www.cadanda.com
Computer-Aided Design AND APPLICATIONS™

Determining curves in convex hull from a set of planar closed convex curves

Vishwanath A.V. and Ramanathan M.
Department of Engineering Design,
Indian Institute of Technology Madras, Chennai, India
{vishwanathavin, emry01}@gmail.com

ABSTRACT

Convex hull, a minimum enclosing convex envelope is a popular construction in the field of computational geometry. Typical convex hull include not just the points in the hull but also the connectivity between them. However, at times, it may be sufficient to get only points contributing to the hull, which can then be employed in constructions such as positive α-hull. Algorithms for convex hull primarily focus on point-set as input. Very few algorithms handle input curves as such, without discretization. In this paper, algorithm for determining curves that belong to convex hull using minimum spanning tree (MST) is proposed. The edges of the MST can be considered as a rubber-band connecting all the curves with zero enclosed area. Valency of nodes in the MST is used to define curve triplets. Maximum inscribed circle (MIC) for all triplets is identified (and stored in a triplet matrix) and its minimum is then picked to identify the starting triplet. The triplet is then deleted, updating the triplet matrix. The deletion of the triplet is akin to leaving out curves in the interior to the convex hull. Updating a triplet matrix emulates the pushing out the rubber-band (edges of MST) and moving towards the curves in the convex hull. Repeating the process of deleting a triplet and updating the triplet matrix will eventually form pairs of curves that lie in the convex hull. Connectivity information is then derived using re-ordering. Results are presented that show curves belonging to convex hull.

Keywords: convex hull, freeform curves, minimum spanning tree, maximum inscribed circle.
DOI: 10.3722/cadaps.2013.xxx-yyy

1 INTRODUCTION

Convex hull [17, 5] of a set is defined as a minimal area convex enclosure of the set. Quite a few algorithms exist for computing the convex hull of a point set [17], both in R^2 as well as in R^3. Convex hull has found numerous applications, ranging from interference checking [15] to shape matching [4].

Convex hull typically imply that it consists of not just few elements from the set but also polygon (consisting of both points and connecting edges). Moreover, in some applications such as positive α-hull, only points on the convex hull play a role and hence it may be sufficient to determine them.

Problems such as convex hull are usually considered for a set of points, and belong to the filed known as computational geometry. Fields such as CAD, geometric modelling which deal typically with domains like closed curves and surfaces are now being used as input to study such problems. Curved inputs have been used in various forms. For example, splinegons, where the straight edges in the polygons are replaced by curved edges have been employed (for eg. [13] studied the art-gallery problem and whereas [20] computed the medial axis). Single non-convex closed curve without discontinuities [3] (Voronoi diagram) as well as a set of such curves (examples, convex hull [8], minimum enclosing hypershpere [16], minimum enclosing ellipse [2], smallest enclosing ball of a set of balls [11]) have also been considered. Few other works used the input which can be termed as pseudo-circles - a set of disjoint convex curves having no discontinuities. Computation of Vornoi cell for such input has been presented in [12], Voronoi diagram for ellipses in [10], Delaunay graph for ellipses [9], and Visibility graph [18].

Inputs such as curves also call for different kind of approach for the same problem for a set of points. For example, for the computation of convex hull of a set of curves, tangents and bi-tangents computations are required [8]. Similarly, computations of visibility graph [18], shortest path [19] etc. also requires tangents and bi-tangents for a set of curves, which is not the case for a set of points. The computations are typically much more numerically intensive than its point-set counterpart.

To the best of the knowledge of the authors, very few algorithms compute the convex hull for a set of curves represented exactly (i.e. without approximating the curves using sample points). An algorithm for computing the convex hull of a set of freeform curves has been provided in [8] which was then extended to freeform surfaces [21].

It has been shown that the boundary of Delaunay triangulation results in convex hull. However, it should be noted that, though the theoretical foundation as well as algorithms for Voronoi diagram of a set of freeform curves has been well-developed in the recent past (for example, please refer to [12] for computing Voronoi cell), computing Voronoi diagram for such input is still a topic of active research. Moreover, Delaunay triangulation for a set of curves is not well known (under the condition that the curves are not discretized into set of points).

