Constructing Medial Axis Transform for Free-form Trimmed Surfaces

D. Naresh Varma[1] and B. Gurumoorthy[2]

[1] Indian Institute of Science, datlanv@yahoo.com
[2] Indian Institute of Science, bgm@mecheng.iisc.ernet.in

ABSTRACT

This paper presents an algorithm for constructing Medial Axis Transform for free-form surfaces. The algorithm uses a combination of tracing and bisector based techniques. This algorithm finds bisector curves by a tracing method and then trims these bisector curves for finding the branch points and constructing the MAT. Results on typical free-form domains are presented.

Keywords: Medial Axis Transform, Free-form surface, geodesic.

1. INTRODUCTION

One of the fundamental issues in CAD has been to represent objects in a way that not only allows the user to create and modify their shape and other attributes but also enables reasoning about the shape for various tasks in the product design cycle. Objects are normally described by their boundaries, but they can also be described by shifting the description to the interior. This is done by representing the skeleton/symmetric axis of the shape. Various skeleton/symmetric axis representation of shapes like Medial Axis Transform (MAT), Voronoi diagrams, box skeletons, mid surfaces and 2.5D skeletons are possible. Of these MAT has shown to be useful in various applications such as finite element mesh generation, pattern analysis and image analysis, path generation for pocket machining, robot motion planning and more

Work on MAT up to now has been restricted to planar 2D and 3D objects. Very little work has been reported on construction of the MAT for 2D free form surface domains. An algorithm to develop MAT on free form surfaces (exact representation) is presented in this paper.

1.1 Medial Axis Transform (MAT)

The medial axis transform was first introduced by Blum [1] to describe biological shapes. It can be viewed as the locus of the center of a maximal disk as it rolls inside an object.

1.2 Definition of 2D MAT for Planar Domains

The medial axis (MA) of the set D, denoted M(D), is defined as the locus of points inside D which lie at the centers of all closed disks which are maximal in D, together with the limit points of this locus. A closed disk is said to be *maximal* in a subset D of the 2D space if it is contained in D but is not a proper subset of any other disk contained in D. The radius function of the MA of D is a continuous, real-valued function defined on M(D) whose value at each point on the MA is equal to the radius of the associated maximal disk. The medial axis transform (MAT) of D is the MA along with its associated radius function [8]. Fig. 1. illustrates the MA segments corresponding to a simple convex domain.

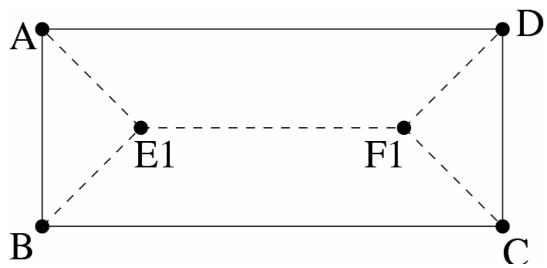

Fig. 1. 2D MAT for a planar rectangular object [7].

1.3 MAT on Free Form Surface

The generalization of MAT to free form surfaces raises special problems that do not occur in the planar case. The first problem arises in computing the distance functions between two points on the surface. In planar case it is simply a straight line connecting the two points where as in the free form case it is the geodesic curve along the surface.

So, to extend the definition of MAT on planar case to MAT on free form surface the disk is replaced by g-disk. A g-disk of radius *r* is defined to be the locus of all points on the surface around the center point whose geodesic minimum distance between center and points is less than or equal to *r*.

The medial axis (MA) of the set D, denoted M(D), is defined as the locus of points inside D which lie at the centers of all closed g-disks which are maximal in D, together with the limit points of this locus. A closed g-disk is said to be *maximal* in a subset D of the 2D space if it is contained in D but is not a proper subset of any other g-disk contained in D. The radius function of the MA of D is a continuous, real-valued function defined on M(D) whose value at each point on the MA is equal to the radius of the associated maximal g-disk. The medial axis transform (MAT) of D is the MA along with its associated radius function.

2. RELATED WORK

Considerable amount of literature is available on the construction of medial axis for planar domains [7]. In the planar domain, the techniques reported thus far can be classified as those based on construction of bisectors, construction of Voronoi regions and tracing along either the domain boundary or the bisectors. The first two techniques need a trimming step to retain only those regions of the bisector or the Voronoi diagram that form the MAT.

