An Adaptive Algorithm to Compute the Medial Axis Transform of 2-D Polygonal Domains

R. Juan-Arinyo[*], L. Pérez-Vidal and E. Gargallo-Monllau

August 3, 1999

Abstract

An adaptive algorithm to compute the medial axis transform of 2D polygonal domains with arbitrary genus is presented. The algorithm is based on the refinement of a coarse medial axis transform by and adaptive subdivision of the domain. The algorithm yields the medial axis represented by a set of triangles of a predefined size and the closest boundary element. Examples of results are also presented to illustrate the method.
Keywords: CAD, CAGD, CAM, geometric modelling, solid modelling, medial axis, Voronoi diagram, skeleton, polyhedra.

[*]Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. Barcelona

1 Introduction

The Medial Axis Transform (MAT) was introduced by Blum in [Blu67] as a means to describe shapes in biology and medicine. Currently the MAT is being used in a wide variety of fields like modelling growth, path planning, feature recognition, and finite element mesh generation. Since it provides a complete representation of a solid, attempts of using it as a representation scheme in solid modelling have been reported. For a recent and thorough review of work on MATand related sets see [SPB95].

Lavender *et al.* [LBD$^+$92] report an algorithm that exhibits some conceptual similarities with the method presented here. Their algorithm approximates Voronoi Diagramas of arbitrary set-theoretic solid models based on recursive subdivision of the object space using octrees. Goldak *et al.* [GYKD91], and Sheehy *et al.* [SAR95], have also proposed algorithms to compute the MAT-based on domain Delaunay triangulations. The strategy applied in both papers is based on the classification of the cells resulting from partitioning the domain against the domain boundary.

A recursive algorithm to approximate the medial axis transform of 2D polygonal domains with arbitrary genus is presented. The algorithm has three main steps. First, a domain-compatible Delaunay triangulation of a set of points is computed. This set of points includes the set of vertices and a relatively sparse set of extra points generated on the boundary of the domain. In the second step, the edges of the triangulation that are interior to the polygonal domain are partitioned and labeled according to their closest boundary element. The computed set of labeled segments, which is a coarse MAT of the domain, is then adaptively refined. The algorithm is both conceptually simple and easy to implement.

2 Mathematical Fundamentals

This section gives some definitions and describes the properties of the medial axis that are important to the algorithm presented in the next section.

2.1 Medial Axis and Medial Axis Transform

Let us start by giving three basic definitions that are standard in the field, [Blu67, Wol92].

Definition The *medial axis* of a planar object is the locus of the centers of all maximal discs in the object, together with the limit points of this locus.

Definition The *radius function* of the medial axis of a domain is a continuous, real-valued function defined on the medial axis whose value at each point is equal to the radius of the maximal disc centered in the point.

Definition The Medial Axis Transform (MAT) of a planar object is its medial axis along with its associated radius function.

From now on we shall consider polygonal domains in a 2-D Euclidian space, with arbitrary genus, straight edges with no two adjacent, colineal edges in the boundary. In this context, the maximal discs that define the medial axis can be tangent only to edges and concave vertices in the boundary. Convex vertices represent limit points of the medial axis where the radius function is zero. The edges and concave vertices in the boundary will be called *active boundary elements*.

We classify the points in the medial axis according to the number of active boundary elements to which the maximal disc centered in the point is tangent. The classification is as follows (see Figure 1):

1. *Junction points* : the maximal disc is tangent to three o more active boundary elements. Several medial axis branches meet at a junction point.
2. *End points* : An end type point results when a medial axis branch runs into the domain boundary. These points are actually limit points of the medial axis where the radius function has zero value. In our context, these points are the convex vertices of the polygon.
3. *Normal points* : the maximal disc touches two different active boundary elements. Those points in the medial axis that are neither junction points nor end points are normal points.