One approach to find the convex hull for a set of curves is to generate set of points on the curves and then use an existing algorithm. However, this approach may result in a very coarsely approximated hull (depending on the sampling on each curve) of the input curves which might impede the accuracy of the results (this argument has been shown to be true for algorithms such as Delaunay graph [10] and medial axis [20]). A bi-arc based approximation of the curves has shown to improve efficiency in computing the convex hull [14], though with an error bound.

Let S be a set of disjoint free-form (parametric) convex curves (curves without inflection points) with no straight line portions and having no discontinuities in R^2. It is to be noted that as the closed curves have well defined exterior and interior unlike that of points. It is assumed that the interior of a closed curve lies to its left as we travel along the increasing direction of parametrization.

In this paper, an algorithm for computing the curves that belong to the convex hull from the set S, has been presented. The algorithm starts from a triplet of curves (say C1 (t1), C2 (t2), and C3 (t3) having counter clockwise direction of parametrization. Maximal inscribed circle (MIC), which is the maximum radius circle that touches all the curves and not containing any of them is employed. Using constraint equations (Equations (1) and (2)) and then the radius of MIC (RMIC) is first computed for a triplet. This triplet is then removed and updated. In general, at each iteration of the algorithm, one curve in left out from further computation. Constraint equations to identify where MIC lies is also formulated. It is to be noted that an algorithm for convex hull of a set of points need not return the points in the convex hull in an ordered manner [17]. In this paper, the output from the algorithm gives the connectivity information of the curves forming the convex hull of the set.

2 CONSTRAINTS

Definition 1 Bitangent (BT) - It is the line segment connecting a pair of points that is tangent to the curves C1(t) and C2(r), forming the end points of the line segment.

Definition 2 Touchpoint - It is the point on a curve at which a circle/disc of radius R touches the curve.

2.1 Constraint for three curves

Consider three curves $C_0(t)$, $C_1(r)$, and $C_2(s)$. For a circle to be touching the three curves (in this paper, it is assumed that each touchpoint are from distinct curves), the following constraints have to be satisfied;

$$\begin{aligned} \langle C_0'(t), P(x,y) - C_0(t) \rangle &= 0, \\ \langle C_1'(r), P(x,y) - C_1(r) \rangle &= 0, \\ \langle C_2'(s), P(x,y) - C_2(s) \rangle &= 0, \\ \|P(x,y) - C_0(t)\| &= \|P(x,y) - C_1(r)\|, \\ \|P(x,y) - C_1(r)\| &= \|P(x,y) - C_2(s)\|. \end{aligned}$$

(1)

where P(x, y) is the center of the disc and ′ denote the tangent at a point on the curve. First three equations denote that the circle is tangent to the curves and the last two denote that touch points are equidistant from the center. Constraints in Equation (1) will lead to finite set of solutions. A containment check is used to decide the validity of solution set that then determines the touchpoints.

2.2 Containment check: Is a curve outside the disc of radius *R*

The following constraint is used to determine if a curve C(t) is outside (Equation (2)) the disc

$$\|D_{min}(x,y) - P(x,y)\| >= R.$$

(2)

where Dmin(x, y) is the minimum distance point on the curve C(t) from the center P(x, y) of a circle with radius *R*.

3 ALGORITHM DETAILS

3.1 MST of a set of curves

As the set of curves are disjoint, MST is used to make a connectivity between them. Moreover, the edges in the MST act as a part of a rubber-band that enclose the curves. MST also guarantees that no other edge obstruct the edge between two curves. Establishing connectivity through MST also determines the starting set of triplets of curves. The minimal spanning tree (MST) of a set of curves is defined as follows: each curve corresponds to a node and the minimum distance line joining two curves corresponds to an edge. To determine the minimum distance line between two curves, it can be noted that they have to be antipodal to both curves [16]. For two planar closed C^1-continuous curves $C_1(t)$ and $C_2(r)$, the antipodal constraints are:

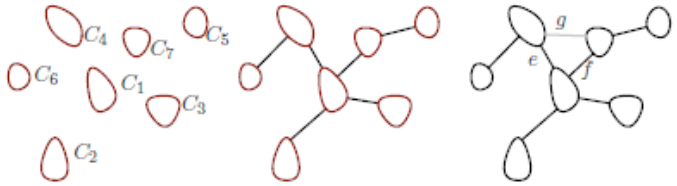
$$\begin{aligned} \left\langle C_1'(t), C_1(t) - \frac{C_1(t) + C_2(r)}{2} \right\rangle &= 0, \\ \left\langle C_2'(r), C_2(r) - \frac{C_1(t) + C_2(r)}{2} \right\rangle &= 0. \end{aligned}$$

(3)

where $C'_1(t)$ and $C'_2(r)$ denote the tangent of the curves at the respective parameters t and r. Equation (3) will generate all the sets of antipodal points, the least of which forms an edge of MST. Prim's algorithm is used to generate the MST for a set of curves (Algorithm 1). Figure 1(b) shows the MST for the set of curves given in Figure 1(a).

Algorithm 1 $MST(S = C_1, C_2, ...C_n)$

```
Add curve C_1 to a set C.
while C ≠ S do
  for Each curve C_i belonging to C do
    Find antipodal line to every curve C_j in S not belonging to C
  end for
  Find min line M_L and add S_j and M_L to C.
end while
return(C)
```

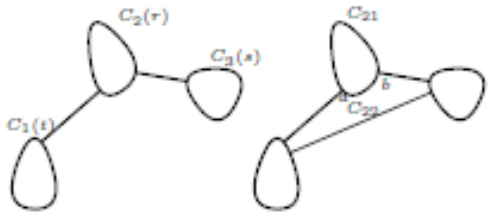


(a) (b) (c)

Fig. 1: MST for a set of curves and dilation of edges: (a) Initial set of curves, (b) MST of the set, (c) Edges **e** and **f** replace to form **g**

3.2 Forming triplets using MST

For the MST of a set of curves shown in Figure 1(b), the adjacent edges (e and f) are then dilated to form a new edge (g) as shown in Figure 1(c). However each time the dilation is performed, a portion of the curves is left from further computation. In the case of Figure 2, curve C2 has edges to curves C1 and C3 in the MST which are replaced by a new edge, a portion of curve C2 is omitted from further computation whereas the other portion is still considered. Thus the curves have to be divided into segments to denote the portions of the curve. In the current case, the curve C2 is called the 'central' curve. The central curve is split into two segments using the antipodal points. Once the curve is split into two segments, it is denoted by the triplet which has the included angle (with respect to the central curve) in the counter clockwise sense. For example (Figure 2(b)), the curve C2 is split into C21 (portion of C2 from the point a to b) and C22 (portion of C2 from b to a). The corresponding triplets are represented as [C1 C2 C3] and [C3 C2 C1] respectively. In the case of curve C3 in Figure 2(a), the entire curve is stored as a single segment and denoted as [C2 C3 C2].



(a) (b)

Fig. 2: Segmenting Curves: (a) Minimum area enclosure, (b) Newly created edge

Algorithm 2 *TripletMatrix(MST)*

```
for Each curve C_i ∈ MST(S) do
  if Valency(C_i) ≥ 2 then
    for Each pair of adjacent MST edges L_m and L_n connecting C_i to C_j and C_k respectively do
      The segment of the curve between L_m and L_n is denoted as [C_j C_i C_k]
      Add [C_j C_i C_k] to triplet matrix TM corresponding to C_i.
    end for
  else
    Add [C_j C_i C_j] to triplet matrix TM corresponding to C_i.
  end if
end for
return TM
```

3.3 Triplet matrix

For a set of curves as shown in Figure 1(a), the triplets of each of the curves form a triplet matrix (TM). The valency of a curve is determined by the number of MST edges attached to it. A curve in a MST is split into segments equal to the valency of the curve. For each such segment obtained, triplets are generated as described in section 3.2. In a similar way, all the triplets identified are stored in TM. Table 1 shows the triplet matrix of each central curve at the start using the MST (Figure 1(b)). Algorithm 2 explains the formation of TM from MST.

