Affinement et adaptation de maillages pour la localisation de l'activité corticale cérébrale

Refined adaptive meshes for the localisation of cortical activity

M. Elena Martínez-Pérez[a],
Line Garnero[b] and Isabelle Bloch[a]

Technical Report

[a] École Nationale Supérieure des Télécommunications (ENST)
[b] Centre MEG-EEG Hôpital Pitié Salpêtrière Paris

Résumé

La localisation de l'activité neuronale sur la surface 3D du cortex à partir de mesures externes de potentiels électriques (EEG) et de champs magnétiques (MEG), associée à une représentation 3D du cortex obtenue par des techniques d'imagerie telles que l'IRM, fournit un outil important d'investigation non invasive de l'activité du cerveau et de sa dynamique, à la fois pour des études cliniques et pour des recherches en sciences cognitives. Cela nécessite de résoudre deux problèmes différents mais non indépendants. Le problème direct consiste à déterminer un modèle suffisamment précis des propriétés électromagnétiques des tissus de la tête pour permettre le calcul des champs externes créés par une source de courant connue dans le cerveau. Le problème inverse consiste à trouver la distribution et l'amplitude des sources qui correspondent au champ mesuré à la surface de la tête. Afin de construire un modèle de la tête pour résoudre le problème direct, des techniques de segmentation doivent être mises en œuvre pour détecter les surfaces ou volumes des différents tissus de la tête dans des images IRM. L'obtention et la discrétisation de ces surfaces constituent la première étape de la construction du modèle. À partir des volumes segmentés et d'une tétraédrisation presque régulière développée à l'ENST en collaboration avec le LENA, le but de ce projet est de développer un algorithme d'amélioration de ces maillages afin de les faire mieux coïncider avec les surfaces extraites de l'IRM. Ensuite, nous cherchons à adapter les maillages localement pour répondre aux besoins de la localisation des activations. Cela concerne d'une part l'affinement du maillage dans les régions du cerveau plus actives selon la tâche cognitive étudiée, et d'autre part la réduction du temps de calcul en gardant un maillage moins précis dans les zones non activées.

Mots clés : maillage tétraédrique, affinement, adaptativité, éléments finis.

MARIE CURIE FELLOWSHIP

This Research was Supported through a European Community Marie Curie Fellowship

http://www.cordis.lu/improving

Disclaimer: The author is solely responsible for information communicated and the European Commission is not responsible for any view or results expressed

Abstract

The localisation of neuronal activity on the 3D cortical surface from external measurements of the electric potentials (EEG) and the magnetic fields (MEG), mapped into a 3D representation of the cerebral cortex by means of imaging techniques can be an important non-invasive tool for clinical studies and cognitive research into the dynamics aspects of brain activity. Carrying out this representation requires the solution of two separate but interrelated problems. The "direct problem" of determining a sufficiently accurate model of electromagnetic properties of the tissues of the head to enable the calculation of the external fields generated by a known current source within the brain. The second problem is the "inverse problem" of finding the distribution and magnitude of current sources that correspond to a measured external field. In order to derive the model of the head for the direct problem, segmentation techniques have to be implemented to detect the anatomical surfaces or volumes of the different tissues of the head from MRI images. The discretisation of these segmented surfaces is the first step for the definition of the model. Based on the previously segmented volumes and in the almost regular tetrahedrisation (ART) obtained at ENST and LENA, the aim of this part of the project is to develop an algorithm that automatically improves the mesh representation of the surfaces and volumes generated based on the segmented volumes. Once the meshes are properly adapted to the real surface obtained from MRI images, the next stage of this project will be to adapt the mesh locally in order to improve the accuracy of the computation and consequently the localisation of the activity. Local adaptation of mesh is particularly important in the present project for two main reasons: 1) refinement of the mesh in particular areas of the brain where more activity can be found depending on the cognitive task to be studied and 2) the problem size and computational cost grow very rapidly as the grid size is reduced, thus the grid should not be small particularly in areas where less brain activity is expected.

Key words: tetrahedral mesh, refinement, coarsening, adaptive, finite element.

Acknowledgments

This work has been carrier out as a collaboration between the department of Signal and Image Processing TII at ENST and the Cognitive Neuroscience and Brain Images group at LENA.

I would like to express my deep gratitude to Professor Isabelle Bloch for her support, supervision and friendship. I would also like to thank to Dr. Line Garnero from LENA for her support and useful suggestions and to Henri Maître, head of TII, to allow me to work in his department at ENST.

Finally, I am very grateful to the European Commission and the Marie Curie Individual Fellowship for the financial support (Contract no. HPMF-CT-2000-00979).

Contents

Chapter 1

Introduction

The localisation of neuronal activity on the 3D cortical surface from external measurements of the electric potentials (EEG) and the magnetic fields (MEG), mapped into a 3D representation of the cerebral cortex by means of imaging techniques can be an important non-invasive tool for clinical studies and cognitive research into the dynamics aspects of brain activity.

Current functional brain imaging techniques such as functional magnetic resonance (fMRI) or positron emission tomography (PET) have a good spatial resolution (a few mm) but insufficient temporal resolution to describe the brain activity accurately. To overcome this problem of resolution, the idea of representing the EEG and MEG information with high temporal resolution (ms) in conjunction with anatomical information that can be extracted from magnetic resonance imaging (MRI) techniques has been introduced [Phillips *et al.*, 1997].