Rausch, Wolter and Sniehotta [9] have proposed a method for tracing the medial curve of two border curves lying on a free-form surface. The method is designed to work on arbitrary regular surfaces where distance is understood to be the geodesic distance. The main contribution is the formulation of a differential equation that is useful in tracing the medial curve by following its tangent expressed by the differential equation. The medial curve respective two given curves *A, B* is obtained by intersection points of offset curves with same offset distance from the respective curves *A*, *B*. Medial curves constructed here are only the local equidistant set of certain sub-arcs of two given border curves. Though the authors refer to the computed curve as a medial curve, in reality these are only the bisector curves as these are not guaranteed to satisfy maximal g-disk criterion in the definition of MAT.

Kunze, Wolter and Rausch [4] present a divide and conquer scheme to construct Voronoi diagram on parametric surfaces. The main step here is to compute the bisector of two curves. This is the medial curve as computed in [9]. The Voronoi diagram *V(P)* of the set *P* (the boundary entities on the parametric surface) is constructed by dividing the set *P* in two subsets *L* and *R* of approximately equal size and recursively calculating *V(L)* and *V(R)*. The Voronoi diagram constructed in each recursion is sewed to obtain *V(P)*. The merging step in divide and conquer methods is the difficult step even in planar case. This becomes even more complicated in the free form case.

It is possible to discretize the free-form domain and use the techniques for planar domains to construct the MAT for a free-form domain. However as pointed out in [10] and [8] the MAT for discretized domains do not retain the topology of the original object. The output consists of artificial MAT segments that are artifacts of the representation used and not due to the underlying part geometry. Even in 2D planar case if the boundary curves are discretized the the output MAT do not conform to the definition of MAT for exact domain. An additional effort is required in trimming and post-processing the generated MAT to be in conformity with the topology of the original free form object.

Tracing methods do the check for branch points while the MAT points are generated. The check for branch point involves finding the minimum distance boundary curve from the MAT point calculated, which is straightforward in the case of planar domains. However for free-form domains this check requires the computation of minimum distance geodesic between a point and a curve, which is computationally costly. Therefore tracing methods from the literature cannot be extended to handle free-form domains in a straight forward manner.

In this paper a bisector method is used to construct MAT for free form surface domains that are represented exactly. A combination of bisector based and tracing based techniques is used for constructing the MAT. First the bisector for a pair of boundary entities is constructed using a tracing scheme. This bisector is then used to trim the extraneous portions of bisectors constructed prior to this step using Chou's [2] scheme.

3. DEFINITIONS AND TERMINOLOGY

3.1 Classification of Vertices

A vertex on a surface is said to be *concave (reflex) vertex* if the interior angle between the tangent vectors to the incident edges at the vertex is greater than 180 ^{0}. If the angle is equal to 180 ^{0}, then it is termed a *smooth vertex*. Otherwise, it is termed a *convex vertex*.

3.2 Classification of points on MAT

Points on the MAT can be classified based on the properties of the maximal g-disk. For convenience the description of points have been illustrated for a planar domain (Fig. 2.). A point whose maximal g-disk touches exactly two boundary segments is called *normal point*, point N.

A point whose maximal g-disk touches the domain boundary in three or more segments is called a *branch point*. Points E1 and F1 in Fig. 2(c) are branch points.

A point whose maximal g-disk touches the boundary in exactly one contiguous set is called an *end point*. Fig. 2(a), shows the end points A, B, C and D. These points touch the boundary at a point and the corresponding maximal g-disk is of radius zero.

A point of contact with the domain boundary, of the underlying g-disk of a point on MAT is called the *foot point* of the point on the MAT. Points Fp1 and Fp2 in Fig. 2(b).

From the definition of point types on MAT, a normal point will have two foot points, a branch point will have three or more foot points and an end point will have one or more foot points.

The boundary entities (boundary segments) corresponding to the MAT point are referred to as *progenitor curves*. For example, a normal point has two progenitor curves and a branch point will have at least three progenitor curves.

