IMESH: GENERATING QUALITY MESHES FROM IMAGES

Alex Jesus Cuadros-Vargas*, Luis Gustavo Nonato*

*Universidade de Sao Paulo, Instituto de Ciencias Matematicas e de Computacao
Av. Trabalhador Sao-Carleuse, 400, POBox 668
13560-970, Sao Carlos - SP, Brasil
e-mail: {alexj,gnonato}@icmc.usp.br

Key words: Image Mesh Generation, Mesh Segmentation, Delaunay Triangulation

Abstract. Techniques devoted to generate a triangular mesh from images either take as starting point a segmented image or generate a mesh without distinguishing different structures contained in the image. The need for pre-segmentation and the absence of well defined structures may rule out the use of the resulting mesh in some applications, as numerical simulations. In this work we present an novel algorithm, named imesh, which can automatically generate a mesh from a given image while identifying different regions and maintaining good quality of the triangular elements. The regions are identified by building a segmentation strategy into the mesh generation process, thus avoiding a pre-processing step. The quality of the mesh is obtained by adapting Ruppert’s algorithm to work in the “segmented” mesh. Therefore, the proposed algorithm assembles two important aspects, namely, the automatic generation of a mesh that distinguishes different structures in the image and the generation of good quality mesh elements.

1 INTRODUCTION

Numerical simulation in domains defined from images has stimulated the technological development in many branches of science. Typical examples are blood flow simulation, elastic deformation of organic structures, and studies of structure cracks and their propagation. One of the main drawbacks in this context is to generate a mesh that fits features of interest contained in the images while being adequate for numerical simulation.

Opposite to the classical mesh generation problem, little has been achieved towards automating the process of generating a mesh directly from image data. Most algorithms strongly rely on extensive pre-processing steps, which typically consist in segmenting regions of interest from the images, using the boundaries of such regions as input to a mesh generator. Although largely employed, such an approach demands a stressful user intervention and specific segmentation softwares in order to obtain well defined models.

Techniques that act directly on images, in general, line up the mesh with the image features, but not worry about segmenting structures contained in the images. To ensure good quality meshes is another problem of such techniques. The identification of distinct
structures in the image is essential in applications such as multi-fluid flows\textsuperscript{15} and fluid-structure interaction\textsuperscript{2}, since these applications take into account the interface among regions to define interfacial strengths. Additionally, good quality meshes are primordial for accurate and efficient numerical simulation of natural phenomena modeled by partial differential equations. Therefore, in order to be effective, any mesh generator devoted to numerical applications in domains defined from images must be concerned with both aspects.

In this work we present a novel algorithm, named *Imesh*, which aims at sorting out both aspects discussed above, that is, generating a good quality mesh that respects image features while still identifying different regions. The regions are identified by building a segmentation strategy into the mesh generation process. In fact, this segmentation first guides an initial refinement step by tracking the boundaries of the regions contained in the image. Then, it groups such regions by employing either a labelling strategy or a region growing strategy. The quality of the mesh is obtained by adapting Ruppert’s algorithm\textsuperscript{12} to work in the “segmented” mesh.

As we shall show in Section 5, the mesh produced by our framework turns out effective for numerical simulation as well as image mesh modeling. Before discussing the results we present, in Section 2, some related work on image based mesh generation. Basic definitions necessary to a better understanding of *Imesh* are given in Section 3. *Imesh* is detailed in Section 4. Conclusions and future work are given in Section 6.

2 RELATED WORK

Some techniques devoted to generate a mesh straightly from an image aim at representing the image by a set of triangular cells. Such techniques, also called mesh modeling, intend to build a mesh that minimizes the approximation error between the original image and the image represented by the triangular mesh. Garcia et al.\textsuperscript{6}, for example, have presented an algorithm that controls the maximum root-mean-square error (RMS) by choosing the vertices of the mesh from a curvature image, that is, more vertices are placed in regions with high curvatures. The mesh is built by generating the Delaunay triangulation\textsuperscript{5} from the chosen vertices. Regions with high RMS error are resampled and the Delaunay triangulation updated. Garcia’s method is a typical example of an adaptive approach, which is characterized by begining with an initial mesh that is iteratively refined in order to reduce the interpolation error. Many algorithms devoted to represent images by meshes are based on adaptive approaches\textsuperscript{10,8,7}. Alternatively, some techniques have adopted an opposite strategy, i.e., a fine mesh is successively coarsened until the approximation error reaches a tolerance\textsuperscript{4}. Mixed approaches that combine refinement and coarsening\textsuperscript{11} as well as optimization for re-positioning the vertices in minimum error places have also been developed\textsuperscript{16}. Still envisioning mesh modeling, Yang et al.\textsuperscript{17} proposed an one pass method that makes use of zero-crossing to detect edges jointly with an error diffusion algorithm to choose a set of vertices from which the Delaunay triangulation is built. Besides reducing the approximation error, the authors argue that this strategy produces meshes of good
Another class of image based mesh generation techniques are concerned with numerical simulation. These approaches, in general, divide the process in two main steps: pre-processing and the mesh generation itself. The pre-processing step aims at filtering and segmenting the image in order to detect regions of interest, which are “meshed” in the mesh generation step. Cebral and Lohner for example, binarize the original image in order to extract well defined contours from which a mesh is generated. Binarization has also been employed as a pre-processing strategy by Zhang et al. and Berti. In both algorithms the mesh is built by defining, from the binary image, an implicit function that guides an space partitioning strategy. They also add a post-processing step to improve the quality of mesh elements.

