the convex hull computation. Early algorithms considered only discrete objects such as points, lines, polygons, or polyhedra [3,5,10,11,17–19]. Some theoretical algorithms were developed for planar curved regions [4,6,20]; however, they assumed certain curve operations which are difficult to implement in an efficient and robust way. In particular, the algorithms are based on rather expensive curve operations such as common tangent computation for pairs of curve segments. However, many of the common tangent lines may not appear in the final output of the convex hull boundary and thus their computation is wasted.

More practical solutions were reported with full implementation in recent results [8,15,21]. Elber et al. [8] presented a robust implementation for the convex hull of planar curves. Seong et al. [21] extended the result to the convex hull computation for freeform space curves and freeform surfaces. Their approach is based on the characterization that a curve/surface point is on the convex hull boundary if the tangent line/plane at the point intersects the curve/surface at no other points except the tangent point itself. This geometric constraint has been formulated as a system of polynomial equations. Nevertheless, it is quite difficult to implement the algorithm for real-time performance since this approach entails a large number of subdivisions of the multivariate B-spline representations.

In a recent work, Aichholzer et al. [1] proposed an efficient and robust algorithm for computing the medial axis for planar freeform shapes using G^1-biarc approximation of the planar curves. Motivated by this approach, and employing the hardware-based method of Hoff et al. [12] for a fast computation of the distance map for the G^1-biarcs using the graphics depth-buffer, Kim et al. [16] proposed a real-time computation technique for the Hausdorff distance between two planar freeform curves. In the current paper, without using any graphics hardware capability, we introduce a new real-time algorithm for the convex hull computation for planar freeform curves using precomputed G^1-biarc approximations to the planar curves.

3. Convex hull for planar freeform curves

In this section, we present an efficient algorithm for computing the convex hull of planar freeform curves with G^1-biarc approximation.

3.1. Preprocessing for G^1-biarc approximation

The planar freeform curves are first segmented into x and y-monotone pieces and further subdivided at each inflection point. Each convex monotone curve segment is then approximated by G^1-biarcs quite tightly within a given error bound $\varepsilon > 0$ by recursively subdividing the curve segment. Each biarc interpolates the curve end points and the tangent directions at the end points, using two G^1 circular arcs [22]. Fig. 2 shows an example of

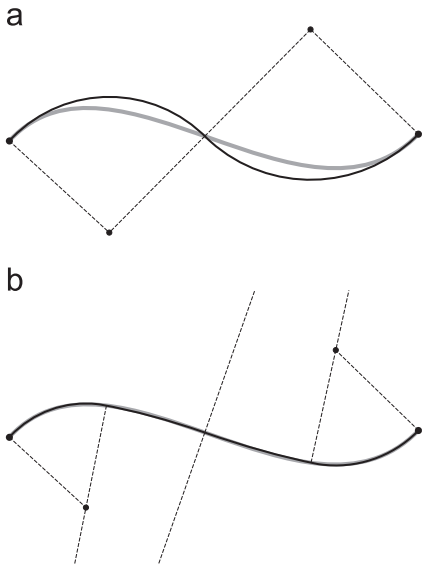

Fig. 2. Biarc approximation for a cubic Bézier curve: (a) one biarc and (b) two biarcs.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

biarc approximation for a cubic Bézier curve, where we start with one biarc in Fig. 2(a) and improve to a much tighter approximation of Fig. 2(b) in just one step refinement.

There are many different algorithms for computing the biarc approximation. In this paper, we use the algorithm of Šír et al. [22]. Given a planar curve segment $C(t)$, for $0 \le t \le 1$, with two end points $\mathbf{p}_0$ and $\mathbf{p}_1$ and two unit tangent directions $\mathbf{u}_0$ and $\mathbf{u}_1$ to be interpolated by a biarc (i.e., a pair of G^1 arcs meeting at a joint), the locus of all possible locations of the joint $\mathbf{j}$ forms a circle O passing through the two end points $\mathbf{p}_0$ and $\mathbf{p}_1$ and having its center at the intersection of two bisector lines: one between $\mathbf{p}_0$ and $\mathbf{p}_1$, and the other between $\mathbf{p}_0+\mathbf{u}_0$ and $\mathbf{p}_1+\mathbf{u}_1$ [22]. The circle O intersects the given curve $C(t)$ at least at one point $\mathbf{j}$, which can be taken as a joint for the biarc construction as shown in Fig. 3.

The construction algorithm of Šír et al. [22] has an important advantage of having the biarc joint $\mathbf{j}$ located on an interior point of the curve. This property greatly simplifies the parameter matching between the given curve and its component arcs in the biarc approximation, since each arc has its two end points located on the given curve. By measuring the maximum deviation of the curve from the arc center, we can easily compute the Hausdorff distance between the arc and the curve segment.