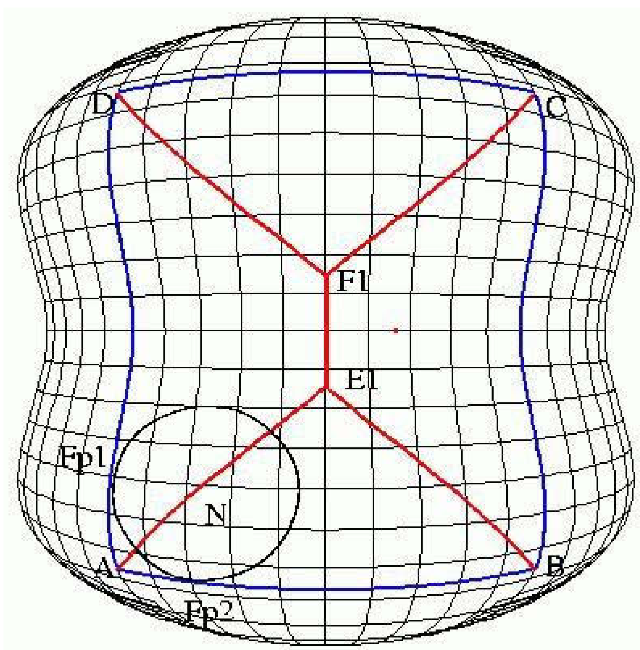

Fig. 2. Classification of points on MAT(surface represented by iso-parametric curves, trimmed object boundary by blue curves and Medial Axis by red curves).

4. OVERVIEW OF THE CONSTRUCTION ALGORITHM

The algorithm works by generating bisectors for pairs of boundary entities of the domain by a tracing method. As each bisector is being traced, it is also used to trim the existing set of bisectors to form the MAT. Chou's algorithm [2] for the construction of Voronoi diagrams for planar domains is used for the trimming step. The bisector is traced using a modified version of the tracing technique proposed by Ramanathan [7]. The modifications are made to account for the computation of shortest distance along a surface. Tracing of bisector starts from a convex vertex as it forms the initial guess for the tracing scheme. The input to the algorithm is a boundary representation of the trimmed patch along with the underlying surface. The surface and the boundary curves are assumed to be available in the NURBS representation.

In extending the tracing technique described in [7] the critical difference is in finding points on the medial curves/bisectors on the free form surface. This involves computing shortest distances along the surface, namely the shortest distance geodesic.

The tracing procedure works by finding the foot point of a MAT point on one edge given the foot point on another edge. In the planar domain this is accomplished by solving the intersection of the normals to the 2 edges and imposing the distance criterion. In the free-form domain this would involve computing geodesic curve of a specified length given a point on the surface and a direction. The length of the geodesic path is taken to be slightly larger than the radius function at the previous foot point on the same edge. The direction is the normal to the edge at the foot point.

5. COMPUTAIOTN OF POINTS ON THE BISECTOR/MEDIAL CURVE OF TWO BOUNDARY CURVES

As mentioned above, the computation of points on the bisector curve requires the determination of geodesic on the surface.

A geodesic curve on a surface is governed by the following set of differential equations [3]. *u,v* are the surface parameters.

$$\frac{du}{ds} = u_1 \tag{1}$$

$$\frac{dv}{ds} = v_1 \tag{2}$$

$$\frac{du_1}{ds} = -d_{11}u_1^2 - 2d_{12}u_1v_1 - d_{22}v_1^2 \tag{3}$$

$$\frac{dv_1}{ds} = -e_{11}u_1^2 - 2e_{12}u_1v_1 - e_{22}v_1^2 \tag{4}$$

Given a starting point on the surface and an initial direction a geodesic trajectory can be traced for a specified length [5] by integrating the above system of differential equations with the initial values using a 4^{th} order Range-Kutta method [6]. Computation of the geodesic is terminated when the geodesic curve reaches the specified length.

Two algorithms have been developed to find the medial curves given two progenitor curves (boundary curves).

5.1 Geodesic-Geodesic Intersection Algorithm

Given two boundary curves $C_1(t)$ and $C_2(s)$(Fig. 3.) their medial curve is traced in this step. Given a MAT point P and the corresponding foot points $C_1(t_0)$ and $C_2(s_0)$ the next MAT point P^* and the corresponding foot points $C_1(t^*)$ and $C_2(s^*)$ are calculated by this procedure.

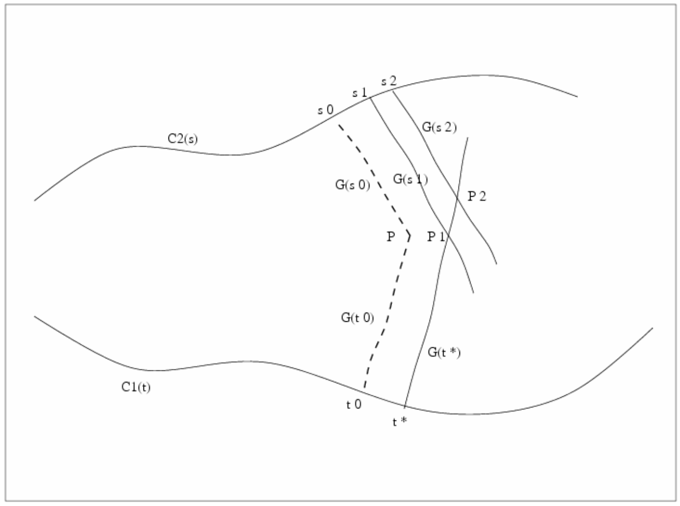

Fig. 3. Illustration of the Geodesic-Geodesic Intersection algorithm in parametric space.