Carrying out this representation requires the solution of two separate but interrelated problems: the "direct problem" of determining a sufficiently accurate model of the electromagnetic properties of the tissues of the head to enable the calculation of the external fields generated by a known current source within the brain and the "inverse problem" of finding the distribution and magnitude of current sources that generated a measured external field. Figure 1.1 shows in an schematic way the source localisation general principle.

In order to solve the direct problem it is necessary to develop a specific model of the head. The EEG direct computation requires specification of the boundaries between the brain cerebro-spinal fluid and skull, the skull and scalp, and the scalp and air, and the relative conductivities of each of the regions. The MEG direct solution, however, requires only the specification of the inner skull boundary, which could be constrained to be within the cortical surface and the inner skull [Liu *et al.*, 1998].

There have been many studies based on idealised geometries (like concentric spheres) with isotropic electromagnetic properties and, more recently, models based on more realistic shapes [Cuffin, 1996]. The solutions of the direct problem can be calculated for

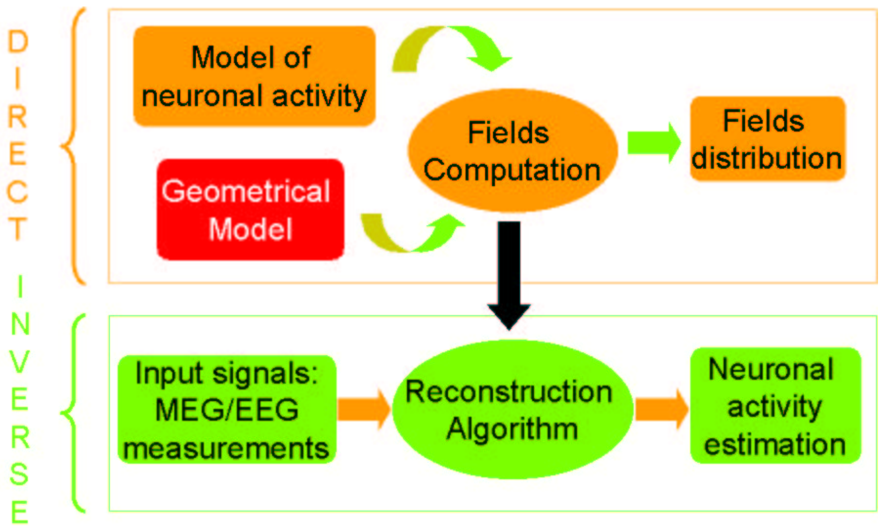


Figure 1.1: Source localisation principle.

isotropic properties using boundary element methods (BEM) [Mosher *et al.*, 1999]. The development of volumetric meshes to model the different tissues of the head with a finite element method to calculate the electromagnetic fields have also been reported [Marin *et al.*, 1998]. Figure 1.2 shows an example of the different geometric models that have been used in the literature for EEG/MEG localisation.

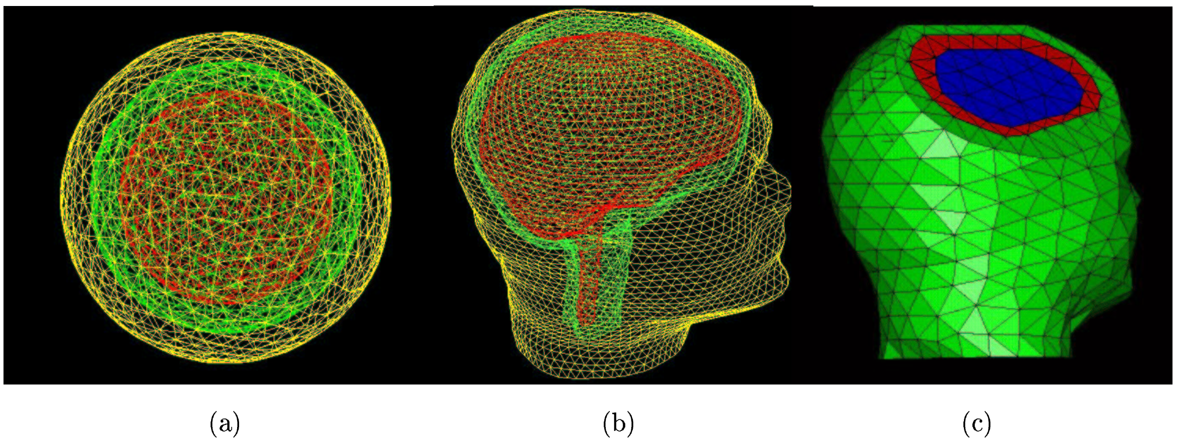


Figure 1.2: Different geometry models. (a) Concentric spherical model with analytical solution, (b) realistic surface models for BEM and (c) realistic volume models for FEM.

In order to derive the model of the head for the direct problem, segmentation techniques have to be implemented to detect the anatomical surfaces or volumes of the different tissues of the head from MRI images. This stage will be based mainly on existing algorithms developed at ENST and LENA [Dokladal *et al.*, 2001]. The discretisation of these segmented surfaces is the first step for the definition of the model. Figure 1.3 shows the steps followed to obtain the meshed models.