3.2. Efficient elimination of redundancies

To speed up the construction of an approximate convex hull using the biarcs, we apply a simple throw-away technique, which is similar to the Akl-Toussaint approach [3]. Given a set of n points in the plane, Akl and Toussaint [3] find four extreme points along the x and y-directions which are guaranteed to be on the convex hull boundary. We can then throw away all points inside the quadrilateral formed by the four points.

A slight difference is that we employ 8 test directions instead of 4. Elber et al. [8] also used a similar interior culling technique but with 16 normal directions as shown in Fig. 4, since the performance of Elber et al. [8] is better when using 16 directions instead of 8. To check the redundancy of a curve segment, the control points are tested for the containment in the convex polygon determined by 16 extreme points along the uniform normal directions.

In our case, since we have tightly fitting circular arcs, the culling test can be carried out significantly more efficiently. Instead of testing the containment in the convex polygon with eight edges (Fig. 5(a)), we need to check each circular arc against only two edges. For example, given a convex monotone circular arc:

$$C(\theta) = (x_0, y_0) + r(\cos\theta, \sin\theta), \quad \text{for } 0 \le \theta \le \frac{\pi}{2},$$

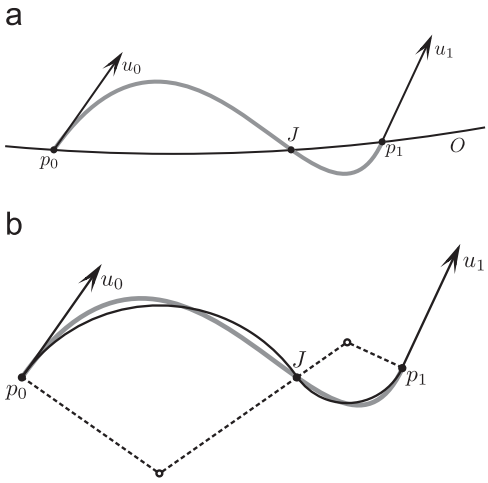

Fig. 3. Biarc construction: (a) the locus of all possible joints $\mathbf{j}$ forms a circle O that intersects the given curve at a point $\mathbf{j}$, and (b) a pair of G^1 arcs interpolating the end conditions and meeting at the joint $\mathbf{j}$.

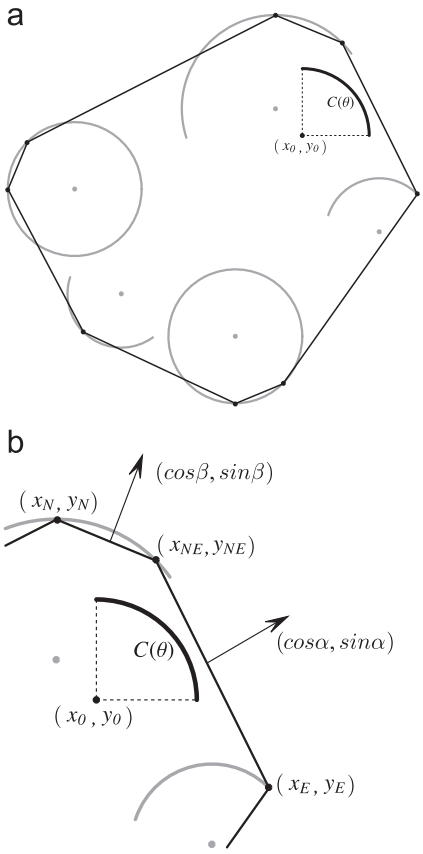

Fig. 5. Interior arc culling: (a) an arc is redundant if it is contained inside a convex polygon formed by 8 extreme points and (b) a test against only two edges is sufficient for the interior culling.

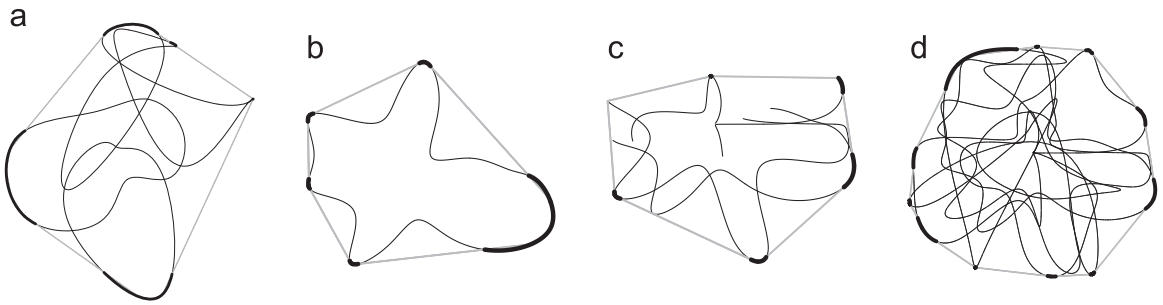

Fig. 4. Interior culling and clipping of the Bézier curves using the convex hull of 16 extreme points.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