C_1	C_2	C_3	C_4	C_5	C_6	C_7
$[C_3\ C_1\ C_7]$	$[C_1\ C_2\ C_1]$	$[C_1\ C_3\ C_1]$	$[C_6C_4C_1]$	$[C_7\ C_5\ C_7]$	$[C_4\ C_6\ C_4]$	$[C_5\ C_7\ C_1]$
$[C_7\ C_1\ C_4]$			$[C_1C_4C_6]$			$[C_1C_7C_5]$
$[C_4\ C_1\ C_2]$						
$[C_2\ C_1\ C_3]$						

Tab 1: Starting Triplet Matrix for convex hull

C_1	C_2	C_3	C_4	C_5	C_6	C_7
$[C_3C_1C_7]$	$[C_1C_2C_1]$	$/C_1C_3C_1/$	$[C_6C_4C_1]$	$[C_7C_5C_7]$	$[C_4C_6C_4]$	$[C_5C_7C_4]$
$[C_4C_1C_2]$			$[C_7C_4C_6]$			$/C_1C_7C_5/$
$[C_2C_1C_3]$						

Tab 2: Updated Triplet Matrix cvx-hull

3.4 Updating triplet matrix

Suppose a triplet be picked from the triplet matrix (say [C7 C1 C4]). Though the triplet denotes the segment of the curve C1, it should be noted that from the triplet [C7 C1 C4], pairs of curves [C7 C1] and [C1 C4] may be part of another triplet in TM (see Table 1). Thus, when concavities are removed by deleting triplets, other dependent triplets have to be updated. The triplet matrix is updated in the following way: we check to see if the pair [C7 C1] exists as a part of any other triplet under C7 and replace C1 by C4. Similarly we check for [C1 C4] under C4 and replace C1 by C7 as shown in Table 2. At each iteration one triplet gets eliminated from the matrix and two triplets are updated.

Effect of the changes in the Table 2, edges in the initial MST now dilates as shown in Figure 1(c) (newly formed edge is shown in grey). The dilation of the initial MST is a result of deletion of the triplet or in other words the segment of the curve from the triplet matrix. This is akin to pushing out the rubber-band formed by MST towards the exterior (this can also be thought of eliminating right oriented triplets until only left oriented triplets remain, which will then correspond to the convex

chain). Also, the segment of the curve will play no role in the further computations. Algorithm 3 describes the procedure for updating the triplet matrix.

Algorithm 3 *UpdateTM*$(TM, [C_iC_jC_k])$

```
if The pair [C_i C_j] is present in the triplets under [C_i] then
  Replace the triplet [C_n C_i C_j] by [C_n C_i C_k]
end if
if The pair [C_j C_k] is present in the triplets under [C_k] then
  Replace the triplet [C_j C_k C_m] by C_i C_k C_m]
end if
Delete [C_i C_j C_k] from TM.
```

3.5 Starting Triplet

Section 3.4 described how to update the triplet. However, algorithm for updating a triplet matrix typically needs a starting triplet. At the start of the updating step of the algorithm in each stage till the triplet matrix becomes empty and pairs are formed, the maximum inscribed circle is found for all triplets in the current triplet matrix. The minimum radius out of the maximum inscribed circle and its corresponding triplet is chosen as the starting triplet. Also, if no circle is available from the triplets in the triplet matrix, then it means that all the curves in the triplets are a part of the convex hull. Union of all the pairs of curves will form the convex hull. The reason for picking the triplet with smallest radius of MIC is very similar to the empty circumcircle of a Delaunay triangulation. Since our aim is to push the rubber band out, the smallest radius MIC is picked.