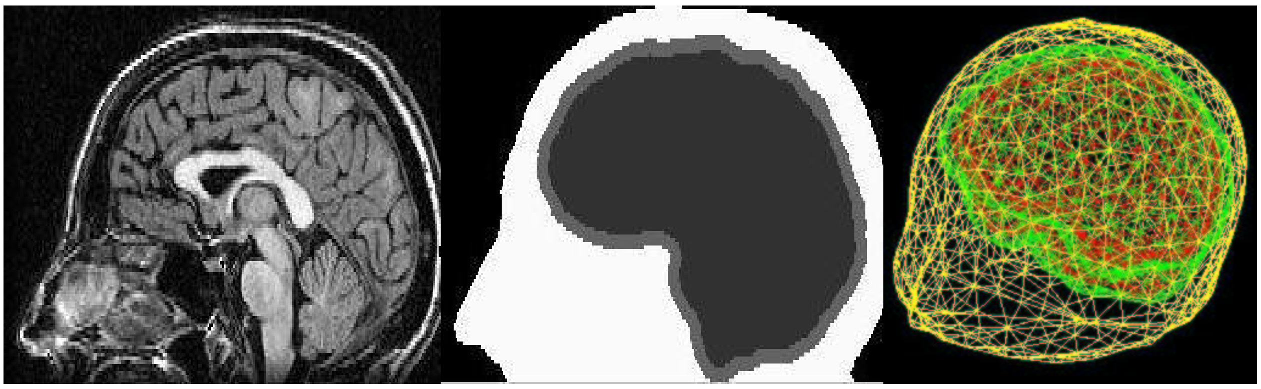


Figure 1.3: Steps to obtain the mesh model. (a) Anatomical information from MRI images, (b) volume segmentation and (c) surface mesh of the segmented volumes.

Based on the work by Pescatore [2001] at ENST on almost regular tetrahedral (ART) meshes (recursive construction of meshes and labelling according to the different tissues) and on information acquired from anatomical MRI, the problem of mesh refinement will be addressed. At the present, the ART surface and tetrahedral meshes obtained from the MRI segmented surfaces do not match precisely with the tissue surfaces as seen in MRI acquisitions, they are not sufficiently smooth. Therefore the first aim on this stage is to refine the current mesh in order to have a better geometrical representation of the tissue interfaces (brain, skull and scalp) extracted from MRI data. These refinements have to be performed under topological constraints in order to preserve the topological arrangement and the conductivity properties of the tissues of the head. This stage is very important for source localisation, therefore a compromise between the distance of the surface meshes to the actual tissue surfaces as well as the shape of the tetrahedral volumes has to be found. Achieving good properties for both surface meshes (for BEM models) and tetrahedral meshes (for FEM models) will allow us to compare boundary element methods with finite element methods for solving the inverse problem.

Once the meshes are properly adjusted to the surfaces obtained from MRI images, the next stage of this project will be to adapt the mesh locally, to a specific area of the tissues, in order to improve the accuracy of the computation and consequently the localisation of the neurological activity. Common applications of mesh generation are, for example, in the area of computational fluid dynamics simulations where meshes are constructed and adapted to a specific model flow constrained to the geometry and the flow properties. In our problem an adaptive tetrahedral mesh can be built constrained to the geometry of the head and the properties of tissue conductivity included in the direct model preserving the topology of specific areas of brain activity. This local adaptation of the mesh is important for two main reasons: 1) improved resolution by reducing grid size in regions of increased neurological activity and 2) reduced computational costs by increasing grid size in areas of less brain activity. Figure 1.4 shows the mesh adaptation idea in the feedback process.

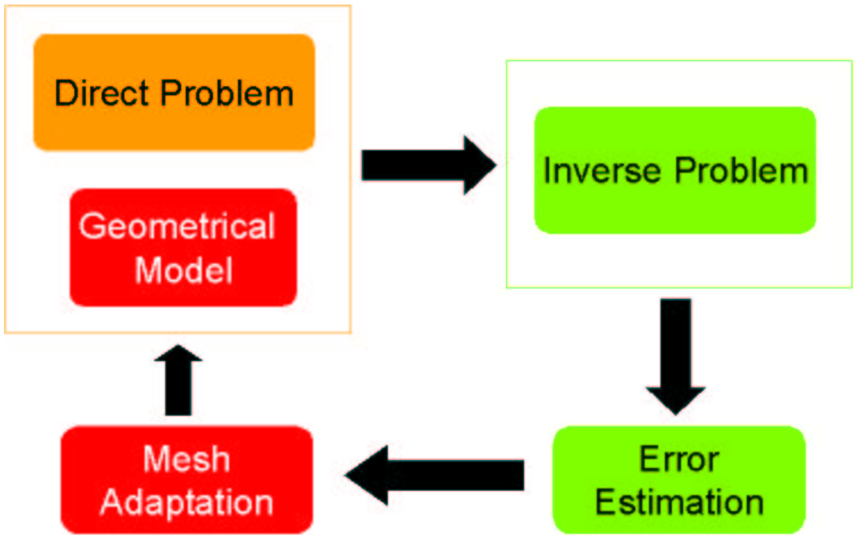


Figure 1.4: Mesh adaptation and feedback process.

Methods of refinement/derefinement of meshes such as bisection will be investigated [Rivara and Levin, 1992; Plaza *et al.*, 2000; Molinari and Ortiz, 2002]. In addition to iterative adaptation of the mesh based upon the solution of the inverse problem itself, information obtained from other imaging techniques such as fMRI will be used. This local adaptive refinement will allow us to focus on the most interesting areas, where a higher resolution is required.