A different approach has been presented by Hale. Hale’s algorithm makes use of a pre-processing step to reduce noise and highlight sharp features. Following that, a potential energy function is employed to align a lattice of points with the image features. The mesh is finally generated by Delaunay triangulation from the moved points. The main problem with Hale’s strategy is that distinct regions are not identified, impairing the definition of boundary conditions in applications as multiphase flows.

The method proposed in this work differs from the techniques described above in two main aspects: the segmentation strategy, commonly employed as a pre-processing step, is built into the mesh generation process. Furthermore, a theoretically guaranteed good quality mesh is produced at the end of the process. The algorithm can also distinguish different structures contained in the image, producing a set of conforming independent submeshes, where each submesh models a structure contained in the original image.

3 BASIC CONCEPTS

In this section we present some basic definitions and terminology used in the remaining of the text.

Let $S$ be a set of points in $\mathbb{R}^2$. A triangulation (mesh) of $S$ is a two-dimensional simplicial complex $M$ whose vertices are the points of $S$, and any $k$-simplex of $M$, $k = 0, 1$, is contained in at least a 2-simplex (triangle) of $M$. If the union of all simplices in $M$ makes up the convex hull of $S$ and the circumcircle of each triangle in $M$ does not contain any point of $S$ in its interior then $M$ is called Delaunay triangulation. Delaunay triangulation is closely related with the Voronoi diagram, which can be obtained by associating to each $k$-simplex of the triangulation an $2 - k$ cell in the diagram. In fact, each vertex of the diagram is the circumcenter of a Delaunay triangle and each vertex $v_i$ of the triangulation is contained in a two-dimensional cell of the diagram that contains the points in $\mathbb{R}^2$ closest to $v_i$.

A good quality mesh is a triangulation where all triangles satisfy a measure of quality, as for example circumradius-to-shortest edge ratio, i.e., the ratio between the radius of the circumcircle and the length of the shortest edge is limited by a constant in all triangles of the mesh.
A planar graph is a graph $G$ with vertices in $\mathbb{R}^2$ where each edge is a straight-line segment with ends in $G$ and if $e_1$ and $e_2$ are two edges of $G$, $e_1 \cap e_2$ is either empty or a vertex of $G$. Given a planar graph $G$, a mesh conformed by $G$ is a triangulation $M$ where each vertex of $G$ is in $M$ and if $e$ is an edge in $G$ then $|e| = |e_1 \cup e_2 \cup \cdots \cup e_k|$, where $e_i \in M$, $i = 1, \ldots, k$ are edges of $M$ and $| \cdot |$ represents the underlying space, i.e., each edge of $G$ can appear subdivided as a set of edges in $M$.

Let $S$ be a set of points and $M$ be a mesh (triangulation) of $S$, if $M = M_1 \cup M_2 \cup \cdots \cup M_k$, where each $M_i$ is a triangulation and $M_i \cap M_j$, $i \neq j$ is either empty or a planar graph then $\{M_1, M_2, \cdots, M_k\}$ is said a $k$-partitioning of $M$ in submeshes $M_i$, $i = 1, \ldots, k$.

An $m \times n$ image is a function $I : [0, \ldots, m] \times [0, \ldots, n] \rightarrow \mathbb{R}^+$ that assigns to each point $p \in [0, \ldots, m] \times [0, \ldots, n] \subset \mathbb{Z}^2$ a non-negative scalar $I(p)$. The pair $(p, I(p))$ is called pixel.

4 THE IMESH METHOD

The Imesh technique is divided in three main steps: initialization, partitioning, and mesh improvement. Initialization step concerns to the process of generating an initial mesh that fits some features contained in the image. Partitioning aims at segmenting the initial mesh in submeshes, thus defining regions of interest. The final step, mesh improvement, refines the submeshes so as to produce a quality mesh. A detailed description of each step is presented in the following.

