Skeleton computation of an image using a geometric approach

J. Martínez[1], M. Vigo[1,2] N. Pla-Garcia[1,2] and D. Ayala[2]
[1]Universitat Politècnica de Catalunya (UPC). Barcelona. Spain
[2]Institut de Bioenginyeria de Catalunya (IBEC). Barcelona. Spain

March 15, 2010

Abstract

In this work we develop two algorithms to compute the skeleton of a binary 2D image. Both algorithms follow a geometric approach and work directly with the boundary of the image which is an orthogonal polygon (OP). One of these algorithms processes the edges of the polygon while the other one uses its vertices. Compared with a thinning method, the presented algorithms show a good performance. This is a work in progress as our final goal is to extend the vertex-based algorithm method to 3D in order to compute the surface skeleton of a binary volume.

Chapter 1

Introduction

One of the challenges of the BioCAD field is to understand the morphology of the pore space of bone, biomaterials, rocks, etc. Several approaches exist that obtain a network with pores and connections between them [VAGT08] or that characterize plate and rod elements [SM06], which rely on a previous computation of a surface skeleton.

In this work we follow a geometric approach to compute the skeleton of a binary 2D image using as input the orthogonal polygon (OP) that constitutes the image boundary. Our final goal is to obtain the surface skeleton of a binary volume by extending the proposed method. We have chosen this approach because geometric methods compared to thinning methods perform fewer steps as they do not need to work at voxel level. As real datasets obtained from μCT tend to be large, computing their skeleton with thinning approaches is very time-consuming. We have compared the proposed method with an optimized thinning approach and the obtained experimental results have been very encouraging, so we will extend our ideas to 3D.

We present two algorithms to compute the skeleton. Both are based on an existing method [PL01], which computes the Voronoi Diagram with L_∞ (VD_∞) of a general polygon. Both are intended to the particular case of OP. The first algorithm processes the edges of the polygon while the second one processes its vertices. The computed skeleton is composed by bisectors of the polygon edges which, in the particular case of OP, are vertical, horizontal and 45° edges. The worst case of method complexity for general polygons as well as that of the two presented methods is $O(n \log n)$ in time and $O(n)$ in space, being n the number of vertices.

Chapter 2

Related work

Methods to compute the skeleton can be classified into three main families: thinning, distance field and geometric. Most methods that apply to discretized data, i.e., images or volumes, are based on thinning and distance field approaches. Thinning methods consist in iteratively removing points from the object boundary. Distance field approaches rely on the previous computation of the distance field. These methods are linear in the number of voxels. Geometric methods are generally applied to polygons and polyhedra and they are based on the Voronoi diagram. The worst case of complexity is quadratic with the number of vertices. See [CSM05] for a survey on skeletonization methods.

The straight skeleton, introduced in [AA96], is a concept that coincides with the Voronoi Diagram with L_∞ (VD_∞) when applied to a boundary with axis-aligned edges. It consists in angular bisectors between edges (2D) or faces (3D). In [PL01] a method that applies to rectilinear polygons is developed and used to compute the critical area for shorts in VLSI. It is a sweep-line based approach that characterizes several types of events to deal with. In [SPK05] a VD_∞ is computed and applied to lithography path generation. This algorithm applies to general polygons without horizontal and vertical edges. Other approaches that compute the straight skeleton for polygons can be found in [EE99] and [TV03].

Concerning 3D algorithms, in [SPB96] the medial axis skeleton of general polyhedra without cavities is computed. They follow a tracing approach from the boundary elements inwards to the interior taking into account the encountered junction points. A shrinking process beginning at the boundary is also applied in the two methods presented in [BEGV08] to compute the straight skeleton of an orthogonal polyhedron. These methods characterize and deal with several types of events.