- March along the curve $C_1(t)$ from the current foot point $C_1(t_0)$ to $C_1(t^*)$ where $t^*=t_0+\delta t$.
- Shoot a geodesic curve $G(t^*)$, orthogonal to the curve $C_1(t)$ from $C_1(t^*)$.
- Shoot two geodesics curves $G(s_1)$ and $G(s_2)$ orthogonal to the curve $C_2(s)$ from $C_2(s_1)$ and $C_2(s_2)$ respectively. Where $s_1=s_0-\delta s$ and $s_2=s_0-2\delta s$.
- Intersect the curves $G(t^*)$ and $G(s_1)$ and find the intersection point P_1. Denote d_t^1 as the distance between $C_1(t^*)$ and the intersection point P_1 along the curve $G(t^*)$. Like wise d_s^1 is the distance between $C_2(s_1)$ and the intersection point P_1 along the curve $G(s_1)$.
- Intersect the curves $G(t^*)$ and $G(s_2)$ and find the intersection point P_2. Denote d_t^2 as the distance between $C_1(t^*)$ and the intersection point P_2 along the curve $G(t^*)$. Like wise d_s^2 as the distance between $C_2(s_2)$ and the intersection point P_2 along the curve $G(s_2)$.
- We define the length differences between the geodesics at the two intersection points as $ds_1=d_t^1-d_s^1$ and $ds_2=d_t^2-d_s^2$.
- We know that the two normal geodesics from progenitor curves intersecting at the MAT point are equal length. We have to find particular s^* such that the normal geodesic $G(s^*)$ intersects the other normal geodesic $G(t^*)$ in equal length, i.e. $ds^*=0$. We do this by linear interpolation, which gives

$$\frac{ds^* - ds_1}{s^* - s_1} = \frac{ds_2 - ds_1}{s_2 - s_1} \tag{5}$$

as $ds^*=0$ we find s^* as

$$s^* = s_1 - ds_1\left(\frac{s_2 - s_1}{ds_2 - ds_1}\right) \tag{6}$$

- Knowing s^* we verify whether $ds^*=0$, if not iterate until $ds^*=0$.

This algorithm is simple, easy to implement and computationally effective. The intersection of the normal to edge geodesics may fail when both the normal geodesics are close to being parallel. This case can be trapped by tracking the angle between the normal geodesics at the previous MAT point. If the angle between them is close to 180^0 then this condition is likely to occur. This is a case that can be predicted but cannot be avoided. We consider this as a special case and use the following algorithm to overcome this situation.

5.2 MAT Tangent-Geodesic Intersection Algorithm

The two boundary curves are $C_1(t)$ and $C_2(s)$ (Fig. 4.) their medial curve is traced. A MAT point P, MA tangent vector T at P and the corresponding foot points as $C_1(t_0)$ and $C_2(s_0)$ are known and the next MAT point P^* and the corresponding foot points $C_1(t^*)$ and $C_2(s^*)$ are calculated as follows.

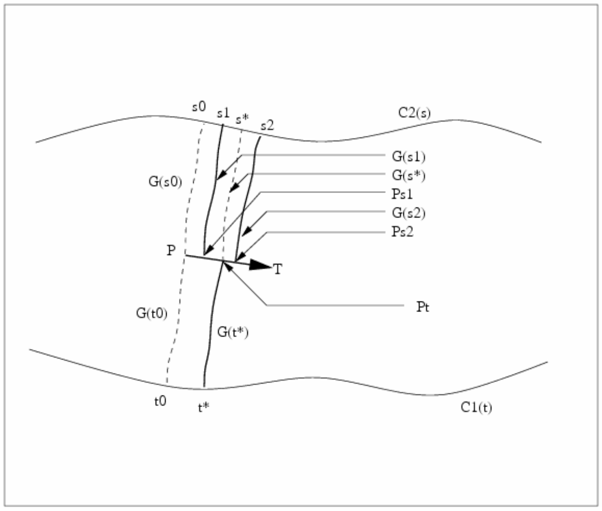

Fig. 4. Illustration of the MA Tangent-Geodesic intersection algorithm in parametric space