which faces the north eastern directions, we test this arc against two lines of the convex polygon that face the same directions (Fig. 5(b)). The two lines meet at an extreme point (x_{NE},y_{NE}) and their normal angles are $\alpha,\beta \in [0,\frac{\pi}{2}]$. This arc is redundant if the distance from the center (x_0,y_0) to each line is larger than the radius r:

$$(x_{NE}-x_0)\cdot\cos\alpha+(y_{NE}-y_0)\cdot\sin\alpha > r,$$

$$(x_{NE}-x_0)\cdot\cos\beta+(y_{NE}-y_0)\cdot\sin\beta > r.$$

In the above test, we could have compared the distance to the lines with a slightly larger value $r+2\varepsilon$. If this stronger condition is met for two adjacent circular arcs in a G^1-biarc, the corresponding curve segment should be completely contained in the interior of the precise convex hull and consequently it is redundant from the perspective of convex hull computation. Thus in the proceeding discussions, we assume these interior curve segments have been eliminated at this stage of interior culling. The end points of the original input curves are considered as circular arcs of radius zero with normal angular range π. The redundancy of these end points can be tested by checking the containment in the interior of the convex polygon with eight edges.

3.3. Approximate convex hull boundary

Using the reduced set of circular arcs, we compute an approximate convex hull boundary that consists of circular arcs and line segments.

For each remaining circular arc C_i, $i=1,\ldots,n$, with center (x_i,y_i), radius r_i, and normal angles θ, for $\underline{\theta}_i \le \theta \le \overline{\theta}_i$, we take a function of the angle θ:

$$d_i(\theta)=\begin{cases} r_i+x_i\cos\theta+y_i\sin\theta, & \text{if } \underline{\theta}_i \le \theta \le \overline{\theta}_i,\\ 0, & \text{otherwise,}\end{cases}$$

which measures the signed distance from the origin (0,0) to the tangent lines of C_i along the normal direction $(\cos\theta,\sin\theta)$. Now we consider the upper envelope of these functions:

$$\mathcal{U}(\theta)=\max_i d_i(\theta),$$

which corresponds to the circular $C_i(\theta)$ that appear on the approximate convex hull boundary with the normal angle θ. (Appendix A has more details on the relationship between this upper envelope function and the convex hull computation.) Note that the distance d_i itself is coordinate dependent; nevertheless, the relative distance between $d_i(\theta)$ and $d_j(\theta)$ is coordinate independent. The translation and rotation of one coordinate system to another will add the same constant to both d_i and d_j and add/subtract the same rotation angle α to/from θ. Thus the relative arrangement of $d_i(\theta)$'s should be invariant.

Fig. 6 shows how the convex hull computation can be carried out for the most complicated example of Fig. 4(d). In Fig. 6(a), a convex polygon connecting 8 extreme points are shown together with the G^1-biarc approximation for the input planar curves of Fig. 4(d). The result of interior culling (based on two polygon edges facing the same range of normal directions) is shown in Fig. 6(b). Note that this result is better than the containment test inside the convex polygon. There are some isolated arcs outside the convex polygon and their adjacent arcs are eliminated even though not completely inside the convex polygon. This is because our test has eliminated some arcs because their angle ranges do not match with the normal directions of the nearby edges of the convex polygon. For example, when a circular arc, with normal angles $\pi/2 \le \theta \le \pi$, is located outside the polygon edges facing the north eastern directions, there is no chance for this arc to contain any point extremal to one of its normal directions $\pi/2 \le \theta \le \pi$. Thus we can eliminate this arc.

For the case of this complicated example, the performance can be improved by considering additional 8 normal directions. Thus each of the remaining arcs is tested against two additional edge directions, the result of which is shown in Fig. 6(c). The upper envelope of Fig. 6(d) consists of different parts from the distance functions $d_i(\theta)$. Note that each intersection point between two adjacent functions d_i and d_j on the upper envelope corresponds to a common tangent line $\overline{C_i(\theta)C_j(\theta)}$ between two corresponding circular arcs. There are also some redundant intersection points below the upper envelope. The redundancy of an intersection point at the parameter θ can easily be tested by checking if there is another function value $d_i(\theta)$ above it. Thus a robust construction of the upper envelope can be done in a straightforward manner.