Chapter 3

Algorithm for general polygons

The L_∞ distance between two points a and b is the maximum between the horizontal and vertical distance between them i.e, $d_\infty(a,b) = max(d_x(a,b), d_y(a,b))$. Intuitively, the distance between two elements (points or lines) is defined by the smallest isothetic square touching the two elements. The L_∞ bisector of two elements can be regarded as the locus of centres of squares touching both elements.

Our method is based on the method presented in [PL01] that we outline here briefly. This method proposes a sweep algorithm to compute the Voronoi diagram (VD) of a planar straight-line graph G using the L_∞ distance. A vertical sweep-line L is swept from left to right.

At any instant the VD_∞ of the portion of G that lies on the left of L is computed. The boundary of the Voronoi cells of L is called the wavefront. It is a y-monotone polygonal line obtained by connecting the consecutive waves. A wave is sliding along two tracks which are either bisectors or intersected edges so that at any instant the endpoint of a bisector corresponds to the centre of the square defined by the sweep-line and the two elements inducing the bisector. The wavefront is stored in a height-balanced binary tree $\mathcal{T}$. There are two kinds of events that cause the wavefront to change: the occurrence of a point or a vertical edge (site event) and the intersection point of two neighbour spike bisectors (spike event). A priority queue referred to as an event list is used to store the events in increasing priority.

Let p_1 and p_2 be the two vertices of a vertical edge defining a site event (p_1 is above p_2). If a site event is defined by a point it is treated as a vertical edge with zero length. To process a site event, points v_1 and v_2 are computed as the intersection between the bisectors of p_1 and p_2 and the wavefront. Points v_1 and v_2 are found using a binary search in $\mathcal{T}$. Then the finalized portion of the wavefront between v_1 and v_2 is inserted in VD_∞ and deleted from $\mathcal{T}$. New spike bisectors emanating from every pair of edges incident to the site event edge

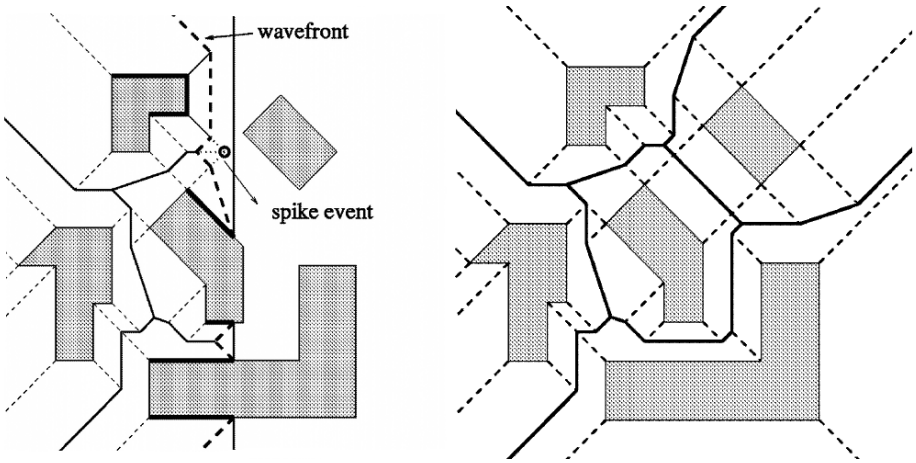

Figure 3.1: The L_∞ Voronoi diagram of general polygons. Wavefront status at the left before processing an spike event. Final Voronoi Diagram at the right. Extracted from [PL01].

and from new Voronoi vertices v_i are inserted in $\mathcal{T}$. Finally, for every pair of inserted spike bisectors we create new spike events induced by the intersection with neighbouring spike bisectors.

A spike event is the result of the intersection of two spike bisectors. It is also induced by three edges e_1, e_2 and e_3 because the two spike bisectors must share an edge e_2. The priority of a spike event is defined by the rightmost abscissa of the square defined by the two bisectors and the sweep-line. It is valid if the inducing bisectors are neighbours in $\mathcal{T}$. A valid spike event is a Voronoi vertex and the incident spike bisectors are Voronoi edges. A new spike bisector induced by e_1 and e_3 is generated and new spike events are inserted due to the intersection with neighbouring spike bisectors.