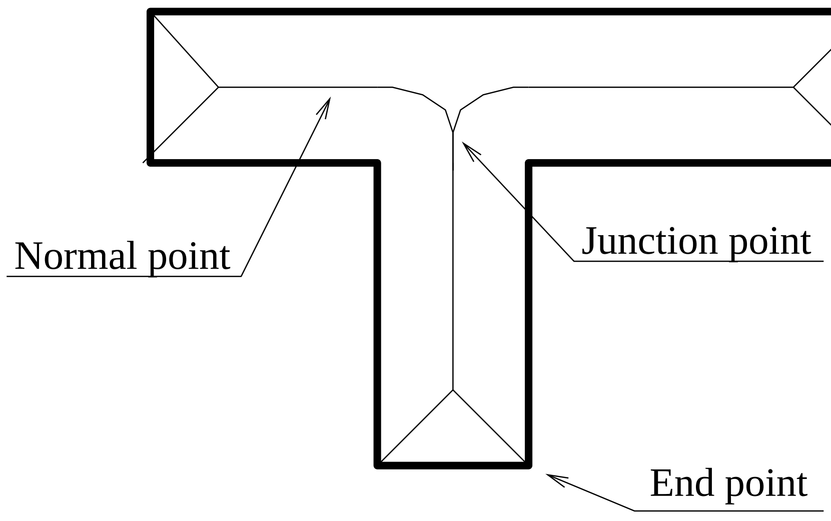

Figure 1: Taxonomy of Medial Axis points.

Every connected subset of points of the medial axis limited by two points each of them being either a joint point or an end point, defines a medial axis *branch.*

2.2 Voronoi Diagrams

Since all maximal discs centered in medial axis points, except for the end points, are tangent to two or more different boundary points, the junction and normal points on the medial axis are equidistant from two or more active elements on the domain boundary. This fact results in the existence of strong relations between the medial axis and point sets with some equidistance function defined on them. Among all these point sets we are interested here in the Voronoi diagram, [Aur91, PS85]

Definition Consider a collection of sets $\{s_i\}$ in the 2-D Euclidean space. The *Voronoi polygon* associated with s_i, denoted by $V(s_i)$, is the locus of points closer to s_i than to any other set. The *Voronoi diagram* of the collection, denoted as $Vor(\{s_i\})$, is the locus of points in the space belonging to two or more Voronoi polygons. The vertices of the Voronoi diagram are called *Voronoi vertices* and its line segments are *Voronoi edges.*

This definition is a generalization of the usual definition where each set s_i is an isolated point, [Kir79] [PS85], . See Aurenhammer, [Aur91], for a survey of

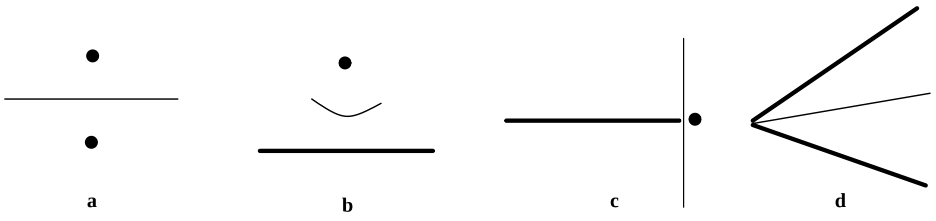

Figure 2: Generalized Voronoi diagrams of points and open, straight edges.

Voronoi Diagrams.

Since we focus on polygonal objects, it is convenient for our pourposes to consider that sets $\{s_i\}$ are made from points and open, straight segments, [Kir79]. They will represent respectively concave vertices and open, straight edges in the polygonal boundary. In this context, the generalized Voronoi diagrams associated with every possible pair of active boundary elements are (See Figure 2),

1. *Point-point* pair: straight bisector of the segment defined by the pair of points. Figure 2a.
2. *Segment-point exterior to the segment* pair: arc of parabola defined by the point and the segment. Figure 2b.
3. *Segment-endpoint of the segment* pair: perpendicular to the segment through the segment endpoint. Figure 2c.
4. *Segment-segment* pair meeting at a convex angle: straight bisector of the convex angle. Figure 2d.

We close this section with two definitions that will be used in our algorithm. Figure 3 illustrates de definitions.

Definition 2.1 Let e be an edge of a polygonal domain boundary. Let $V(e)$ be its Voronoi polygon and let (ve_i, ve_j) be the pair of edges in $V(e)$ through the endpoints of e. The *influence region* of e is the set of points bounded by edge e and the two halflines supporting the Voronoi edges (ve_i, ve_j).