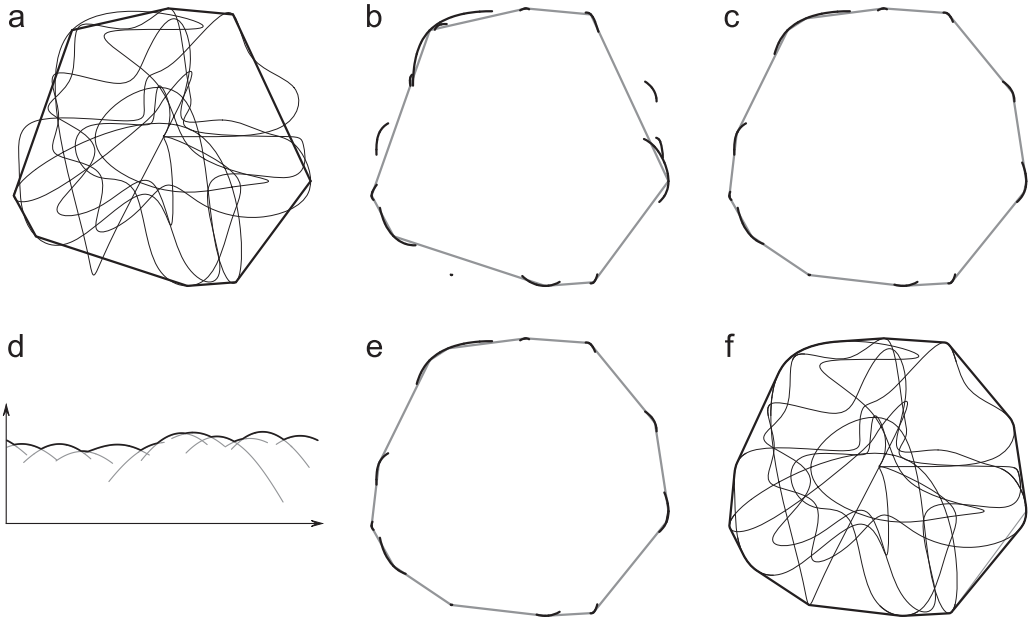

Fig. 6. The convex hull computation for planar freeform curves: (a) the convex polygon with 8 extreme points, (b) interior culling, (c) further culling with 16 normal directions, (d) the upper envelope, (e) the remaining curve segments for the convex hull computation, and (f) the precise convex hull.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

3.4. Convex hull of planar freeform curves

The remaining curve segments from the original curves are shown in Fig. 6(e). We concatenate connected consecutive segments into longer convex curves so that a pair of such curves can admit at most one common tangent line. Now the arrangement of the distance functions for these convex freeform curves should be almost the same as the one shown in Fig. 6(d). Thus a robust construction of the precise convex hull boundary should be quite straightforward.

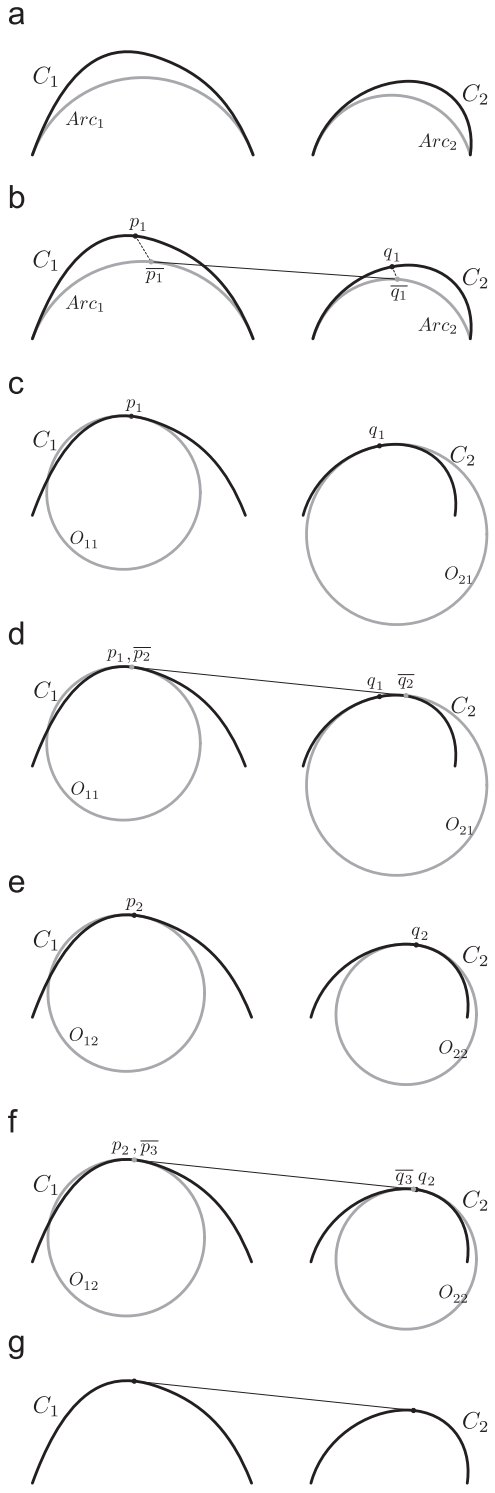

Fig. 7. Common tangent computation using osculating circles.