4.1 Initialization

Initialization concerns to the process of generating a mesh (triangulation) that fits features contained in the image. In our context, we aim at avoiding that the triangles cross different regions of the image, that is, each triangle should be spatially contained in only one region of the image.

Let $T$ be the set of triangles of a Delaunay mesh $M$ whose vertices are points of an image $I$ and $E : T \rightarrow \mathbb{R}^+$ be a function that associates an error measure to each triangle in $T$. In fact, function $E$ measures how good a triangle is regarding a specific property, that is, $E$ enables to decide whether or not a triangle must belong to the triangulation.

Different strategies to define the function $E$ have been presented in the literature, but usually such approaches rely on evaluating $E$ by traversing all pixels inside a triangle $t$ so as to decide, based on some characteristic of the image, whether or not $t$ is an appropriated triangle. In general, when $E$ indicates that $t$ is a bad triangle, the triangulation is updated by inserting new points within $t$, thus eliminating it from $M^7$.

Although widely employed, the bad triangles removal strategy described above presents two main drawbacks. Traversing all pixels within a triangle may demand a high computational cost, as each time the triangulation is updated all the new triangles must be scanned in order to evaluate $E$. Another problem is concerned with the insertion of new points in the triangulation, which, when not handled properly, can result in an accumulation of
points around already existing vertices.

To avoid the problems described above, we adopt a strategy based on the medians of the triangles to define the function $E$. By traversing medians only, one can reduce the computational effort while being effective in detecting triangles that go across different regions of the image.

Let $h_1, h_2, h_3$ be the three medians of a triangle $t \in M$. Consider the sets of points $P^{h_j} = \{ p \in h_j \mid \mathcal{E}(p) \geq c_{\mathcal{E}} \}$, $j = 1, 2, 3$ where $\mathcal{E}$ is an edge detection operator and $c_{\mathcal{E}}$ is an user defined scalar. Therefore, $P^{h_j}$ is the set of points where the edges of the image intersect the median $h_j$.

Let $\alpha_i$, $i = 1, 2, 3$ be the barycentric coordinates of a point $p_k \in P^{h_j}$ and $A(p_k) = \min\{\alpha_i\}$ be a function that associates to each $p_k$ its smallest barycentric coordinate. Let $D_M(p_k)$ be the square distance between $p_k$ and its closest vertex in $M$, that is, $D_M(p_k) = \min_{v_j \in M} \{d^2(p_k, v_j)\}$, where $d(\cdot, \cdot)$ is the euclidean distance. Denoting $p_{h_j}$ the point of $P^{h_j}$ where $D_M(p_k)$ is maximal, we can define the error function $E$ as follows:

$$E(t) = \max\{A(p_{h_j})\}, \quad j = 1, 2, 3$$  \hspace{1cm} (1)

The barycentric coordinates of a point $p_k$ is related with the areas of the triangles formed by $p_k$ and the vertices of the triangle that contains $p_k$. Therefore, $A(p_{h_j})$ measures how much the area of a triangle $t$ is enclosed within a region of the image. A small value of $A(p_{h_j})$ indicates that $h_j$ intersects an edge of the image close to the boundary of the triangle. Thus, values of $E(t)$ close to zero indicate that $t$ is well fitted within a region in the image. Hence, a triangle $t$ is considered unsuitable if $E(t) > c_E$, where $0 \leq c_E \leq 1$ is an user defined scalar.

Unsuitable triangles are eliminated by inserting, in the Delaunay triangulation, the point $p_{h_j}$ such that $E(t) = A(p_{h_j})$. Since $p_{h_j}$ are points chosen to be as far as possible from the vertices of $M$, the problem of dense accumulation of points around existing vertices is reduced.

### 4.2 Partitioning

The partitioning step aims at generating a $k$-partitioning of the mesh produced in the initialization step, where $k$ is previously specified. Such partitioning is carried out by a region growing approach, which is accomplished as described below.

Let $\mathcal{H}(M)$ be a mapping that associates an array of characteristics to a mesh (or submesh). $\mathcal{H}(M)$ may be an array of texture features, histogram of the pixels in $|T|$, or just the average of $I$ in $|T|$.