Definition 2.2 Let v be a concave vertex on a polygonal domain boundary. Let $V(v)$ be its Voronoi polygon and let (ve_i, ve_j) be the pair of edges in $V(v)$ incident to the vertex v. The *influence region* of v is the set of points bounded by the two halflines supporting the Voronoi edges (ve_i, ve_j).

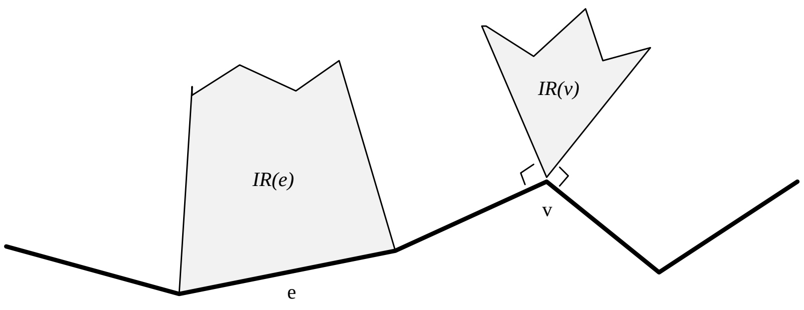

Figure 3: Influence regions of active boundary elements.

We shall denote by $IR(b)$ the influence region of the active boundary element b.

3 The Algorithm

The algorithm we propose here is capable of determining the MAT of arbitrary 2-D polygonal domains. The algorithm has three major parts: 1) Triangulation of the domain, 2) Labeling of the triangulation edges that are inside the domain, and 3) Adaptive refinement of the labeling resulting from step 2.

3.1 Domain Triangulation

The algorithm starts with the computation of a domain-compatible Delaunay triangulation, that is, a Delaunay triangulation that covers exactly the polygonal domain and such that the domain boundary is a subgraph of the Delaunay triangulation.

The set of points to be triangulated is formed by all the vertices in the boundary domain and a relatively sparse set of extra points generated on the boundary. The question of how many points have to be placed on the domain boundary in order to force it to be a subgraph of the Delaunay triangulation is still open, [Aur91]. In our experiments we have placed on each domain boundary edge two additional points, each point closer to each edge endpoint. The resulting set of points is then triangulated using a convenient method; we used the implementation reported in [Vig95] of the De Floriani and Puppo method, [FP92]. In the sequel we will call *interior edges* to those edges in the triangulation that do

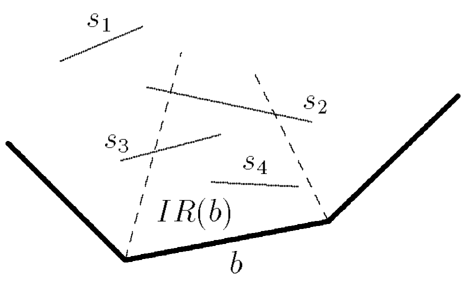

Figure 4: Splitting by Voronoi edges. Only that part of the segment inside $IR(b)$ needs to be considered.

not belong to the domain boundary.

3.2 Labeling interior edges

Those edges of the compatible domain triangulation that are interior are partitioned and labeled according to their closest active element in the domain boundary. This is achieved by classifying every interior edge with respect the influence region of every active domain boundary element.

Assume that a given segment s, belonging to one interior edge, has already been labeled with a closest active boundary element and that we want to update its labeling with respect to the active boundary element b. We start by classifying segment s with respect $IR(b)$. See Fig. 4.

If $s \cap IR(b) = \emptyset$ then we are done. Otherwise two different situations can arise. First case: Assume that $s \subset IR(b)$. If the whole segment s is closer to b than to the previous label, b is the new label of s. If only a part of s is closest to b, segment s is split and labeled according to the Voronoi diagram (bisector) of the pair of active elements formed from b and the current label of s. See Fig. 5. Second case: If s is not a subset of $IR(b)$ but their intersection is not empty, segment s is split by the Voronoi edges of $IR(b)$ into a list with a maximum of three segments with only one of them being inside $IR(b)$. Then the process above applies to the segment interior to $IR(b)$. Now edges in the triangulation can be labeled by traversing the triangulation and running the process above for each interior edge.