Common tangent lines are numerically computed using osculating circles to the convex curves as shown in Fig. 7. The initial solution is taken from the circular arcs sharing a common tangent line on the approximate convex hull boundary (Figs. 7(a)–(b)). We adapt the circle-based point-projection technique of Hu and Wallner [13] to our current problem. In Fig. 7(b), using the common tangent points on the circular arcs, we guess the common tangent locations on the curve segments. Then the osculating circles are computed at the locations (Fig. 7(c)), and a common tangent line is computed for the two osculating circles (Fig. 7(d)). The common tangent points are projected back to the curve segments (Fig. 7(e)), and the same procedure is repeated until there are no considerable changes in the tangent locations (Figs. 7(e)–(g)). The precise convex hull thus computed is shown in Fig. 6(f).

4. Experimental results

We have implemented our convex hull algorithm in C++ on an Intel Core i7-2600 3.4 GHz PC with an 8 GB main memory and an NVIDIA GeForce GTX 570 GPU. To demonstrate the effectiveness of our approach, we have tested the algorithm for the four freeform curves shown in Fig. 4, which were also used in Elber et al. [8].

Table 1 shows the relative performance of our algorithm over the algorithm of Elber et al. [8] as implemented in the IRIT solid modeling system [14]. The panels correspond to the four examples of Fig. 4. In each subtable, the first column shows the error bound for the biarc approximation. The second column under "Arcs" reports the total number of circular arcs in the biarc approximation of the planar freeform curves within the given error bound. The original number of Bézier curve segments is shown (in boldface) in the last row corresponding to the implementation of Elber et al. [8] in the IRIT system [14].

The column under "Cull" reports the percentage of the total arclength of planar curve segments that have been culled/clipped away using the convex polygon generated by 8 or 16 extreme

Table 1
Experimental results for the four examples shown in Fig. 4; the number of Bézier curves in each example is given (in boldface) in the last row of the subtable.

	Arcs	Cull (%)	#Arc	CH	Agen	Prep
(a)						
0.0100	122	89	27	0.06	0.06	1.4
0.0050	166	92	34	0.08	0.08	1.9
0.0010	308	93	62	0.13	0.14	3.0
0.0005	410	94	81	0.14	0.19	3.6
IRIT	**24**	96		23.5		
(b)						
0.0100	82	86	29	0.03	0.03	0.9
0.0050	98	87	33	0.05	0.05	1.2
0.0010	182	92	58	0.08	0.08	1.9
0.0005	232	93	69	0.09	0.11	2.2
IRIT	**18**	93		31.7		
(c)						
0.0100	142	94	22	0.06	0.06	1.7
0.0050	166	95	22	0.08	0.09	2.2
0.0010	314	97	39	0.11	0.15	3.4
0.0005	424	97	48	0.14	0.19	4.4
IRIT	**33**	98		17.8		
(d)						
0.0100	420	96	37	0.20	0.19	4.9
0.0050	532	97	41	0.22	0.25	6.4
0.0010	948	98	68	0.34	0.44	8.4
0.0005	1284	98	81	0.44	0.59	10.8
IRIT	**90**	98		87.3		

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

points. The optimal 100% corresponds to the total length of redundant curve segments inside the convex hull. Thus for a closed convex curve, this quantity is simply undefined.

The best performance of Elber et al. [8] is obtained when using 16 normal directions. On the other hand, our performance is optimal when using 8 directions, for three examples of Fig. 4(a)–(c), and when using 16 directions, for the most complicated example of Fig. 4(d). Thus we have employed 16 directions for Elber et al. [8]. In the case of our current algorithm, we start with 8 directions and check if the remaining circular arcs have the total sum of their normal angle ranges larger than 4π. If it is, we consider 8 additional directions. Thus, for the example of Fig. 4(d), a total of 16 directions have been employed.

The column under "# Arc" reports the number of circular arcs remaining after the interior culling stage. The time (measured in ms) taken in the convex hull computation is shown in the column under "CH". The storage for all the circular arcs is relatively large compared with the total storage for the curve control points. A simple technique for reducing the data size for arcs is to store only the curve parameters corresponding to the arc end points. The column under "Agen" reports the time (in ms) taken in generating all the circular arcs from the curve parameters. It takes about the same time as the convex hull computation itself.

In this paper, we assume that the biarc approximation is given as a part of the geometric input data. Nevertheless, for a fair comparison, the last column under "Prep" reports the preprocessing time (in *ms*) for the biarc approximation. Taking this part into account, the convex hull construction is slowed down about 10 times. For an efficient generation of biarcs, we have developed a numerical technique that starts the search for a biarc joint from the curve mid-point $C(\frac{1}{2})$ and employs osculating circles to the planar curve so as to replace an expensive curve–circle intersection by a sequence of simple circle–circle intersections.

In Table 1, the best performance is realized at the lowest precision of 0.01, which is about 1% of the size of an axis-aligned bounding square for the planar freeform curves. Even at lower precisions, the overall performance still improves. Nevertheless, there occur more complications in the determination of the correct topology as the accuracy gets lower.