Algorithm 4 *CurvesInConvexHull*(S)

```
TM=nil
PM=nil.
MST= MST(S).
TM=TripletMatrix(MST).
while TM ≠ NULL do
  for Each element in TM do
    Find MIC (Maximum inscribed circle) using Equations (1) and (2)
  end for
  if Number of MIC ≠ 0 then
    Circ_min = Minimum (MIC).
    Let [C_iC_jC_k] be the triplet corresponding to Circ_Min
    Call UpdateTM(TM,[C_iC_jC_k])
  else
    for Each Element T_x in TM do
      Split T_x= [C_i C_j C_k] into [C_i C_j ] and [C_j C_k] and add to PM
      delete T_x
    end for
  end if
end while
for Every pair [C_i C_j] in PM do
  Merge the pairs to get curves in the convex hull
end for
```

4 ILLUSTRATION OF THE ALGORITHM

The pseudo-code to compute the curves in the convex hull is presented in Algorithm 4. The algorithm is illustrated for the set of curves shown in Figure 1. Table 3 to Table 8 explain the various deletions and updating that happen in the triplet matrix at every step. In each table, the triplet to be deleted is shown in larger font and the triplets to be updated are shown in italics. At each stage, the edges connecting the curves (the rubber-bands) gets pushed out and at the last stage, the edge connectivity will give the curves that will be on the convex hull. Figures 3(a) to 3(f) indicate the process of formation of the curves in the convex hull. Since no more MIC is possible from the triplets in Table 8, the triplets are split pair-wise and are added to the pair matrix (PM) which gives the curves in the convex hull.

C_1	C_2	C_3	C_4	C_5	C_6	C_7
[C_4 C_1 C_2]	[C_1 C_2 C_1]	[C_1 C_3 C_7]	**[C_6 C_4 C_1]**	[C_7 C_5 C_7]	*[C_4 C_6 C_4]*	[C_5 C_7 C_4]
[C_2 C_1 C_3]			[$C_7C_4C_6$]			[$C_3C_7C_5$]

Tab 3: Updated Triplet Matrix cvx-hull-2

C_1	C_2	C_3	C_4	C_5	C_6	C_7
[C_6 C_1 C_2]	[C_1 C_2 C_1]	*[C_1 C_3 C_7]*	[$C_7C_4C_6$]	*[C_7 C_5 C_7]*	[C_4 C_6 C_1]	[C_5 C_7 C_4]
[C_2 C_1 C_3]						**[C_3 C_7 C_5]**

Tab 4: Updated Triplet Matrix cvx-hull-3

C_1	C_2	C_3	C_4	C_5	C_6	C_7
[C_6 C_1 C_2]	*[C_1 C_2 C_1]*	[C_1 C_3 C_5]	[$C_7C_4C_6$]	[C_3 C_5 C_7]	*[C_4 C_6 C_1]*	[C_5 C_7 C_4]
[C_2 C_1 C_3]						

Tab 5: Updated Triplet Matrix cvx-hull-4

C_1	C_2	C_3	C_4	C_5	C_6	C_7
[C_2 C_1 C_3]	*[C_6 C_2 C_1]*	*[C_1 C_3 C_5]*	[$C_7C_4C_6$]	[C_3 C_5 C_7]	[C_4 C_6 C_2]	[C_5 C_7 C_4]

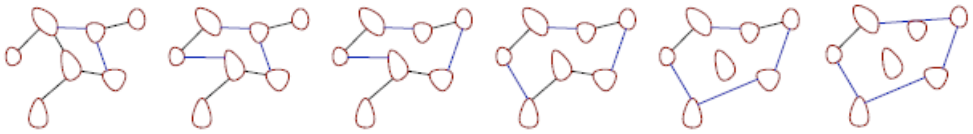
Tab 6: Updated Triplet Matrix cvx-hull-5

C_2	C_3	C_4	C_5	C_6	C_7
[C_6 C_2 C_3]	[C_2 C_3 C_5]	*[$C_7C_4C_6$]*	*[C_3 C_5 C_7]*	[C_4 C_6 C_2]	[C_5 C_7 C_4]

Tab 7: Updated Triplet Matrix cvx-hull-6

C_2	C_3	C_4	C_5	C_6
[C_6 C_2 C_3]	[C_2 C_3 C_5]	[$C_5C_4C_6$]	[C_3 C_5 C_4]	[C_4 C_6 C_2]

Tab 8: Updated Triplet Matrix cvx-hull-7



(a) (b) (c) (d) (e) (f)

Fig 3: Dilation of the MST edges through the various stages of the algorithm

5 RESULTS AND DISCUSSIONS

Implementation of the algorithm (Algorithm 4) has been carried out using IRIT [6], a solid modeling kernel. The constraint equations were solved using the geometric constraint solver [7] in IRIT. Figure 3(f) shows the result for the curves in Figure 1(b), which shows the MST of the set. Figure 4 shows the set of curves with MST in the top row with corresponding output (curves in the convex hull in the bottom row).