Chapter 4

Algorithms for orthogonal polygons

In this section we present two methods for orthogonal polygons based on the algorithm of [PL01]. The first processes the edges and the second the vertices. We get as output a Voronoi diagram. To obtain the straight skeleton we erase all the edges with a vertex of degree lower than two. Then we can rasterize in a straightforward way the skeleton because it is only composed of vertical, horizontal and 45° edges.

4.1 Edge-based algorithm

Several simplifications can be done to the previous algorithm when applied to the particular case of orthogonal polygons. Site events always correspond to the occurrence of a vertical edge. The two vertices that define an edge have a limited number of configurations (see Figure 4.1). Type 1 vertex configurations have the interior of the polygon at the right of the edge while type 2 have the interior at the left. Note that an edge cannot have two vertices of different types. At any instant if a vertex is of type 1 the outgoing spike bisector cannot intersect the current wavefront. Otherwise, if it is of type 2 we have to retrieve the intersected point of $\mathcal{T}$.

Our algorithm first classifies the edge as type 1 or 2. If it is of type 1 we generate two new spike bisectors induced by its vertices and insert new spike events due to the intersection with neighbour spike bisectors. If it is of type 2 we detect $v1$ and $v2$ in the wavefront and finalize the portion between them. Additionally, if we have a vertex of type 2.3 or 2.4 we generate a new spike bisector induced by the site event edge and the intersected edge of the wavefront and insert its neighbouring spike events.

A spike event will be defined by three wavefront elements and is valid if the inducing bisectors are still neighbours in $\mathcal{T}$. If it is valid a new spike bisector, defined by the two different wavefront elements of the two inducing bisectors of

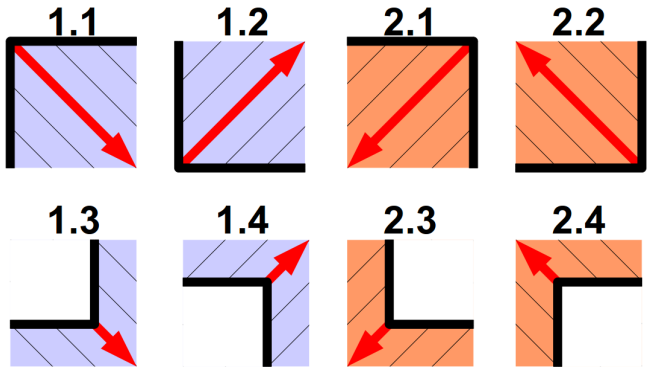

Figure 4.1: Possible vertex configurations with orthogonal polygons. Red line is the induced bisector and the coloured region is the interior of the polygon.

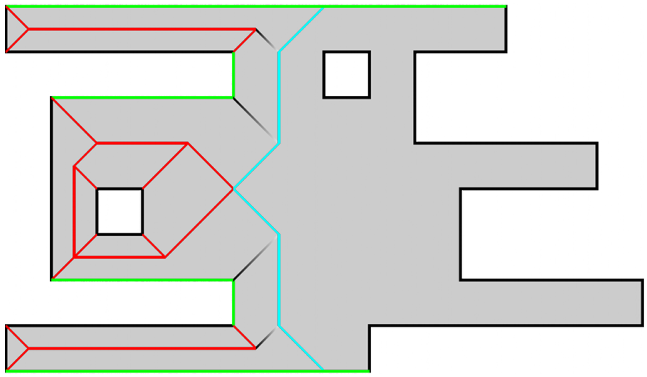

Figure 4.2: Induced wavefront in blue. Wavefront elements in green. Spike bisectors in grey. Current Voronoi diagram extracted in red using edge-based algorithm.

the spike event, is inserted in $\mathcal{T}$. Then new spike events are generated from the intersection with neighbouring spike bisectors.