Layout

In Chapter 2 we address the description of the background work on MRI segmentation and ART generation made at ENST from which this work is based. Chapter 3 focuses in a description of different mesh evaluation criteria to assess the quality of the current surface and volume ART meshes; results of these evaluations are also presented. In Chapter 4 a review of the state of the art on 3D refinement is made and a table of commercial and free share software available at the present for 3D mesh generation and refinement is shown. Chapter 5 discusses two main approaches of mesh refinement that we follow in order to achieve our goal of mesh refinement for the localisation problem. In Chapter 6 the current progress accomplished in mesh refinement is shown, and a comparison between the two techniques presented in Chapter 5 is done. Finally, Chapter 7 presents the conclusions and the recommendations for future work.

Chapter 2

Background work made at ENST

2.1 MRI volume segmentation

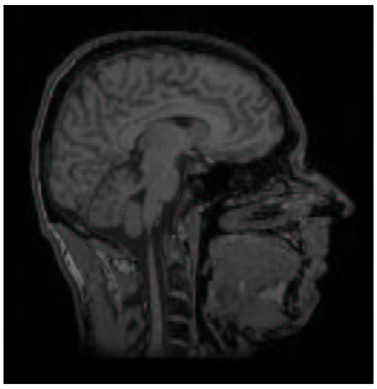
The MRI segmentation procedure that we use to obtain the segmented volume tissues of interest for the mesh construction were developed by Dokladal *et al.* [2001] (Appendix A describes the technical steps to run the programs). The method extracts several anatomical tissues from the head using MRI images.

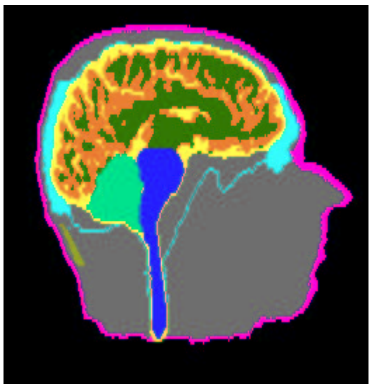
The algorithm is based on a preliminary segmentation of the brain and uses a morphological method to segment the brain stem and the cerebellum, cerebrospinal fluid (CDF), grey and white matter, skull and skin. The method has an adequate and robust representation of the shapes and it uses a very strong constraint to preserve the topology. Robustness is achieved in this morphological approach by an intensive use of reconstruction and conditional operations as well as by reducing the number of parameters, while the topology is preserved using homotopic transformations.

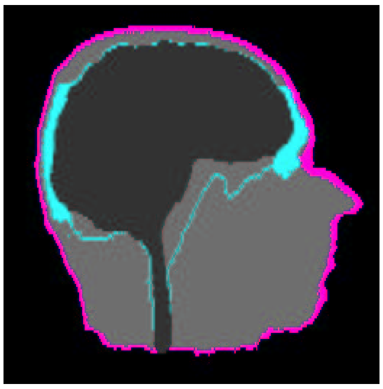
The algorithm begins by an automatic selection of markers that permits to control the number of resulting objects. The selection of markers relies on a directed component tree representation of a certain function (like distance function). To extract a known number or markers it maximises a given criterion which can be based on some components characteristics like height, area or volume as an attributes.

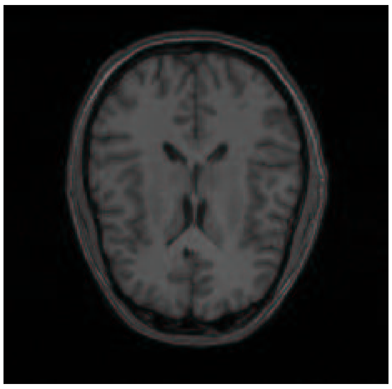
An homotopic deformation that combines topological and other criteria like distance or grey levels is used. The homotopic transformations can be used in discrete spaces thanks to the used of simples points, *i.e.* points that can be deleted or added without modifying the topology. Beginning with the brain stem and the cerebellum, the algorithm successively segments the structures of interest, in the order mentioned above, in a cascade fashion.

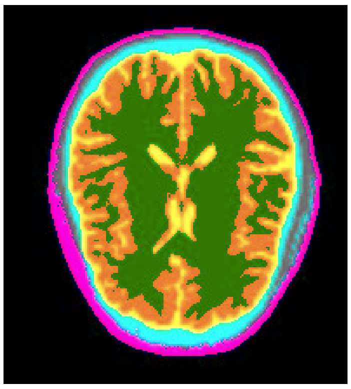
Figure 2.1 shows the result of applying this algorithm. Figures 2.1(a) and (d) show an axial and a sagittal view of an MRI slide, respectively. Figures 2.1 (b) and (e) show the corresponding segmented tissues: brain stem and the cerebellum, cerebrospinal fluid (CDF), grey and white matter, skull and skin. Finally, Figures 2.1(c) and (f) show only those tissues that are of our interest at the moment for this project: brain, skull and skin.