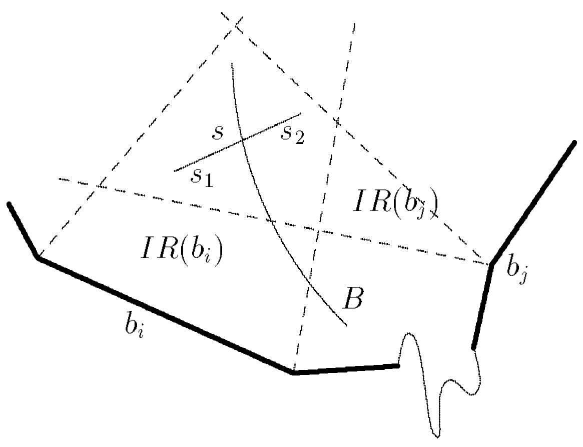

Figure 5: Splitting by B, the bisector of b_i and b_j. Segment s is inside $IR(b_j)$ but only subsegment s_2 is closer to b_j than to the current label, b_i.

3.3 Adaptive Refinement

Note that the interior edges in the triangulation along with their labeling define a coarse Voronoi diagram of the polygonal domain. This coarse Voronoi diagram can be refined by subdividing adaptively the triangulation. Since the procedure is recursive we begin with the recursion. Then starting the recursion from the coarse Voronoi diagram will be clear.

Let the triangle t, defined by points (p_1, p_2, p_3) in 6 be a triangle of the Delauney triangulation. Split triangle t into triangles t_1, t_2, t_3 and t_4 such that their vertices lie on the edges of triangle t and label each vertex according to the closest boundary element of the edge segment of triangle t where it lies.

It is easy to see that triangle t is crossed by a Voronoi edge if and only if at least two of its vertices have a different labeling. Then the algorithm has to traverse the triangulation and recursively subdivide all those triangles for which the Voronoi edge crossing condition holds, and label their vertices. The recursion stops when no branch of the Voronoi diagram crosses the triangle or when a triangle with a predefined size is reached.

Indeed the algorithm above will compute the Voronoi diagram. But given the Voronoi diagram of domain, the MAT can be computed by pruning the Voronoi edges defined by a segment-endpoint of the segment pair (See Fig. 2); that is, by pruning the Voronoi edges associated with every reflez vertex in the domain boundary. These Voronoi edges can be easily pruned on-the-fly. Let e, v be a

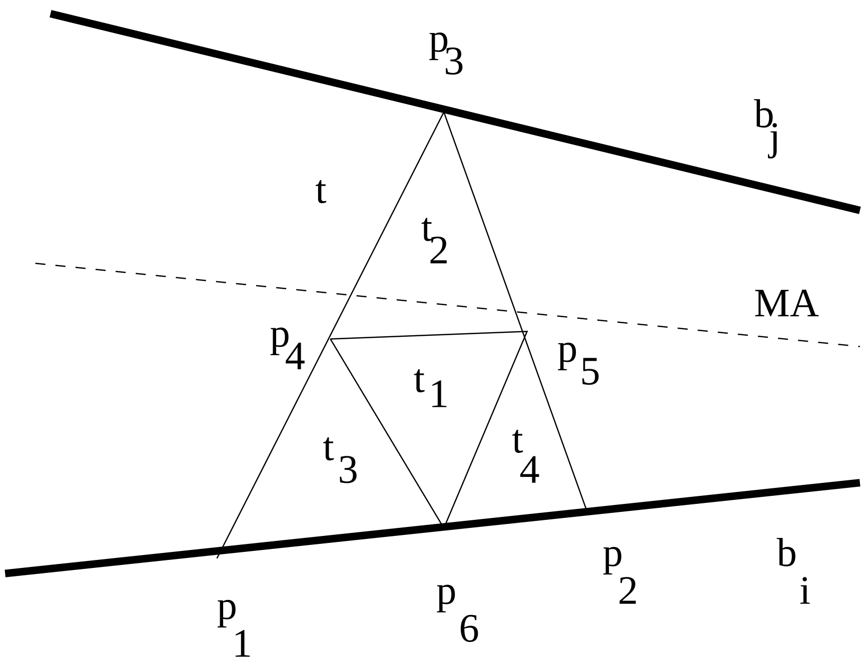

Figure 6: Triangles crossed by Voronoi edges.

segment-endpoint of the segment pair. Note that the vertices of the triangles crossed only by the Voronoi edge associated with the pair e, v are either in $IR(e)$ or in $IR(v)$. See Fig. 7. Then, these Voronoi edges are pruned just by stopping the recursion each time a triangle holding this property is found.