Incorrect results may arise with non-manifold polygons when inserting spike bisectors induced by edges of type 1 that have the same abscissa. An example is shown in Figure 4.3. We have solved this problem by inserting spike bisectors of type 1 and with the same abscissa until a valid event has different abscissa. This solution also improves the performance because there will be less spike bisectors in $\mathcal{T}$ when processing a collection of vertical collinear edges.

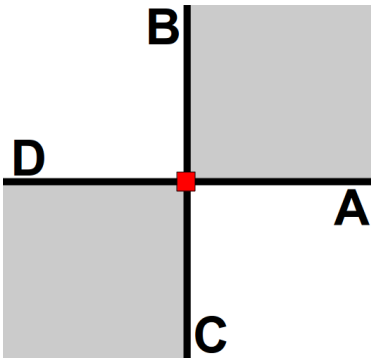

Figure 4.3: Non-manifold vertex marked in red. We may insert first the spike bisector defined by edges A and B. Then, when we retrieve the intersection of the edge C the spike bisector AB collides in the intersection search returning an invalid wave.

4.2 Vertex-based algorithm

The second method that we have developed is based on the previous one (Section 4.1) that processes vertices instead of edges. This new approach simplifies the algorithm because all the events will be defined by a single vertex. The event treatment will be different depending if it is a vertex of the polygon or an spike event vertex. The main idea is to progressively generate single vertex-based events that will finalize the VD. The events are sorted lexicographically in the event list. The priority of a site event is defined first by its abscissa and then by its ordinate. A spike event is defined by two spike bisectors and the priority is defined by the left-most abscissa and lowest ordinate of the square defined with the sweep-line.

This vertex-based algorithm must support the insertion of spike bisectors in the wavefront that do not define a unique square with the sweep-line. We refer them as temporal spike bisectors because they only appear when finalizing a portion of the intersected wavefront. We need a criterion to sort these temporal spike bisectors in $\mathcal{T}$. As they are always associated to an spike event we take the square defined by the current associated spike event to classify them in the wavefront.

The event processing method is briefly shown in the Algorithm 1. To process a site event if the vertex is of type 1 we insert the induced spike bisector and retrieve neighbour spike events. If it is of type 2.1 or 2.2 and the intersected spike bisector is equal to the spike bisector induced by the site vertex we just add an edge between them. Otherwise, we insert the induced spike bisector and create a spike event marked as final with the intersected wave. If it is of type 2.3 it is marked as initial and if it is of type 2.4 it is marked as final (see Figure 4.4).

To process a spike event we tune the direction (upper or lower neighbour in $\mathcal{T}$) where we try to find neighbouring spike events. We take into account the

Algorithm 1 Algorithm to process an event using vertex-based approach

```
{E is the event list}
E ← Site events induced by input polygon sorted lexicographically
while E ≠ ∅ do
  e ← first(E)
  if e = site event then
    w ← wave induced by e
    Add site vertex to VD∞
    if e = type 1 then
      Insert w in T and insert neighbour spike events in E
    else if e = type 2 then
      i ← intersected element in T by w
      Insert w in T
      if (e = type 2.1 or e = type 2.2) and (i = w) then
        Add a Voronoi edge between i and w
      else if e = type 2.3 then
        Insert an initial spike event induced by i and w in E
      else if e = type 2.4 then
        Insert a final spike event induced by i and w in E
      end if
    end if
  else if e = spike event then
    w_a ← site event wave inducing spike event
    w_b ← intersected wave inducing spike event
    w_o ← outgoing wave induced by w_a and w_b
    Add spike event vertex v_s to VD∞
    Add a Voronoi edge from w_a and w_b to v_s
    Delete w_a from T
    Insert w_o in T
    if e = initial spike event then
      Insert upper neighbour spike events of w_b
      Insert lower neighbour spike events of w_o
    else
      Delete w_b from T
      if e = final spike event then
        Insert upper neighbour spike events of w_o
      else
        Insert neighbour spike events of w_o
      end if
    end if
  end if
end while
```