The partitioning is started by setting each triangle as a submesh and arbitrarily select a submesh (triangle) $M_i$. For each submesh $M_j$ sharing a segment (edge) with $M_i$ we compute $|\mathcal{H}(M_i) - \mathcal{H}(M_j)|_\mathcal{H}$, where $|\cdot|_\mathcal{H}$ is a distance measure defined from $\mathcal{H}$. If $|\mathcal{H}(M_i) - \mathcal{H}(M_j)|_\mathcal{H}$ is smaller than an user defined parameter $\text{min}_\text{diff}$ then $M_j$ and $M_i$ are merged together. Therefore, $M_i$ “grows” until there is no neighbor submesh to be
merged with \( M_i \). A new submesh \( M_k \neq M_i \) is then selected and the process is repeated. In the end we have a partitioning \( M = M_1 \cup M_2 \cup \cdots \cup M_m \).

If \( m > k \) we must merge some submeshes in order to obtain a \( k \)-partitioning. This final merge is carried out taking into account the area of the submeshes. Adding the submeshes to a priority queue sorted by area, the smallest submesh is merged with the “closest” submesh regarding the distance \( \| \cdot \|_\varphi \). The process carries on until the number of submeshes is \( k \). Using the priority queue avoids sorting the submeshes every time a merge is executed.

4.3 Mesh Improvement

This final step of the algorithm aims at refining the submeshes in order to produce a triangulation whose triangles respect a minimum angle criterion. A variant of Ruppert’s algorithm\(^{12}\) is employed to achieve such a good quality triangulation.

Ruppert’s algorithm refines a Delaunay mesh by inserting the circumcenters of “poor” quality triangles. The quality of the triangles is measured by the circumradius-to-shortest edge ratio, i.e., the radius of the circumcircle divided by the length of the shortest edge of the triangle. It can be proven that the circumradius-to-shortest edge ratio \( r/d \) of a triangle is related to its smallest angle \( \alpha \) by \( \sin \alpha = d/(2r) \). As the insertion of the circumcenters tends to generate triangles with smaller circumradius, the smallest angle of the new triangles tends to be bigger than the old ones, thus improving the quality of the triangulation. Ruppert’s algorithm inserts circumcenters until all triangles satisfy a quality constraint, i.e., all triangles have the ratio \( r/d \) limited by a constant.

The strategy to insert new vertices in Ruppert’s algorithm is governed by two main rules thus described: let \( G \) be a planar graph and \( M \) be a mesh whose vertices of \( G \) are vertices of \( M \). The first rule of Ruppert’s algorithm verifies, for each segment (edge) \( e \) in \( G \), if a vertex of \( M \) lies strictly inside of the diametral circle (the smallest circle enclosing the segment) of \( e \). In the affirmative case, the segment \( e \) is split in two segments by inserting a vertex at its midpoint. The process follows until the diametral circles of the segments (or subdivided segments, called subsegments) in \( G \) are empty. The second rule aims at inserting a vertex at the circumcenter of each triangle whose circumradius-to-shortest edge ratio is greater than a bound \( B \). However, if the new vertex lies inside of the diametral circle of some segment (or subsegment) of \( G \), then such a vertex is not inserted and the segment is split as in the first rule.

It can be shown that if \( B \geq \sqrt{2} \) then Ruppert’s algorithm terminates; furthermore, if a circumcenter \( v \) is inserted as a new vertex then \( v \) lies inside the planar graph bounding the mesh\(^{13}\).

We generalize Ruppert’s algorithm, originally designed for a single planar graph, to work on a \( k \)-partitioning. The main problem in such a generalization is the fact that new vertices inserted on the segments of the planar graphs bounding submesh may eliminate other subsegments, damaging the partitioning. We overcome this problem by “locking” the segments of the planar graphs. Although the triangulation may become non-Delaunay
momentarily, it is not difficult to show that the Delaunay property will be recovered as locked segments have also diametral circles not empty and thus will be subdivided. As circumcenters are guaranteed to lie inside the planar graphs, their insertion do not affect the partitioning.

5 RESULTS

In this section we present examples of meshes produced by \textit{Imesh} algorithm. It is worth mentioning that the images considered as input have not been pre-processed, i.e., they are input directly into the algorithm. Average gray level and euclidean distance have been used to define $H$ and $| \cdot |_H$.

Figure 1 illustrates the result of each step of \textit{Imesh} when meshing the mechanical piece shown in figure 1a). Figure 1b) is the mesh produced by the Initialization step. The triangles have been colored from the average color of the pixel within them. The result of partitioning the initial mesh in three submeshes is shown in figure 1c). The final mesh obtained from the Mesh Improvement step is depicted in figure 1d).

Figure 2 shows the behavior of \textit{Imesh} in a medical image. In this example we set the algorithm to generate a 2-partitioning, aiming at distinguish the vein (figure 2a)) from the background of the image. Notice in figure 2b) that even thin branches of the vein have been detected properly, showing that \textit{Imesh} is able to model small structures. A 3D view of the mesh is presented in figure 2c).