(a)


(b)


(c)


(d)


(e)

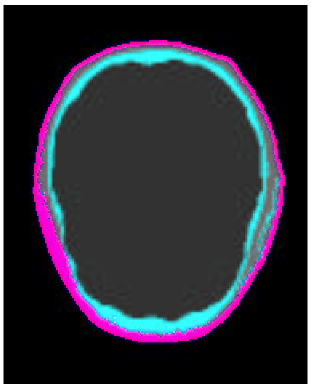

(f)

Figure 2.1: MRI segmentation example. (a) and (d) MRI axial and sagittal views respectively. (b) and (e) segmented images with tissues marked on colours: brain stem, cerebellum, cerebrospinal fluid, grey and white matter, skull and skin. (c) and (f) segmented images showing only the tissues of interest: brain, skull and skin.

We segmented 7 different MRI volumes using this algorithm (Appendix A gives the directories where these segmented images can be found). We find that the selection of the mean and standard deviation of the classes sometimes fails, therefore a manual tuning is needed. Also, knowledge of anatomy in needed since for the segmentation of brain stem and cerebellum the user needs to select the proper MRI slide near those structures before beginning the segmentation. Since the segmentation is in cascade, if brain stem and cerebellum are not segmented properly the rest of the tissues are segmented erroneously.

2.2 The ART volume and surface meshes

The initial mesh from where we will develop the mesh refinement is based on the work made at ENST by Pescatore [2001]. Ideally, a finite element (FE) tetrahedral mesh should consist of elements close in shape to a equilateral tetrahedron and having nearly equal edges. Moreover, the number of elements should be compatible with the computation cost of the forward problem of EEG and MEG. Thus, the 3D mesh generation proposed by Pescatore [2001] relies on a recursive decomposition of an initial segmented volume into congruent tetrahedra called almost regular tessellation of $\Re^3$ (ART), in which the resolution, and therefore the number of elements, can be controlled.

The method is based on the following idea: if a tetrahedron K is split into 8 tetrahedra by bisecting the edges as shown in Figure 2.2, the small tetrahedra should be congruent to K scaled by a factor of 1/2. The subdivision in this scheme is proved to be subdivision invariant [Fuchs, 1998]. Pescatore [2001] decided to chose this subdivision scheme since, from all the possible invariant subdivisions, this is the one which has the best shape quality, which is measured as:

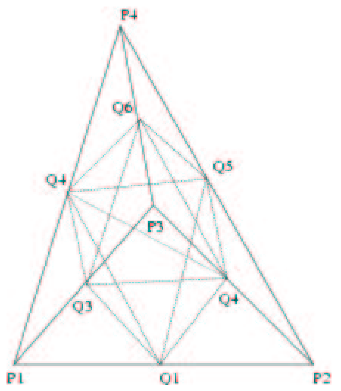


Figure 2.2: Subdivision of tetrahedron K.

$$Q_K = \alpha \frac{h_{max}}{\rho_K}$$

whereh_{max} is the longest side of the tetrahedron K, ρ_K is the inradius of the tetrahedron K and $\alpha = \sqrt{6}/12$ for regular tetrahedron.

The mesh generation can be divided into two parts and summarised as follows: In the first part, an initial mesh T_K is built as a polyhedron of 24 subdivision invariant tetrahedra which share the same vertex, which is called the "geometric constructor". Each of the tetrahedron K is subdivided into 8 tetrahedra following the scheme described above depending on the resolution desired. Once the geometric constructor of an ART is achieved, in the second part of the method a set M_T is constructed which represents the set of proportion of each tissue in each tetrahedron K of a mesh T. Finally, the tetrahedra are labelled as belonging to a particular tissue following some topology constraints. (Appendix B describes the technical steps to run the programs and the mesh file formats used).

```
Algorithm 1
**Geometric constructor of an ART initialisation:** Let $T$ be the polyhedron formed of 24 subdivision invariant tetrahedra $K$ with:

  • T.Vertex the array of all vertices of $T$
  • T.Facet the array of all tetrahedra of $T$

Let $N_T$ be the total number of tetrahedra of $T$
For resolution $= 1$ to $n$ ($n =$ resolution scale)
for tetrahedron $i = 1$ of $T$ to $N_T$ ($N_T =$ number of tetrahedra)

  • Let Child.Vertex be the array of the 10 vertices of the decomposition of the tetrahedron $i$
  • Let Child.Facet be the array of the 8 tetrahedra subdivision invariant created from the tetrahedron $i$

  1. subdivide $i$ in 8 subdivision invariant tetrahedra
  2. insert Child.Vertex in T.Vertex
  3. insert Child.Facet in T.Facet

 endfor $i$
endfor resolution
```

Where N_T of such an ART follows the low: $N_T = 8^n \times 24$. Figure 2.3 shows the ART initialisation for $n = 0$ ($N_T = 24$), $n = 1$ ($N_T = 192$) and $n = 3$ ($N_T = 1536$), respectively.