- March along the curve $C_1(t)$ from the current foot point $C_1(t_0)$ to $C_1(t^*)$ where $t^*=t_0+\delta t$.
- Shoot a geodesic curve $G(t^*)$, orthogonal to the curve $C_1(t)$ from $C_1(t^*)$.
- Intersect this geodesic curve $G(t^*)$ with the MA tangent T at previous MAT point, denote this intersection point as P_t.
- Let l_t denote the length between the previous MAT point and the intersection point P_t.
- Shoot two geodesics curves $G(s_1)$ and $G(s_2)$ orthogonal to the curve $C_2(s)$ from $C_2(s_1)$ and $C_2(s_2)$ respectively. Where $s_1=s_0-\delta s$ and $s_2=s_0-2\delta s$.
- Intersect the curve $G(s_1)$ with the tangent vector T and find the intersection point P_s^1. Note l_s^1 as the length between the previous MAT point P and the intersection point P_s^1.
- Intersect the curve $G(s_2)$ with the tangent vector T and find the intersection point P_s^2. Note l_s^2 as the length between the previous MAT point P and the intersection point P_s^2.
- Now our aim is to find the point $C_2(s^*)$ on the curve $C_2(s)$, such that the orthogonal geodesic from that point will intersect the tangent T at the same point as the previous intersection point P_t or in other word $l_t=l_s^*$.
- $C_2(s^*)$ is determined by linear interpolation

$$\frac{s^* - s_1}{l_s^* - l_s^1} = \frac{s_2 - s_1}{l_s^2 - l_s^1} \tag{7}$$

as $l_t=l_s^*$ we find s^* as

$$s^* = s_1 + \left(l_t - l_s^1\right)\left(\frac{s_2 - s_1}{l_s^2 - l_s^1}\right) \tag{8}$$

- Let d_t^* be the geodesic distance of the curve $G(t^*)$ from $C_1(t^*)$ to P_t. Like wise, let d_s^* be the geodesic distance of the curve $G(s^*)$ from $C_2(s^*)$ to P_t.
- Now we define $ds=d_t^*-d_s^*$, from the definition of MAT we know that $ds=0$.
- So, we iterate the above steps by modifying the tangent vector until we get $ds=0$. The tangent is modified as follows

- The above steps are repeated with a new tangent vector T^1 (taken close to the previous tangent vector T) and the corresponding ds^1 is determined.
- Now using both tangent vectors T and T^1 and the corresponding length differences ds and ds^1, interpolate the correct tangent T^* vector which gives $ds^*=0$.

5.3 Bisector curve between a point and curve

The bisector curve between a point and a curve is required in the MAT algorithm when a concave vertex is encountered while moving along the boundary. This algorithm is similar to the bisector algorithms discussed above with slight modifications. In this case instead of incrementing parameter of the curve the direction along which the geodesic has to be computed is incremented at this concave point. Rest of the algorithm remains the same. The angle is incremented from the normal direction of left curve incident at the concave vertex to the normal direction of the right curve incident at the concave vertex.

6. TRIMMING THE BISECTOR/MEDIAL CURVES

This section presents the algorithm for trimming the medial curves obtained to construct the MAT on free form surface.

The basic tracing procedure as described above, computes a bisector of two boundary segments $C_i(s)$. A point b on the bisector corresponds to two foot points $Fp(b)$ on the boundary segment. Point b satisfies the following conditions:

- b is equidistant to the two points $Fp(b)$.
- b lies on the normal lines of $C_i(s)$ at $Fp(b)$.
- b is on the left of both points.

The three conditions dictate that b is on the center of the g-disk touching $C_i(s)$ at $Fp(b)$. We denote such a g-disk as $\Gamma(b)$. However, $\Gamma(b)$ may enclose points of $C_i(s)$. When $\Gamma(b)$ does not enclose points of $C_i(s)$, b is on the MA, by definition. The bisector is the super set of MA.

The start point of a bisector is either given (convex corner) or taken from the terminating point of previous bisector. The termination of the bisector being traced is determined as follows:
From the start point and its associated foot points, trace out a bisector by stepping along the bisector point by point and finding each point that satisfies the conditions listed above. When such a point cannot be found, the process stops. It also stops when the two foot points collapse into one (*convex vertex*).

From each tracing we obtain a 1D bisector curve. If we start at one endpoint of a MA segment and trace forward, the traced bisector curve includes the MA segment and may continue beyond. For closed domains, the tracing always terminates, since either a valid bisector point cannot be found or Fp^l (foot point of left curve) and Fp^r (foot point of right curve) meet.