To demonstrate the real-time performance of our algorithm, we have computed the convex hull for two character strings "SMI" and "2011" under continuous animation while changing their relative position and orientation. Fig. 8 shows some snapshots from the animation, each snapshot with the convex hull bounding all the characters from the two strings. In this application where each component character undergoes a rigid-body motion, the precomputed biarc approximation can be used for the whole animation sequence.

5. Conclusions

In this paper, we have presented an efficient algorithm for computing the convex hull of planar freeform curves. Using the G^1-biarc approximation, we have shown that the precise convex hull boundary can be constructed in a very efficient and robust manner. In future work, we plan to extend the current result to a real-time convex hull algorithm for freeform surfaces and space curves. The main technical challenge in the extension is that arcs in space behave much like truncated tori. Moreover, it is extremely difficult to approximate freeform surfaces using truncated tori connected even with C^0 continuity.

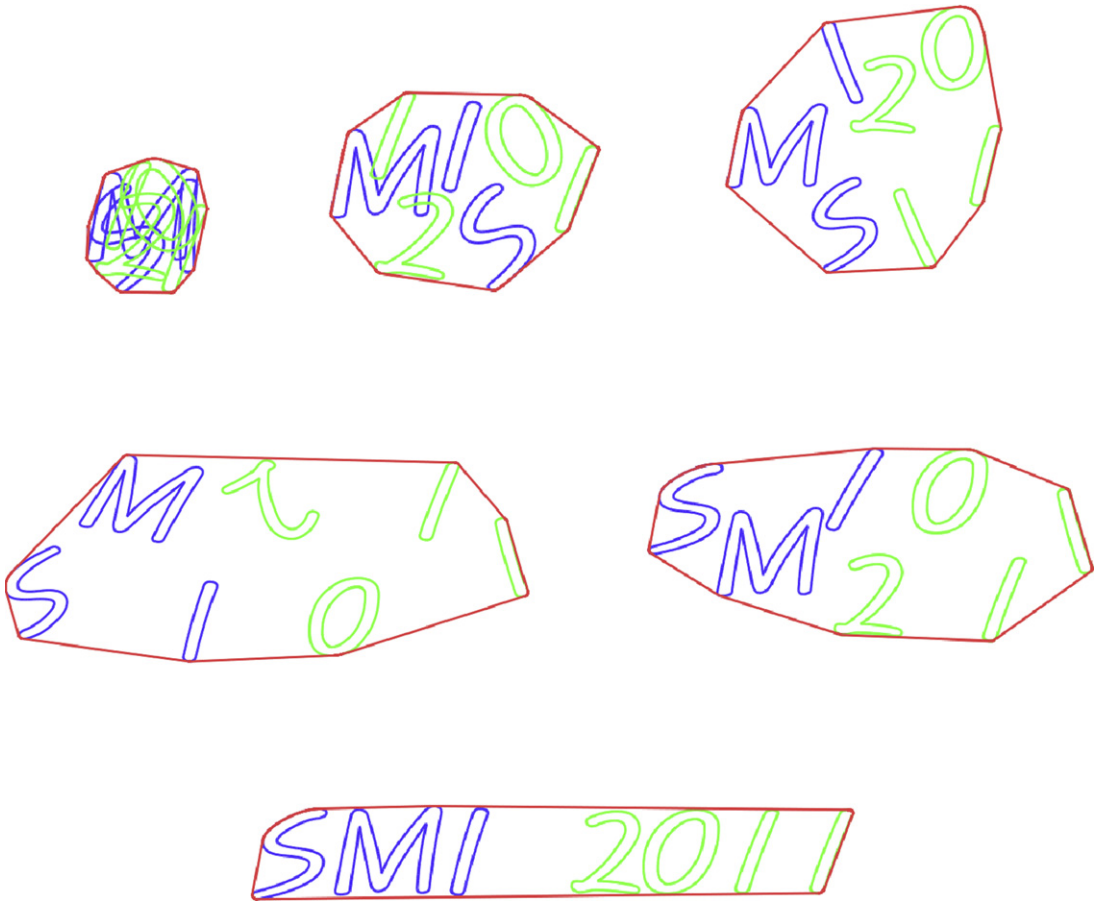

Fig. 8. Real-time convex hull computation for two character strings in animation.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

Acknowledgments

This research was supported in part by the Israeli Ministry of Science, Grant no. 3-38273 and in part by NRF, Research Grant (nos. 2010-0014351 and 2010-0005597).

Appendix A. Convex hull computation using support distance functions

Each point $\mathbf{p}$ on the boundary of a convex set C is characterized by an outward normal direction $\mathbf{n}$ to which the point $\mathbf{p}$ is extremal:

$$\langle \mathbf{p},\mathbf{n} \rangle \geq \langle \mathbf{q},\mathbf{n} \rangle \quad \text{or equivalently } \langle \mathbf{p}-\mathbf{q},\mathbf{n} \rangle \geq 0$$

for all $\mathbf{q} \in C$. For a convex curve $C(t)$, $0 \leq t \leq 1$, each curve point $C(t)$ is extremal to its outward normal direction $\mathbf{n}(t)$:

$$\langle C(t)-C(s),\mathbf{n}(t) \rangle \geq 0$$

for all $0 \leq s \leq 1$. Using this characterization, we can design an algorithm for computing the convex hull of planar freeform curves using the upper envelope of support distance functions defined by these curves.