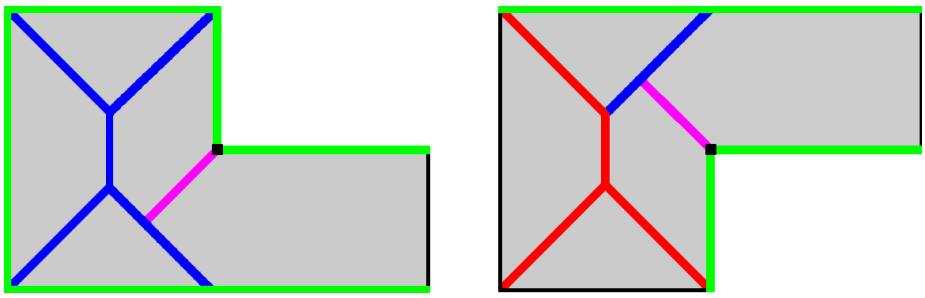

Figure 4.4: Spike bisectors and induced wavefront (blue edges), intersecting bisector (purple edge), finished Voronoi diagram (red edges) and edges inducing wavefront (green edges). Initial spike event on the left polygon and final on the right one.

junction points of the intersections and the swept direction (from left to right). If the spike event is not marked as initial or final, the process is identical to that related in Section 4.1. Otherwise, let A be the site event wave and B the intersected edge wave. If the spike event is an initial spike event, we only search for upper neighbour spike events of B and lower spike events of the outgoing spike bisector. If it is a final spike event we only search for upper neighbour spike events of the outgoing wave. By labeling spike events (as initial and final) we avoid the creation of new spike events with less priority than the current event.

4.3 Repairing arbitrary oriented Voronoi edges

When two edges are collinear along the vertical or horizontal axis when using L_∞ the induced bisector is an entire area (see Figure 4.5). We resolve this by choosing the boundary of that area as the bisector. An equidistant region is assigned to one of the two points or segments arbitrarily. In our case the first element processed by the sweep-line algorithm. Depending on the direction of sweep-line process and the OP orientation we may have different results. This is correct in the sense of L_∞ metric although the output skeleton is not unique. We solved this problem by selecting the central bisector of the bisector area. The central bisector is the bisector defined by the two boundary bisectors of the area.

The selection of the central bisector is done in a post-processing step. We only need to process the graph. We see that an arbitrary edge induced by a collinear pair of edges have a common characteristic in the final VD_∞ (see Figure 4.5). For every vertex of VD_∞ that has to be repaired, we erase the arbitrary edge and find the intersection of the central bisector with VD_∞ (see Algorithm 2). We see in the Figure 4.6 the post-processing done in an input

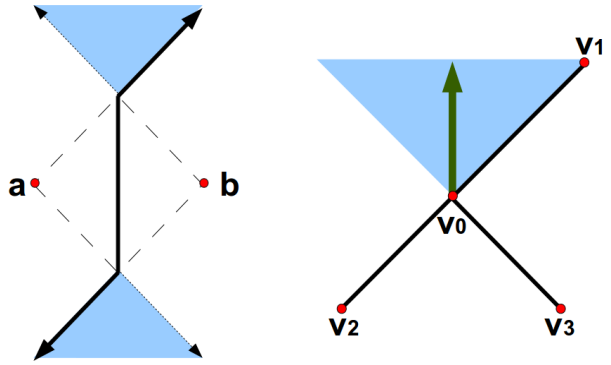

Figure 4.5: On the left there are two horizontal collinear points a and b inducing a bisector area shown in blue. On the right, a case of an arbitrarily selected edge. The vertex v_0 is linked with v_1 and v_3 (not with v_2). The vertex v_1 is also linked with v_2. We detect that v_0 is contained in the edge defined by v_1 and v_2. The central induced bisector is shown in green.