6.1 Algorithm

The algorithm used here is similar to the method used by Chou[2]. To compute the MAT, we first construct a list M of convex corners on boundary. The points in the list are ordered along the boundary. We generate the MAT by tracing the bisectors starting from each point in the list and proceeding one after another according to the list. For the bisector starting at b, the initial inputs to the procedure sets are b and $Fp(b)$,
with $Fp^l=Fp^r$. In addition to the stepping we have to perform the following operations during tracing. If any of the foot points Fp^l or Fp^r belong to the foot points of an existing bisector, we need to merge that bisector and the the bisector under tracing. Suppose Fp^l is such a point and belong to the Foot point of B^1, the existing bisector. (The case for Fp^r is handled similarly.) Let B^r be the bisector under tracing. The bisector points corresponding to Fp^l, are $B^1(Fp^l)$ and $B^r(Fp^l)$. There are three possible relationships between the radii of $\Gamma(B^l(Fp^l))$ and $\Gamma(B^r(Fp^l))$:

- Case 1. If the radius of $\Gamma(B^r(Fp^l))$ is smaller, we remove $\Gamma(B^l(Fp^l))$ from our diagram and keep $\Gamma(B^r(Fp^l))$. The tracing of B^r continues.
- Case 2. If the radius of $\Gamma(B^r(Fp^l))$ is greater, the tracing stops.
- Case 3. If the two radii are equal, B^r and B^l intersect.

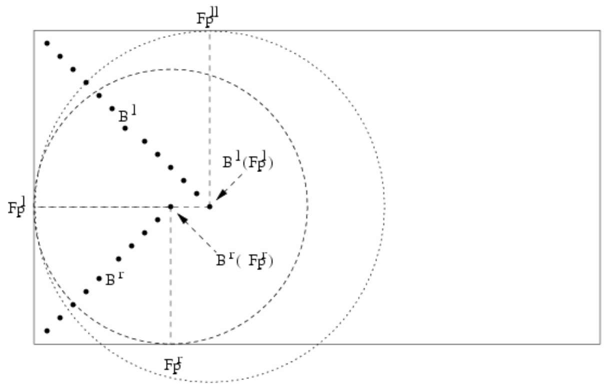

Fig. 5. Illustration for case 1 of the algorithm

When B^l and B^r intersect, we encounter an *intersection point* (at $\Gamma(B^l(Fp^l))=\Gamma(B^r(Fp^l))$). After that, the tracing continues from the intersection point and starts a new bisector. Let $Fp(B^l\ (Fp^l))=Fp^{ll}$, Fp^l and $T(B^r\ (Fp^l))= Fp^l,Fp^r$. The new bisector starts at the intersection point, with Fp^{ll} and Fp^r as the foot points on the left and right boundary segments, respectively. The tracing proceeds in the direction of increasing s^r and decreasing Fp^{ll}. An intersection point is a branch point if not trimmed by subsequent tracing.

7. RESULTS AND DISCUSSION

The algorithm discussed above has been implemented and this section presents the results obtained for some typical free form surface patches. The input used is a trimmed surface patch modeled in Rhino software. A sub-routine is written to convert the .sat format output by Rhino to the format required by our program. The base surface is represented in NURBS and the trimmed edges (boundary segments) as B-Spline curves. The output is set of MAT segments, each MAT segment is a set of points. These MAT points can be processed to form a B-Spline curve. MAT obtained for some free form surface objects is shown in Figures 6-9. In each figure, the iso-parametric mesh of the underlying surface, rendered trimmed patch and the medial axis (curves in red) are shown.

Table shows the time taken for generating MAT for some of the figures. The implementation is on a PIV machine with 256MB RAM, CPU 1.7GHz on the RedHat Linux platform. OpenGL (version 1.2) library functions have been used for display. The step size for tracing is $1/50^{th}$ of the curve parameter range.

7.1 Discussion

The step size for tracing is chosen to be smaller than the smallest feature in the domain (tracked through the length of the boundary segments) so that no features are missed. Presently, the implementation uses a step size $1/50^{th}$ of the length of an edge. This choice gives a fine approximation of the MAT with reasonable calculation time. A larger step size will result in a coarser approximation but faster result.

The analysis of the algorithm with respect to the effect of numerical errors due to finite precision indicates that the algorithm is stable with respect to round-off errors. The MAT algorithm was implemented with float, double and long double precision and there was no significant change in the results. Implementation with double and long double gave the same results with a difference of 1E-12, where as float and double gave a difference of 1E-6.