Several strategies to subdivide a triangle can be devised. The strategy currently applied splits each triangle into four triangles each of them defined by a vertex of the initial triangle and the two midpoints of the edges sharing that vertex. Labeling the sides of the triangles resulting from the subdivision is easely performed; sides can be labeled just taking into account the closest active boundary elements to the edges segments where they lay on.

4 Examples

Fig. 8 shows a simple E-shaped polygonal domain and a compatible Delaunay triangulation.

Fig. 9 shows the partition of the internal edges of the triangulation in Fig. 8.

This partition is the initial MAT. Pseudo code of the algorithm to compute

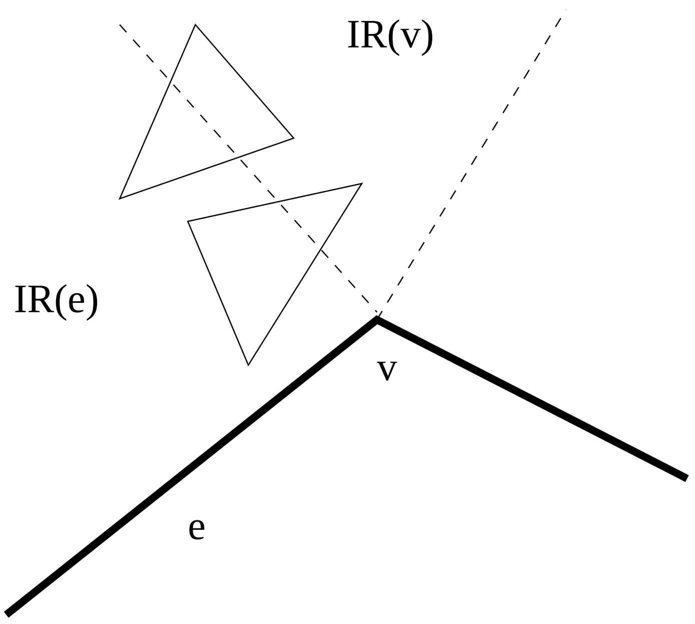

Figure 7: Triangles crossed only by one Voronoi edge associated with a reflex vertex in the domain boundary.

the initial MATis given in Appendix A. Fig. 10 shows the MATafter a few refinement steps performed by the refinement algorithm given in Appendix B. Finally, Fig. 11 shows the refined MATresultinf from the subdivision process for a given minimum size of the triangles.

A polygonal domain with holes and complex shape, and the results generated

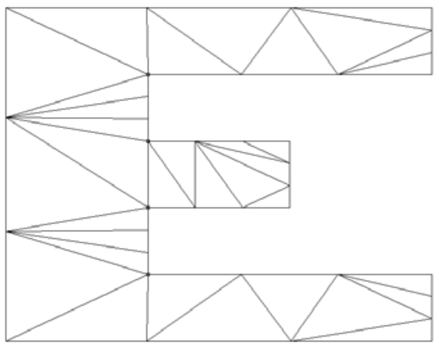

Figure 8: Domain boundary and compatible Delaunay triangulation.

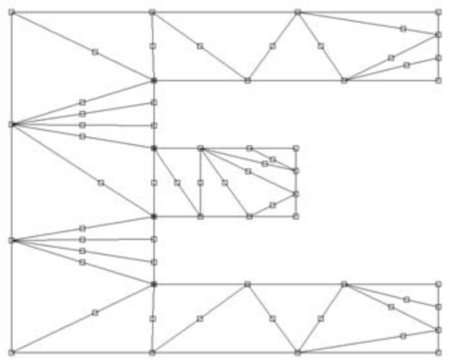

Figure 9: Initial partition of internal edges of the triangulation.

by the algorithm are shown in Figs. 12 to 14.

5 Conclusions

An algorithm to compute the MAT of polygonal domains in a 2-D Euclidian space has been introduced. The algorithm is based on the refiniment of an initial coarse MAT by subdividing the domain adaptively. The resulting algorithm is conceptually simple and easy to implement.