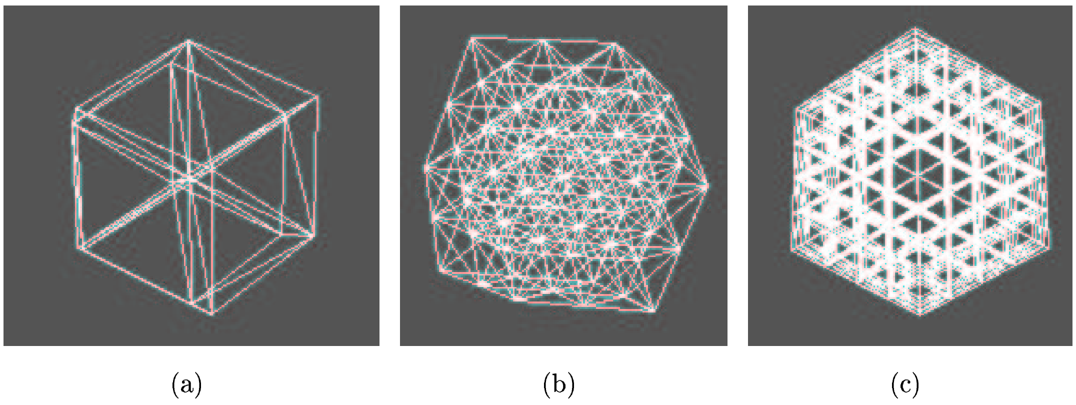


Figure 2.3: Geometrical ART constructor. (a) $n = 0$ ($N_T = 24$), (b) $n = 1$ ($N_T = 192$) and (c) $n = 3$ ($N_T = 1536$).

The next step, after building the ART initialisation, is the computation of the tissue composition (M_T) for each tetrahedron K of the whole volume mesh T. Algorithm 2 shows the construction of the set M_T of T from the segmented image composed by V different objects (in this case tissues).

```
Algorithm 2
**Construction of the set M:** Let $I(o_1, \dots, o_V)$ be the segmented
based image with $V$ objects.
Let $N_T$ be the total number of tetrahedra of $T$.
Let $o_l$ be the $l^{th}$ segmented object of $o_1, \dots, o_V$.
Let $M(K)$ be the vector of size $V$ representing the proportion of
each object $o_l$ in the tetrahedron $K$.
**Initialisation:**
For all $K$ , $M(T) = (o_1 = 0, \dots, o_V = 0)$
**for** $K = 1$ **to** $N_T$
Find the tissue composition of each tetrahedra $K$ in $M_T(K)(o_l)$
**endfor** $K$
```

After computing the tissue composition a labelling procedure is done under some topological constraints by applying homotopic deformations on the different tissues. Homotopic deformations are deformations that preserve topological properties. This notion is usually defined as the preservation of the number of connected components and of the number of tunnels. This leads to the notion of simple points which are the elements (voxels, polyhedra) that can be added/removed from an object without changing its topology. Pescatore [2001] proposed a local characterisation of simple tetrahedra. The labelling of tetrahedra consists in, starting with a connected mesh model initialised on the inner brain volume, a simple tetrahedra is added by taking into account the adjacency properties of the object such that the brain labelled tetrahedra should not be connected to a skull tetrahedra.

To label the skull, the process is repeated considering the union of brain and skull as a single connected object. Finally, in an analog manner, to label the skin the process considers the union of brain, skull and skin tissues as a single connected object. Figure 2.4 shows and example of the whole ART construction process.

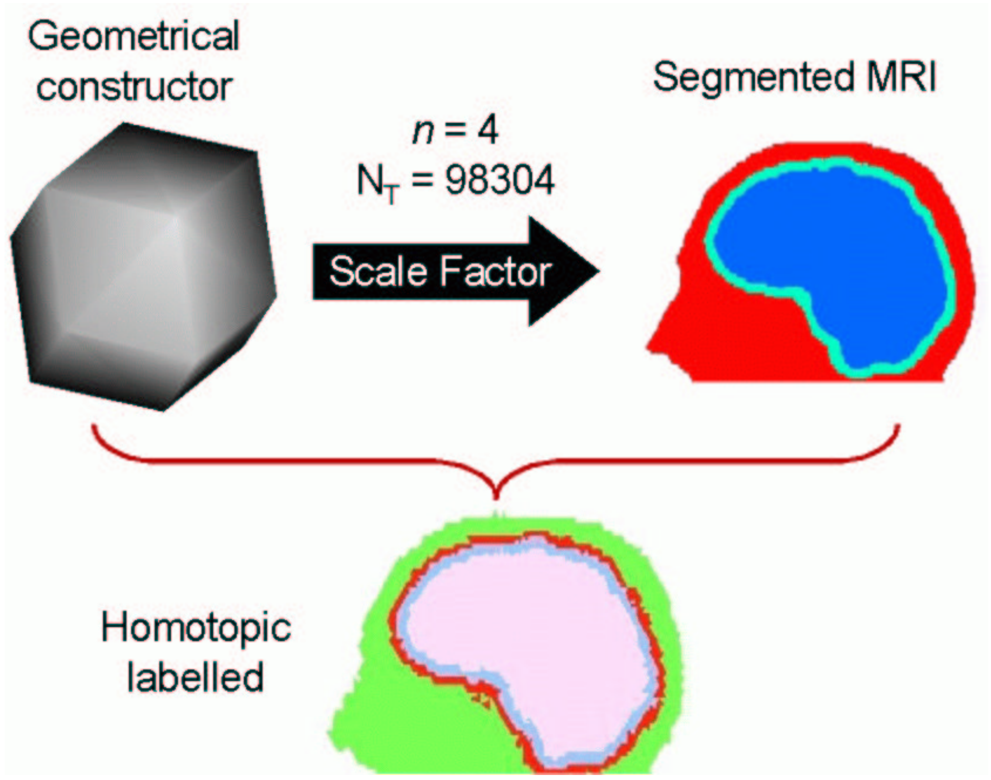


Figure 2.4: Construction process of an ART mesh.