Type of Precision	Time (in seconds)	Radius at branch point	Foot points at branch point
Float	31.30	39.877182006835938	-78.96517944335937 78.95900726318359
Double	33.72	39.877187295998851	-78.96516433880020 78.95900345460933
Long Double	54.47	39.87718729599895	-78.96516433880001 78.95900345460937

Tab. 1. Time taken for generation of MAT and the precision of calculation of branch points with different precisions for Fig. 7.

The algorithm proposed here requires at least one convex vertex to initiate the algorithm. Identifying points with locally maximal curvature would be required to eliminate this restriction while handling smooth domains. Currently the step size used in the tracing procedure is fixed. An adaptive step size (depending on the local geometry) could reduce the computational effort and improve performance. For instance, the step size can be increased in places where the surface is locally planar and decreased where the surface is having high torsion. Construction of MAT for multiply connected free-form domains have not been addressed thus far to the best of our knowledge. Efforts are underway to extend the proposed approach to handle multiply connected free-form domains, analogous to Sreenivasan and Nackman's extension of the algorithm for planar domains.

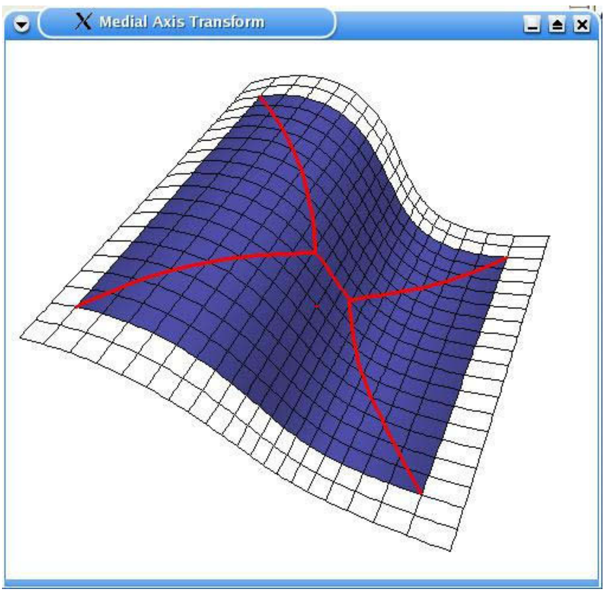

Fig. 6. MAT of a four edged object

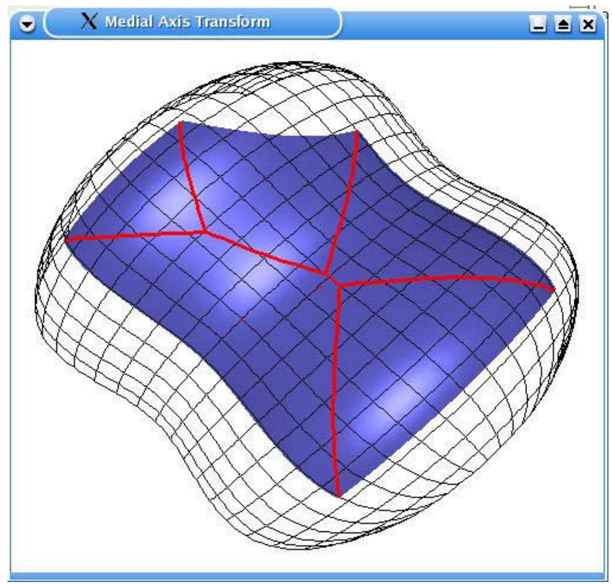

Fig. 7. MAT of an object, used as a case study for stability analysis

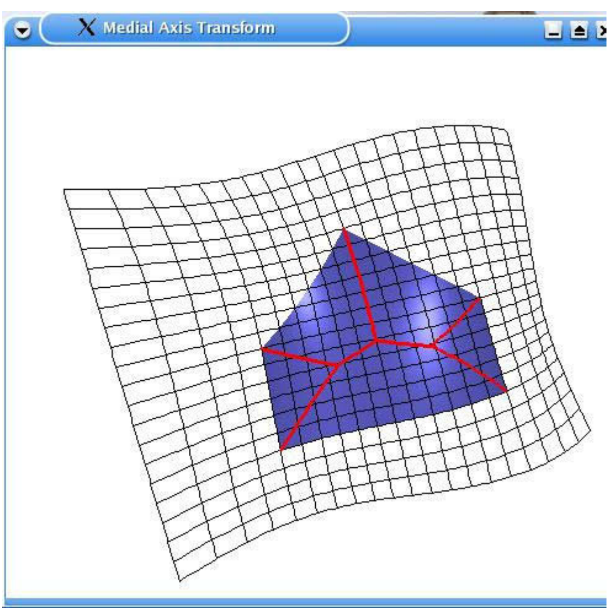

Fig. 8. MAT for a five edged object

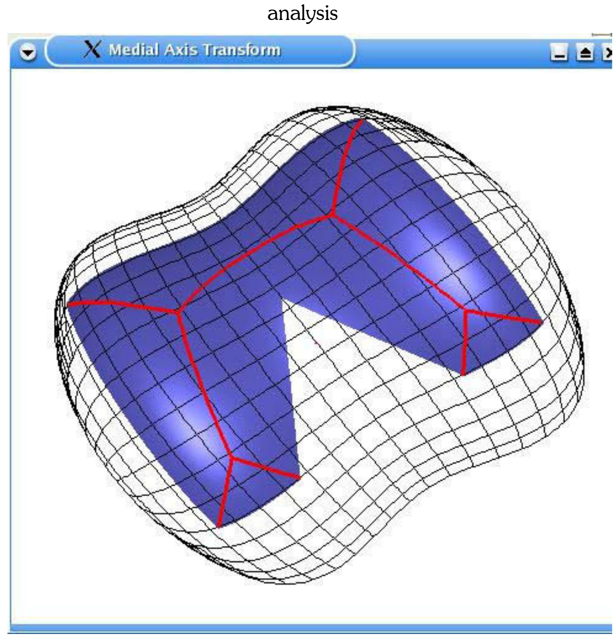

Fig. 9. MAT for an object with a concave vertex

8. CONCLUSIONS

An algorithm has been proposed and implemented generation of MAT for 2D free form surface. The tracing step is an extension of Ramnathan's algorithm [7] given in the planar case. The MAT algorithm is extension of Chou's algorithm [2] given in planar case. It is found that finding branch points by distance check [7] with the boundary segments is computationally inefficient so intersection of bisectors is used to overcome this. The algorithm is found to be stable.