Given a set of planar regular curves $C_i(t)$, $0 \leq t \leq 1$, $i=1, \ldots, n$, we consider the unit normal vectors $\mathbf{n}_i(t)$, $0 \leq t \leq 1$, $i=1, \ldots, n$, that point toward the convex side of $C_i(t)$. (Note that a regular curve point $C_i(t)$ can be extremal to no other direction than $\mathbf{n}_i(t)$.) We define a scalar function

$$d_i(t) = \langle C_i(t),\mathbf{n}_i(t) \rangle$$

for $0 \leq t \leq 1$ and $i=1, \ldots, n$. This function is called the *support distance* of the tangent line of $C_i(t)$ with a unit normal $\mathbf{n}_i(t)$.

The unit normal vector is well-defined

$$\mathbf{n}_i(t) = (\cos\theta(t),\sin\theta(t)) \in S^1.$$

We consider an upper envelope of the functions d_i parameterized by the angle θ:

$$\mathcal{U}(\theta) = \max_i d_i(t(\theta)) = \max_i \langle C_i(t(\theta)),\mathbf{n}_i(t(\theta)) \rangle.$$

For each normal direction $\mathbf{n} = (\cos\theta,\sin\theta)$, the curve index i and the curve parameter $t = t(\theta)$ that realize the upper envelope value $\mathcal{U}(\theta) = d_i(t(\theta))$ determine the curve point $C_i(t)$ that appear on the convex hull boundary and extremal in the normal direction $\mathbf{n}$. There may be multiple curve indices that share the same upper envelope value in the same normal direction $\mathbf{n}$, i.e.,

$$\mathcal{U}(\theta) = d_i(t(\theta)) = d_j(\bar{t}(\theta)) \quad \text{for some } i \neq j.$$

Then the common tangent line segment $\overline{C_i(t(\theta))C_j(\bar{t}(\theta))}$ is on the convex hull boundary and extremal in the normal direction $\mathbf{n}$.

Fig. 9 shows a simple example where the convex hull is computed for a set of three circles and two circular arcs in the plane. The upper envelope consists of five different parts from the support distance functions, where the first function is connected to the last one through an identical angle $\theta = 0 = 2\pi$. The color for each function $d_i(t(\theta))$ corresponds to the identity of each circle $C_i(t)$. Note that the intersection between two adjacent functions d_i and d_j on the upper envelope corresponds to a common tangent line $\overline{C_i(t(\theta))C_j(\bar{t}(\theta))}$ between two circles or circular arcs with the corresponding colors.

We can eliminate a curve segment $C_j(t)$, $t_1 \leq t \leq t_2$, as redundant from the convex hull computation when the support distance of the curve has no chance of reaching the upper envelope:

$$d_j(t(\theta)) < \mathcal{U}(\theta)$$

for $\theta_1 \leq \theta \leq \theta_2$, where $t(\theta_1) = t_1$ and $t(\theta_2) = t_2$.

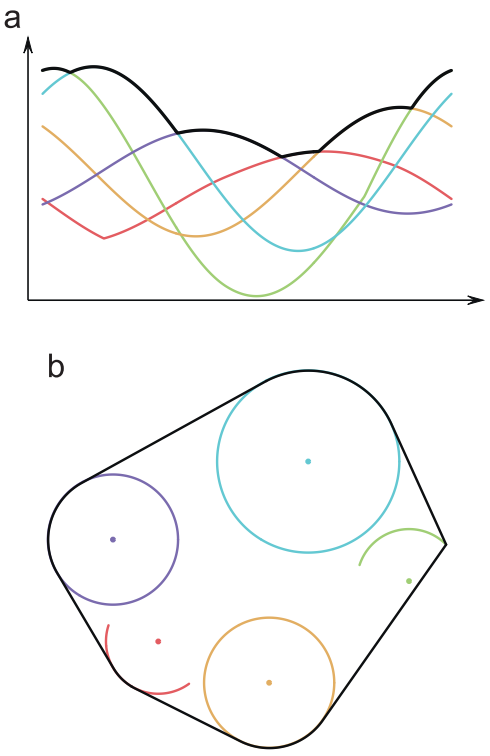

Fig. 9. The convex hull computation for a set of three circles and two circular arcs in the plane: (a) the upper envelope of five support distance functions $d_i(\theta)$ and (b) the convex hull corresponding to the upper envelope.