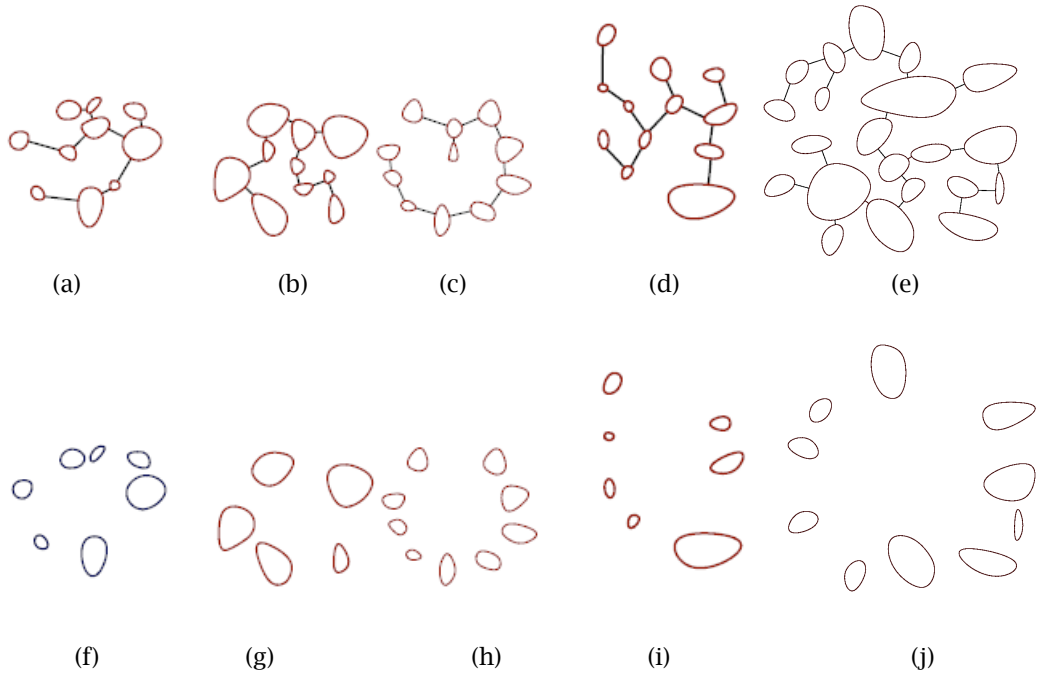


Fig 4: Test results showing MST(top) and curves in convex hull(bottom)

5.1 Complexity if the algorithm

Using Algorithm 4, the complexity can be derived as follows: MST computation starts from the antipodal ones, which is a complete graph and takes $O(n^2)$. Hence, the MST computation of the prim's algorithm will be of $O(n^2 \log n)$ [1]. Assume constraint equations are performed in constant time O(1). If there are e edges in the MST, in the worst case, the number of triplets is 2e. At the most, the updating of the triplet matrix has to be carried out e times. This implies that that Algorithm 4 runs in $O(n^2 \log n)$ in the worst case.

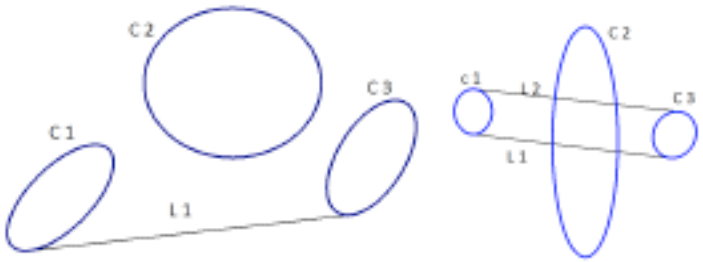
5.2 Correctness proof of the algorithm

A convexity exists between three curves C1, C2 and C3 only if a straight line can be drawn between curves C1 and C3 which does not intersect curve C2 (Figure 5(a)).