OP. We repair all the ambiguities induced by the collinear segments obtaining a unique solution that is independent of the sweep direction and the skeleton extraction technique.

Algorithm 2 Algorithm to repair an arbitrary oriented Voronoi edge

{Let v_0, v_1, v_2 and v_3 be the detected vertices as related in Figure 4.5}
Remove edge (v_1, v_2)
Remove edge (v_1, v_0)
Add edge (v_0, v_2)
Remove vertex v_1
$v_{i-1} \leftarrow \emptyset$
$v_i \leftarrow v_1$
repeat
  $v_i \leftarrow v_{i-1}$
  $v_i \leftarrow$ next inner vertex of v_i
until edge (v_i, v_{i-1}) not intersects the central bisector defined by v_2 and v_3
$v_n \leftarrow$ vertex intersected by central bisector and edge (v_i, v_{i-1})
Add edge (v_n, v_{i-1})
Add edge (v_n, v_i)
Remove edge (v_i, v_{i-1})

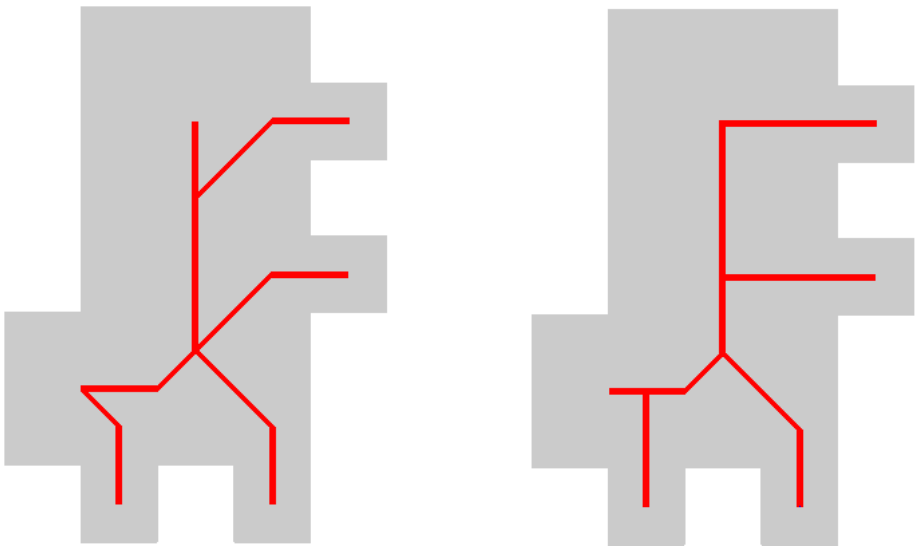

Figure 4.6: In the left, the skeleton obtained without modifying the arbitrary edges induced by collinear edges. In the right the repaired graph using the our method. In gray the input orthogonal polygon and in red the straight skeleton obtained.

Chapter 5

Experimental Results

We have compared the presented geometric approaches with an optimized thinning approach [AVV07] which we are currently using to devise pore network morphologies. The three algorithms have been implemented in C on a PC Intel E6600 2.40 Ghz with 3.2 GB RAM. Table 5.1 shows the results for six datasets that are shown in Figure 5.1. Datasets Lizard and Random are phantom; Newspaper is a public image. Dataset Biomaterial is a slice of a sample of biomaterial for bone regeneration and both Rock datasets are a slice of a sample of a sedimentary rock (Rock1 is the solid space and Rock2 the porous space). We show the running times of the vertex-based approach as this is the approach that we are going to extend to 3D. Nevertheless, the running times of the edge-based approach are almost equal to those of the vertex-based approach. These times include the rasterization time overhead which is almost negligible. We also show the overhead due to the repairing of collinear ambiguities. It represents at most of 2% of the total time.