The construction of a precise upper envelope is expensive since one has to compute the inverse function $t(\theta)$. Thus we consider circles and circular arcs which can be directly parameterized by the angle θ. Let O_i, $i=1, \ldots, n$, denote n circles with center (x_i,y_i) and radius r_i:

$$(x_i + r_i\cos\theta, y_i + r_i\sin\theta) \quad \text{for } 0 \leq \theta \leq 2\pi.$$

Then we have

$$d_i(\theta) = r_i + x_i\cos\theta + y_i\sin\theta$$

and

$$\mathcal{U}(\theta) = \max_i d_i(\theta) \quad \text{for } 0 \leq \theta \leq 2\pi.$$

For circular arcs C_i, $i=1, \ldots, n$, with center (x_i,y_i), radius r_i, normal angles θ, for $\theta_0 \leq \theta \leq \theta_1$ with two end points $C_i(\theta_0) = (x_i^0,y_i^0)$ and $C_i(\theta_1) = (x_i^1,y_i^1)$, we take

$$d_i(\theta) = \begin{cases} r_i + x_i\cos\theta + y_i\sin\theta & \text{if } \theta_0 \leq \theta \leq \theta_1, \\ \max_{k=0,1}(x_i^k\cos\theta + y_i^k\sin\theta) & \text{otherwise.} \end{cases}$$

This means that the two end points are considered as circular arcs of radius 0. Now the upper envelope $\mathcal{U}(\theta)$ is taken as the maximum of $d_i(\theta)$ for all circles and circular arcs.

References

[1] Aichholzer O, Aigner W, Aurenhammer F, Hackl T, Oberneder M, Jüttler B. Medial axis computation for planar free-form shapes. Computer-Aided Design 2009;41(5):339–49.
[2] Akenine-Moller T, Hains E, Hoffman N. Real-time rendering. 3rd ed. A.K. Peters; 2008.
[3] Akl S, Toussaint G. Efficient convex hull algorithms for pattern recognition applications. In: Proceedings of the fourth international joint conference on pattern recognition, Kyoto, Japan; November 7–10, 1978. p. 483–8.
[4] Bajaj C, Kim M-S. Convex hulls of objects bounded by algebraic curves. Algorithmica 1991;6(4):533–53.
[5] de Berg M, van Kreveld M, Overmars M, Schwarzkodf O. Computational geometry: algorithms and applications. Berlin: Springer-Verlag; 1997.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028

[6] Dobkin D, Souvaine D. Computational geometry in a curved world. Algorithmica 1990;5(3):421–57.
[7] Elber G, Kim M-S. Geometric constraint solver using multivariate rational spline functions. In: Proceedings of ACM symposium on solid modeling and applications, Ann Arbor, MI; June 4–8, 2001.
[8] Elber G, Kim M-S, Heo H-S. The convex hull of rational plane curves. Graphical Models 2001;63(3):151–62.
[9] Gilbert E, Johnson D, Keerthi S. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Transactions on Robotics and Automation 1988;4(2):193–203.
[10] Graham R. An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters 1972;1(4):132–3.
[11] Graham R, Yao F. Finding the convex hull of a simple polygon. Journal of Algorithms 1983;4(4):324–31.
[12] Hoff K, Culver T, Keyser J, Lin MC, Manocha D. Fast computation of generalized Voronoi diagrams using graphic hardware. In: Proceedings of the SIGGRAPH 99, Computer graphics annual conference series; 1999. p. 277–86.
[13] Hu S-M, Wallner J. A second order algorithm for orthogonal projection onto curves and surfaces. Computer-Aided Geometric Design 2005;22(3):251–60.
[14] IRIT 10.0 user's manual, Technion 〈http://www.cs.technion.ac.il/~irit〉; 2009.
[15] Johnstone J. A parametric solution to common tangents. In: Proceedings of international conference on shape modeling and applications. Genova, Italy; May 7–11, 2001.
[16] Kim Y-J, Oh Y-T, Yoon S-H, Kim M-S, Elber G. Precise Hausdorff distance computation for planar freeform curves using biarcs and depth buffer. The Visual Computer 2010;26(6–8):1007–16.
[17] Lee DT. On finding the convex hull of a simple polygon. International Journal of Computer and Information Sciences 1983;12(2):87–98.
[18] Preparata F, Hong S. Convex hulls of finite sets of points in two and three dimensions. Communications of ACM 1977;20(2):89–93.
[19] Preparata F, Shamos M. Computational geometry: an introduction. New York, NY: Springer-Verlag; 1985.
[20] Schäffer A, van Wyk C. Convex hulls of piecewise-smooth Jordan curves. Journal of Algorithms 1987;8(1):66–94.
[21] Seong J-K, Elber G, Johnstone JK, Kim M-S. The convex hull of freeform surfaces. Computing 2004;72(1–2):171–83.
[22] Šír Z, Feichtinger R, Jüettler B. Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics. Computer-Aided Design 2006;38(6):608–18.

Please cite this article as: Kim Y-J, et al. Efficient convex hull computation for planar freeform curves. Computers and Graphics (2011), doi:10.1016/j.cag.2011.03.028