Lemma 1 Maximum inscribed circle exists between three curves C1, C2 and C3 exist if and only if the convexity exists.

Proof Let us assume that there exists a MIC between curves C1, C2 and C3 which does not have convexity (Figure 5(b)). Maximum inscribed circle between three curves is always tangential to the curves. A general property of circles is that any point in the circumference of a circle should be visible from any other point. In other words there must be some point in each of the three curves from which the other two curves are visible. In Figure 5(b) there exist no straight line connecting curves C1 and C3

that does not intersect C2 . This means that no MIC is possible between curves where there is no convexity. If there exists two points on C1 and C3 (say a and c), consider the intersection point of the normals from the two points (let the point be I). From I, circles can be drawn using I as center and Ia and Ib as respective radii. If at least one of the circles cut the curve C2, then it can be shrunk to find mic, since the curves are c1-continuous and hence the normal field is also continuous (if both circles do not cut c2, then one can find a pair of different set of points and then use a similar argument). In our algorithm, each time the triplet matrix is updated, the rubber band enclosure surrounding the set of curves reduces in length. In actual terms, two edges of a triangle are replaced by a single edge as shown in figure 1(c). Since the algorithm is terminated at a stage when there are no more triplets with concavity, it is concluded that the remaining set of curves will be in the convex hull of the set.



(a) (b)

Fig 5: Concavity and convexity between three curves: (a) Bi-tangent line L1 does not intersect curve C2 (b) No Line exists between c1 and c3 that does not intersect C2

5.3 Comparison

Though there are numerous algorithms to compute the points on a convex hull from a set of points, to the best of the knowledge of the author, only [8] handle curves without approximating them (it should be noted our algorithm does not actually compute the hull). Algorithm in [8] uses point-curve tangents and bi-tangents, formulation constraint equations to obtain them. They are then processed for transversal intersection with a curve, either to be part of the convex hull or reject them. Complexity of the algorithm appears to be $O(d^2)$, where d is the number of bitangents. The number of bitangents used in [8] to compute convex hull seems to be quite high as it involves all curves in the set. Using the algorithm in this paper, the number can be substantially reduced as we know which curves contribute to the convex hull and also the connectivity, essentially requiring 4n bitangents, where n is the number of curves in the convex hull.

It is also to be noted the algorithms such as gift-wrapping can be extended to the curve domain as well. In any case, it appears that the computation of bitangent is not avoidable along with the predicate computation of whether the curve is right or left of a bitangent. Curves in the convex hull presented here will reduce the number of curves that have to be used in the computation of convex hull thereby reducing the number of bitangents and a possible elimination of the predicate tests. The algorithm in [14] has been shown to improve the efficiency of the computation of the convex hull. However, the algorithm requires a pre-processing step that approximates the curves into a set of bi-arcs with an error bound. Moreover, this kind of approximation does not seem to generalize for higher dimensions such as in three dimensions, where as the constraints proposed in this paper are extendable to higher dimensions.

5.4 Limitation

Though it has been assumed that the disc touches only one point on a curve, computing multi-touching point within the same curve is still possible. For example, in the case of two points lying on the same curve, the constraint equations can be solved considering C1 = C2. However, as the same

curve can play a role in multiple touchpoints, maintaining and updating triplet matrix has to be appropriately done.

6 CONCLUSION

In this paper, an algorithm for determining curves that belong to the convex hull from a set of planar closed disjoint curves based on MST computation. It has been shown that the curves in convex hull can be obtained without geometrical structures such as Voronoi diagram or Delaunay triangulation, that are traditionally proven to be difficult to compute for curves. The curves are assumed to be simply-connected convex curves, having no straight line portions and no discontinuities. Results indicate that the algorithm is very amenable for implementation. Possible future work can include GPU-based computation and also handling of intersecting curves.