The same general ideas presented in this work would lead to an algorithm for

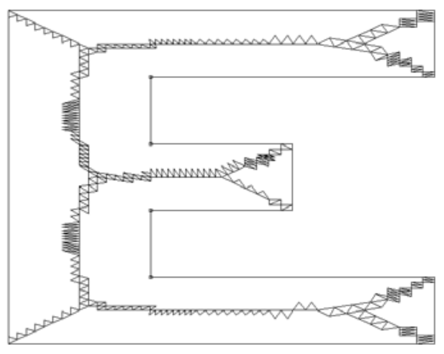

Figure 10: MAT at an intermediate step.

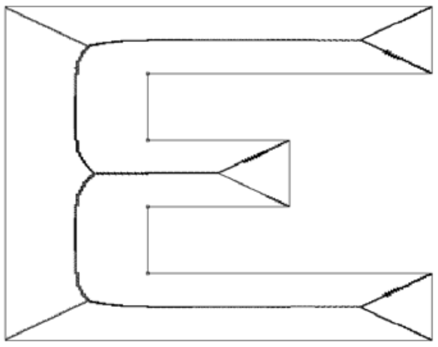
Figure 11: Refined MAT.

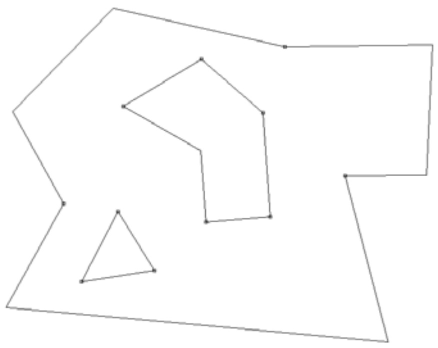
Figure 12: Complex shape. Domain boundary.

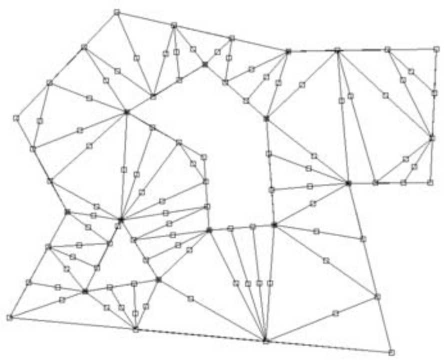

Figure 13: Complex shape. Compatible Delaunay triangulation and initial partition of internal edges.

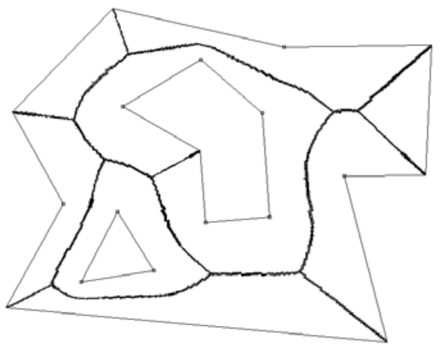

Figure 14: Complex shape. Refined MAT.

computing the MAT of polyhedra in 3-D where the Delaunay triangulation compatible with the domain boundary is replaced by the equivalent domain tetrahedrization in 3-D.

6 Acknowledgements

R. Joan-Arinyo was partially supported by a CIRIT fellowship of the Government of Catalonia under grant 1995BEAI400071.

References

[Aur91] F. Aurenhammer. Voronoi diagrams – a survey of fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Whaten-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, MA, 1967.

[FP92] L. De Floriani and E. Puppo. An on-line algorithm for constrained Delaunay triangulation. *CVGIP: Graphical Models and Image Processing*, 54(3):290–300, July 1992.

[GYKD91] J. A. Goldak, X. YU, A. Knight, and L. Dong. Constructing discrete medial axis of 3D objects. *Int. Journal of Computational Geometry and Applications*, 1(3):327–339, 1991.

[Kir79] D.G. Kirpatrick. Efficient computation of continuous skeletons. In *Proocedings of the 20th Annual Symposium of Foundations of Computer Science*, pages 18–27, October 1979.

[LBD$^+$92] D. Lavender, A. Bowyer, J. Davenport, A. Wallis, and J. Woodwark. Voronoi diagrams of set-theoretic solid models. *IEEE Computer Graphics and Applications*, 12(5):69–77, September 1992.