Chapter 3

Current surface and volumetric mesh evaluation

After studied the work made at ENST by Pescatore [2001], we decided first to perform an evaluation of the current surface and volume ART meshes. Because for finite element methods the quality of the geometric approximation is of significant importance due to its effect on the accuracy of the numerical solutions and the convergence of the computational scheme, we consider the need to have proper tools to evaluate ART and future modified meshes. The aim of the present chapter is to show the criteria followed for evaluation and the results found in the evaluation of ART meshes for each of the tissues of interest.

3.1 Surface mesh evaluation criteria

A surface mesh must be an optimal piecewise planar approximation of the original surface such that the maximal distance between the original and the approximating surface does not exceed a given tolerance and moreover, the shape of the elements must be as optimal as possible. The criteria use to evaluate the surface meshes of each of the tissues are based on two main approaches: the first is related to the distance that each vertex has with respect to the original surface, and the second is based on geometric feature analysis of the mesh itself such as: quality of elements, planarity, smoothness and deviation [Frey and Borouchaki, 1998].

3.1.1 Distance map

A *distance map* converts a binary image to a grey scale image in which each pixel value gives the straight-line distance from each object pixel to the nearest background pixel.

Chamfer distance transformations are an approximation of Euclidean distance and rely on the assumption that it is possible to deduce the value of the distance at a pixel from the value of the distance at its neighbors for regular metrics, i.e. metrics for which for all p, q such that: $dist_M(p,q) \leq 2$, there exists an r different from p and q such that $dist_M(p,q) = dist_M(p,r) + dist_M(r,q)$.

Borgefors [1984] reviewed a number of metrics in 2 and 3 dimensions. Chamfer distance transformations are produced in two raster scans over the image, using the masks of Figure 3.1. In the forward scan, the mask starts in the upper left corner of the picture, moves from left to right and from top to bottom. In the backward scan, it starts in the lower right corner, moves from right to left and from bottom to top. The local distances, $d1$ and $d2$, in the mask pixels are added to the pixel values in the distance map and the new value of the zero pixel is the minimum of the five sums.

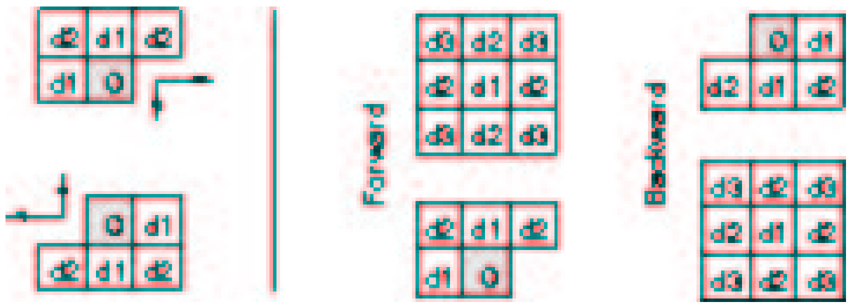


Figure 3.1: Chamfer masks. Left: 3×3 mask in 2D, right: $3 \times 3 \times 3$ mask in 3D.

Borgefors [1986] computes the optimal values for $d1$ and $d2$ in order to minimize the maximal difference between the chamfer metric and the Euclidean one. She extends the chamfer masks to 5×5 and 7×7. She computes the optimal values for the local distances and evaluates several integer approximations for 3×3 to 7×7 masks. She recommends using (3:4) and (5:7:11) approximations for 3×3 and 5×5 masks, but finds no significant interest in using 7×7 masks. The chamfer (5:7:11) mask is illustrated in Figure 3.2.

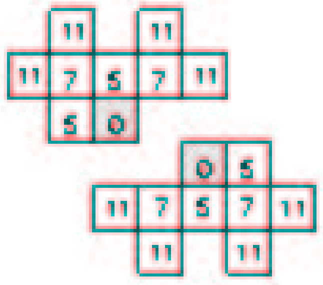


Figure 3.2: Chamfer (5:7:11) mask in 2D.

With the use of the library TIVOLI, developed at ENST (http://www.enst.fr/externe/tsi.html), we compute the chamfer distance of the MRI volumes with mask of $3 \times 3 \times 3$, for each of the tissues of interest. Then we match the vertices of the surface mesh to the pixels of the distance map of the volume contours to find out how far are the vertices from the real surfaces. Figure 3.3(a) shows a 2D slide of the binary image of the brain tissue, (b) the contour and (c) the distance map.