An application where \textit{Imesh} can be very useful is terrain modeling, as illustrated in figure 3 illustrates this kind of application. A satellite image of "Lake Superior" (obtained by google earth) is presented in figure 3a). A 3D view of the mesh produced by \textit{Imesh} (considering two submeshes) is shown in figure 3b).

Table 1 shows the computational times involved in building the meshes shown in figures 1, 2, and 3. Notice that the total time spent by \textit{Imesh} is very acceptable, even for big images as the lake (second column of table 1 contains the image sizes).

<table>
<thead>
<tr>
<th></th>
<th>Dimension</th>
<th>Initialization</th>
<th>Partitioning</th>
<th>Mesh Improvement</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piece</td>
<td>256x256</td>
<td>0.30</td>
<td>0.18</td>
<td>0.43</td>
<td>0.91</td>
</tr>
<tr>
<td>Vein</td>
<td>474x794</td>
<td>0.62</td>
<td>0.03</td>
<td>1.98</td>
<td>2.63</td>
</tr>
<tr>
<td>Lake</td>
<td>1144x570</td>
<td>0.98</td>
<td>0.48</td>
<td>0.96</td>
<td>2.42</td>
</tr>
</tbody>
</table>

Table 1: Computational times (in seconds)

We finish this section showing an interesting characteristic of \textit{Imesh} . As discussed in section 2, most of the existing methods aim at generating meshes that minimize interpolation errors, being thus effective to image representation ends. On the other hand, \textit{Imesh} produces triangles fitted within homogeneous regions of the image, what allows a more accurate mesh partitioning (mesh segmentation), a desirable characteristic in applications whose goal is to detect and mesh some structures contained in the image.

Although \textit{Imesh} has not been designed to image representation, its performance in
A.J. Cuadros-Vargas and L.G. Nonato

Figure 1: Imesh process. a) Original image; b) Mesh produced in the Initialization step; c) Mesh partitioned in three submeshes; d) Final quality mesh.

this kind of application is not completely bad, as shown in figure 4. Taking as test case the well known “Lena” picture (figure 4a)), we compare, qualitatively, Imesh with a quadtree method that generates a mesh by minimizing the linear interpolation error [19]. Figures 4b) and 4c) show the mesh obtained by the quadtree method and the image obtained by interpolating the gray levels from the mesh vertices. Figures 4d) and 4e) depict the mesh and the interpolated image generated by Imesh. It is important to say that the meshes in figures 4b) and 4d) have almost the same number of vertices (7520 and 7521 respectively). Notice that the image produce by Imesh (figure 4e)) has a similar quality when compared with the image in figure 4c), which has been generated by the quadtree method that minimizes interpolation error. However, an interesting fact occurs when we do not interpolate the gray levels in the mesh produced by Imesh, as shown in figure 4f). In fact, to generate the image in figure 4f), we just assign to the pixels inside a triangle the average color stored in such a triangle. It is not difficult to see that in the
regions where the image edges are well define, $Imesh$ without interpolation produces a sharper image (Lena’s face and hat).

6 CONCLUSIONS

In this work we have presented a novel image based mesh generation algorithm that automatically distinguishes different structures contained in an image while ensuring good quality of the resulting triangular elements. The algorithm unifies concepts from mesh generation and image processing in an integrated framework for automatic meshing from images.
The strategy adopted in the first step of the algorithm, which inserts points not close to already existing vertices, turns out to be very efficient, as it avoids accumulating points. The partitioning strategy, which generates submeshes based on similarity as well as in area, has also produced good results in the tests we have run. The adaptation of Ruppert’s algorithm to work in a partitioned mesh has given rise to a refinement strategy that improves the quality of all submeshes simultaneously.

We are currently working in the three-dimensional version of *Imesh*, which will allow to build tetrahedral models straightly from volumetric images. New mesh partitioning strategies are also under investigation, mainly regarding textures.

**Acknowledgements**

We acknowledge the financial support of FAPESP - the State of São Paulo Research Funding Agency (Grant# 03/02815-0 and #02/05243-4), and CNPq, the Brazilian National Research Council (Grant # 300531/99-0). The vein image is courtesy of Toshiba Medical Systems (http://www.toshiba-europe.com/medical/Materials/ImageGallery/Aorta.jpg)

**REFERENCES**


Figure 4: Comparison between *Imesh* and a quadtree based method regarding image representation. a) Original image; b) and c) Mesh and interpolated image generated by a quadtree method; c) and d) Mesh and interpolated image generated by *Imesh*; d) Image from *Imesh* without interpolation.