9. REFERENCES

[1] Blum. H., A Transformation for extracting new descriptors of shape, *Models for the Perception of Speech and Visual Form*, ed. Walthen Dunn, MIT Press, 1967, pp 362-380.

[2] Chou, J. J., Voronoi diagrams for planar shapes, *IEEE Computer Graphics and Applications*, Volume 15, Number 2, March 1995, pp 52--59.

[3] do Carmo, M. P., Differential geometry of curves and surfaces, *Prentice-Hall Inc.*, 1976.

[4] Kunze, R., Wolter, F. E. and Rausch, T., Geodesic Voronoi Diagrams on Parametric Surfaces, in *Welfen Laboratory Reports*, Report No. 2, July 1997.

[5] Patrikalakis, N. M. and Badris, L., Offsets of Curves on Rational B-Spline Surfaces, *Engineering with Computers*, Volume 9, 1989, pp 39-46.

[6] Press, W.H., et al., Numerical Recipes in C, *Cambridge University Press*, 1988.

[7] Ramanathan, M. and Gurumoorthy, B., Constructing medial axis transform of planar domains with curved boundaries, *Computer-Aided Design*, Volume 35, 2003, pp 619-632.

[8] Ramanathan, M., *Construction of Medial Axis Transform of Domains Bounded by Free-form Entities*, Ph.D. Thesis, Indian Institute of Science, Department of Mechanical Engineering, December 2003.

[9] Rausch, T., Wolter, F. E. and Sniehotta, O., Computations of Medial Curves on Surfaces, *in Welfen Laboratory Reports*, Report No. 1, August 1996.

[10] Sherbrooke, E. C., Patrikalakis, N. M. and Brisson, E., An algorithm for medial axis transform of 3-D polyhedral solids, *IEEE Transaction on Visualization and Computer Graphics*, Volume 2, Number 1, 1996, pp 44--61.

[11] Srinivasan, V. and Nackman, R. L., Voronoi diagram for multiply-connected polygonal domains I: Algorithm, *IBM Journal of Research and Development*, Volume 31, Number 3, May 1987, pp 361--372.

[12] Wolter, F. E. and Friese, K. I., Local and Global Geometric Methods for Analysis Interrogation, Reconstruction, Modification and Design of Shapes, in *Welfen Laboratory Reports*.

[13] Wolter, F. E., *Cut loci in Bordered and Unbordered Riemannian Manifolds*, Ph.D. Thesis, Technical University of Berlin, Department of Mathematics, December 1985.