Model	**Size Voxel**	**Size #Vertices**	**Time Thin**	**Time Geom**	**Time Repair**
Lizard	430×466	1536	7.8	0.07	$\sim$0
Random	100×100	7554	3.22	0.24	0.01
Biomaterial	357×356	11588	14.5	0.45	0.01
Rock1	474×811	18330	45.4	0.81	0.03
Rock2	474×811	18334	24.4	0.67	0.03
Newspaper	1615×2251	244528	155	10.29	0.21

Table 5.1: Results for several datasets. Values shown are model size, number of vertices and running times in seconds.

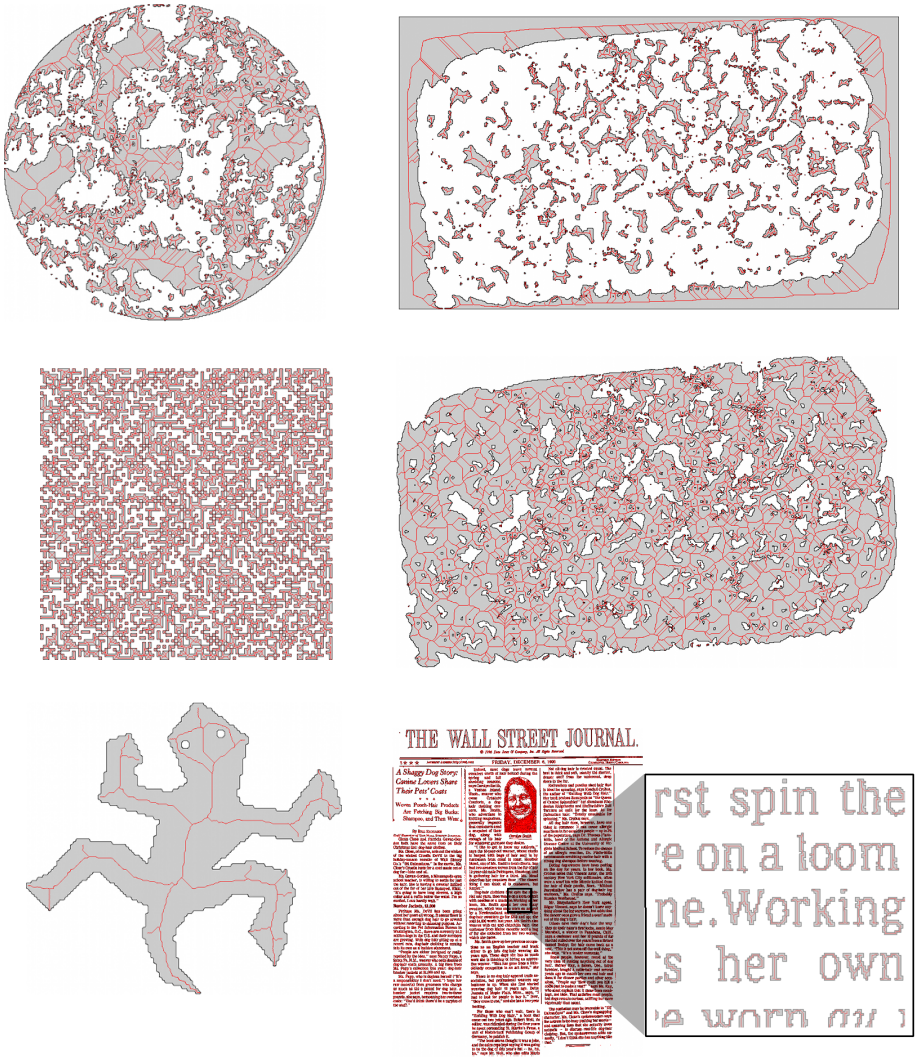

Figure 5.1: Tested datasets. From top to down and left to right: Biomaterial, Rock2, Random, Rock1, Lizard and Newspaper. Straight skeleton shown in red and input polygon shown in grey.