[PS85] F. Preparata and M. Shamos. *Computational Geometry.* Springer Verlag, New York, 1985.

[SAR95] D.J. Sheehy, C.G. Amstrong, and D.J. Robinson. Computing the medial surface of a solid from a domain Delaunay triangulation. In *Proc. ACM Symp. Solid Modeling Found. and CAD/CAM Applic.*, pages 210–212, Salt lake City, UT, 1995.

[SPB95] E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson. Computation of the medial axis transform of 3-D polyhedra. In C.M. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 187–199, Salt Lake City, Utah, 17-19 May 1995. ACM Press.

[Vig95] M. Vigo. An incremental algorithm to construct restricted Delaunay triangulations. Technical Report LSI-95-43-R, Universitat Politècnica de Catalunya, Department LiSI, 1995.

[Wol92] F.-E. Wolter. Cut locus and medial axis transform in global shape interrogation and representation. Memorandum 92-2, MIT Ocean Engineering Design Laboratory, January 1992.

A Pseudo Code to Compute the Initial MAT

Assume that the boolean function `OutRegion (s, b)` returns
tt true if segment `s` is totally outside $IR(b)$ and returns
tt false otherwise. Similarly, boolean function `InRegion (s, b)` returns `true` whenever `s` is totally inside $IR(b)$ and returns `false` otherwise.

```
procedure UpdateSegmentLabel (t_segment        s,
                              t_activeb        b,
                              t_listofsegments l )
  l := EMPTY
  if not OutRegion (s, b) then
    if InRegion (s, b) then
      UpdateInRegionSegment (s, b, l)
    else
      SplitSegment (s, b, l, in)
      UpdateInRegionSegment (l^in, b, l_aux)
      ReplaceSegmentByList (l, in, l_aux)
    endif
  endif
```

```
endprocedure

procedure UpdateInRegionSegment (t_segment        s,
                                 t_activeb        b,
                                 t_listofsegments l )
  if TotallyCloserToB (s, b) then
    LabelSegment (s, b)
    AddSegment (l, s)
  else
    if PartiallyCloserToB (s, b) then
      BisectSegment (s, b, l, in)
      LabelSegment (l^in, b)
    endif
  endif
endprocedure
```

Now, interior edges can be partitioned and labeled according to their closest active boundary element as follows.

```
procedure InitialMat ()
  FirstInteriorEdge (T, e, ok)
  while ok do
    LabelSegment (e, INFINITY)
    UpdateEdgeLabels (e, P)
    NextInteriorEdge (T, e, ok)
  endwhile
  CompactEdgesPartitioning ()
endprocedure

procedure UpdateEdgeLabels (t_segment e, t_polygon P)
  ListOfSegments := e
  FirstActiveBoundaryElement (P, b, ok)
  while ok do
    NewListOfSegments := EMPTY
    FirstSegment (ListOfSegments, s, ok1)
    while (ok1) do
      UpdateSegmentLabel (s, b, AuxList)
      NewListOfSegments := NewListOfSegements + AuxList
      NextSegment (ListOfSegments, s, ok1)
    endwhile
    ListOfSegments := NewListOfSegments
    NextActiveBoundaryElement (P, b, ok)
  endwhile
endprocedure
```

B Pseudo Code to Refine the Initial MAT

Let `WhiteTriangle (t)` be a boolean function that returns `true` whenever the three vertices of triangle `t` have the same closest active boundary element and `false` otherwise. Boolean function `ConcaveVertexBranch (t)` returns `true` if and only if triangle `t` is crossed by one Voronoi edge corresponding to a segment-endpoint of a segment pair. Assume that the final MAT will be stored in the static variable `mat`. The pseudo code for the algorithm that refines the intial coarse MAT follows.

```
procedure MatRefinement (t_triangulation T)
  for each triangle t in T do
    RefineTriangle (t)
  endfor
endprocedure

procedure RefineTriangle (t_triangle t)
  while not (WhiteTriangle (t) or MinimumSize (t)) do
    SubdivideTriangle (t, soft)
    LabelSides (sof)
    for each t in soft do
      RefineTriangle (t)
    endfor
  endwhile
endprocedure
```