REFERENCES

[1] Aho A. V.; Hopcroft J. E.; Ullman J. D.: *Data Structures and Algorithms*. Addison-Wesley, 1983

[2] Albocher D.; Elber G.: On the computation of the minimal ellipse enclosing a set of planar curves. In *Shape Modeling International*, pages 185–192, 2009

[3] Chou J. J.: Voronoi diagrams for planar shapes. *IEEE Comput. Graph. Appl.*, 15(2):52–59, 1995.

[4] Corney J.; Rea H.; Clark D.; Pritchard J.; Breaks M.; MacLeod R.: Coarse filters for shape matching. *IEEE Computer Graphics and Applications*, 22:65–74, 2002.

[5] De Berg M.; Cheong O.; van Kreveld M.; Overmars M.: *Computational geometry: algorithms and applications*. Springer, 2008.

[6] Elber G.: *IRIT 10.0 User's Manual*. The Technion—Israel Institute of Technology, Haifa, Israel, 2009.

[7] Elber G.; Kim M.-S.: Geometric constraint solver using multivariate rational spline functions. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 1–10, New York, NY, USA, 2001. ACM.

[8] Elber G.; Kim M.-S.; Heo H.-S.: The convex hull of rational plane curves. *Graph.Models*, 63(3):151–162, 2001.

[9] Emiris I. Z.; Tsigaridas E. P.; Tzoumas G. M.: Exact delaunay graph of smooth convex pseudo-circles: general predicates, and implementation for ellipses. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, SPM '09, pages 211–222, New York, NY, USA, 2009. ACM.

[10] Emiris I. Z.; Tzoumas G. M.: A real-time and exact implementation of the predicates for the voronoi diagram of parametric ellipses. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, SPM '07, pages 133–142, New York, NY, USA, 2007. ACM.

[11] Hanniel I.; Muthuganapathy R.; Elber G.; Kim M.-S.: Precise Voronoi cell extraction of free-form rational planar closed curves. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 51–59, New York, NY, USA, 2005. ACM.

[12] Fischer K.; Gartner B.: The smallest enclosing ball of balls: combinatorial structure and algorithms. In *Proceedings of the nineteenth annual symposium on Computational geometry*, SCG '03, pages 292–301, New York, NY, USA, 2003. ACM.

[13] Karavelas M. I.: Guarding curvilinear art galleries with edge or mobile guards. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, SPM '08, pages 339–345, New York, NY, USA, 2008. ACM

[14] Kim Y.-J.; Lee J.; Kim M.-S.; and Elber G. : Efficient convex hull computation for planar freeform curves. Computers & Graphics, 35(3):698 – 705, 2011. Shape Modeling International (SMI) Conference 2011.

[15] Lee Y.-S.; Chang T.-C.: 2-phase approach to global tool interference avoidance in 5-axis machining. *Computer-Aided Design*, 27(10):715 – 729, 1995.

[16] Muthuganapathy R.; Elbe G.; Barequet G.; Kim M.-S.: Computing the minimum enclosing sphere of free-form hypersurfaces in arbitrary dimensions. *Comput. Aided Des.*, 43:247–257, March 2011.

[17] O'Rourke J. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 1998.

[18] Pocchiola M.; Vegter G.: Computing the visibility graph via pseudo-triangulations. In *Proceedings of the eleventh annual symposium on Computational geometry*, SCG '95, pages 248–257, New York, NY, USA, 1995. ACM.

[19] Ram S. B.; Ramanathan M.: The shortest path in a simply-connected domain having a curved boundary. *Computer-Aided Design*, 43(8):923–933, 2011.
[20] Ramanathan M.; Gurumoorthy B.: Constructing medial axis transform of planar domains with curved boundaries. *Computer-Aided Design*, 35(7):619–632, June 2003.
[21] Seong J.-K.; Elber G.; Johnstone J. K.; Kim M.-S.: The convex hull of freeform surfaces. *Computing*, 72(1-2):171–183, 2004.