Chapter 6

Conclusions and future work

We have presented an edge-based and a vertex-based algorithms to compute the straight skeleton of an orthogonal polygon. Both methods can deal with the particular case of non-manifold vertices and collinear edges which are very usual in an image boundary. Collinear edges are treated in a post-processing step solving the implicit ambiguity of L_∞ distance [PL01].

The good performance of the presented methods compared to the thinning approach has encouraged us to continue, as immediate future work, with the extension of the vertex-based approach to orthogonal polyhedra. We have chosen the vertex-based approach because it has less configurations and therefore is easier to extend to 3D, since it does not implies to deal with faces, which in 3D may have a high number of vertices.

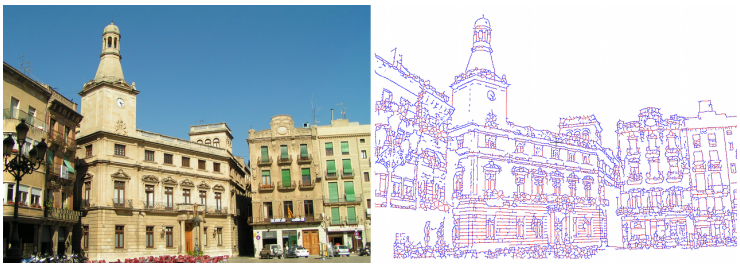

Figure 6.1: Original image on the left (a view of the city hall of Reus) and skeleton extracted from the binarized image on the right.

Chapter 7

Acknowledgements

This work has been partially supported by the project TIN2008-02903 of the Spanish government and by the IBEC (Bioengineering Institute of Catalonia).

Bibliography

[AA96] Aichholzer O., Aurenhammer F.: Straight skeletons for general figures in the plane. In *LNCS 1090* (1996), pp. 117 – 126.

[AVV07] Ayala D., Vergara E., Vergés E.: Improved skeleton computation of an encoded volume. In *Proc. of Eurographics* (2007), pp. 33–36.

[BEGV08] Barequet G., Eppstein D., Goodrich M. T., Vaxman A.: Straight skeletons of three-dimensional polyhedra. *LNCS 5193* (2008), 148 – 160.

[CSM05] Cornea N. D., Silver D., Min P.: Curve-skeleton applications. In *IEEE Visualization* (2005), pp. 23 – 28.

[EE99] Eppstein D., Erickson J.: Raising roofs, crashing cycles and playing pool: applications of a data structure for finding pairwise interactions. *Discrete & Computational Geometry 22*, 4 (1999), 569 – 592.

[PL01] Papadopoulou E., Lee D. T.: The l_∞ Voronoi diagram of segments and VLSI applications. *Int. J. Computational Geometry & Appl. 11*, 5 (2001), 503 – 508.

[SM06] Stauber M., Müller R.: Volumetric spatial decomposition of trabecular bone into rods and plates: A new method for local bone morphometry. *Bone 38*, 4 (2006), 475 – 484.

[SPB96] Sherbrooke E. C., Patrikalakis N. M., Brisson E.: An algorithm for the medial axis transform of 3-D polyhedral solids. *IEEE Trans. on Visualization and Computer Graphics 2*, 1 (1996), 44 – 61.

[SPK05] Shin H., Park E., Kim D.: Voronoi diagram of a polygon in chessboard metric and maskless lithographic applications. In *Voronoi Symposium* (2005).

[TV03] Tănase M., Veltkamp C.: Polygon decomposition based on the straight line skeleton. In *Proc. 19th ACM Sym. Computational Geometry* (2003), pp. 58 – 67.

[VAGT08] Vergés E., Ayala D., Grau S., Tost D.: Virtual porosimeter. *Computer-Aided Design and Applications 5*, 1-4 (2008), 557–564.