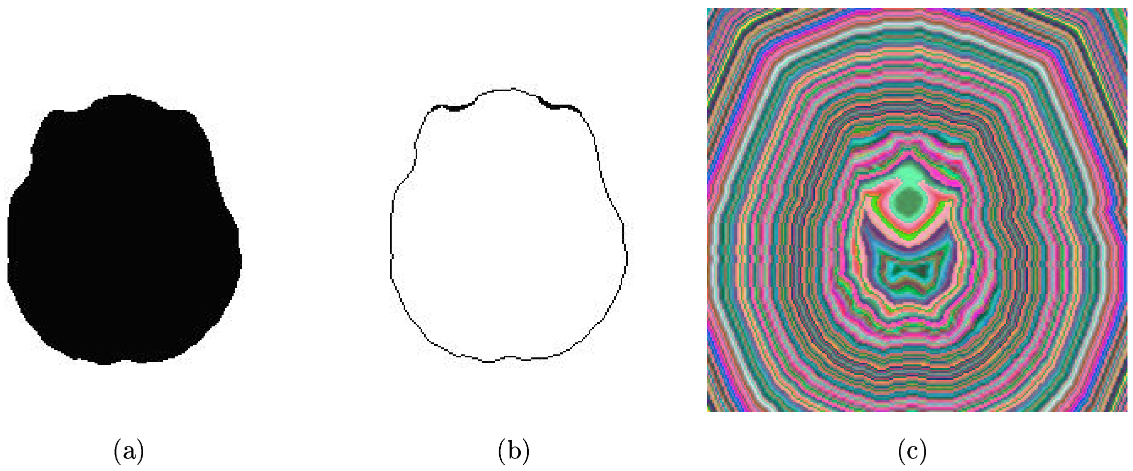


Figure 3.3: Brain tissue. (a) 2D slide of the binary brain volume, (b) the contour and (c) the distance map.

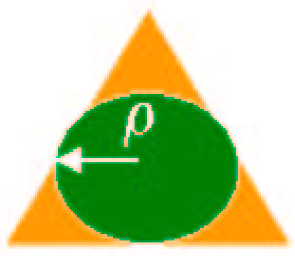
After obtaining the distance values for each mesh tissue, statistics like mean, standard deviation, median and maximum values were calculated.

3.1.2 Geometric features.

The criteria used for evaluating finite element meshes, proposed by Frey and Borouchaki [1998], are exclusively based on geometric feature analysis and base on the sole discretisation (no parametric or implicit surface is required). For this reason, it is needed that all the vertices of the mesh fall as near as possible to the real surface.

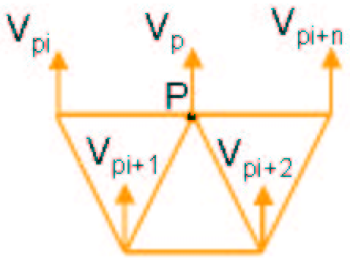
Triangle shape quality: The shape quality Q_k of a triangle K is given by:

$$Q_k = \alpha \rho_k / h_k$$



where h_k is the element diameter (the length of the longest edge), ρ_k is the in-radius and $\alpha = 6/\sqrt{3}$ is a normalisation coefficient so that $Q_k = 1$ for an equilateral triangle. Therefore, the shape quality ranges from 0 (for a flat element) to 1 (for an equilateral triangle) and it is suitable to discriminate well-shaped from degenerated elements.

Mesh planarity: The geometric discontinuities of a surface triangulation usually represent an abrupt change between the normal directions at the surface from a vertex to its neighbours, as well as between adjacent faces. A large change of the normal between two adjacent vertices, where the surface is supposed to be continuous, indicates that the element density is not able to capture the local variations of the surface. The planarity Pl_p at vertex P is defined as the largest angle between the normal v_P and the normals v_{P_i} at the vertices P_i.



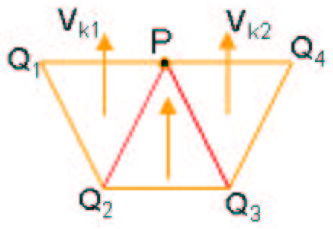
$$Pl_p = \frac{1}{2}\left(1 + \min_{P_i} \langle v_P, v_{P_i} \rangle\right)$$

Mesh smoothness: The smoothness is a measure of the roughness of the surface at a vertex P, the so-called degree of smoothness, and can be introduced as: S_P at P is the minimum value of the edge planarity P_{PQ} of all edges PQ sharing P.

$$P_{PQ_2} = \frac{1}{2}\left(1 + \langle V_{k_1}, V_{k_2} \rangle\right)$$

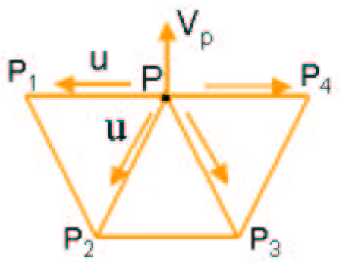
$$S_p = \min_p (P_{PQ_2}, P_{PQ_3})$$

$$S_P = \min_{P_i} P_{PQ}$$



where the edge planarity measures the dihedral angle between triangles K_1 and K_2, which characterises the geometric continuity of the surface along the edge.

Mesh deviation: A variant of the planarity consists in defining the so called deviation of the mesh edges as compared to the original surface. The deviation D_P is defined as the maximum angle between edge PP_i and the tangent to the plane Π_P:



$$D_P = 1 - \min_{P_i} \left\langle v_P, \frac{\vec{PP_i}}{\left\|\vec{PP_i}\right\|} \right\rangle$$

$$D_p = 1 - \min_{P_i} \left| \langle V_p, \vec{u}_{pP_i} \rangle \right|$$

All of the previous geometrical features are normalised and range from 0 to 1, being 1 the optimum feature value.

3.2 Volume mesh evaluation criteria

With the same philosophy as with surface meshes, volume meshes, and in this case tetrahedral meshes, used in FEM applications need to have a good geometrical quality. To evaluate the geometrical quality a concept of *shape tetrahedral measure* has to be defined. Figure 3.4 shows some configurations of *poorly* shape tetrahedra, such as a *silver*, *cap* or *needle* among others.

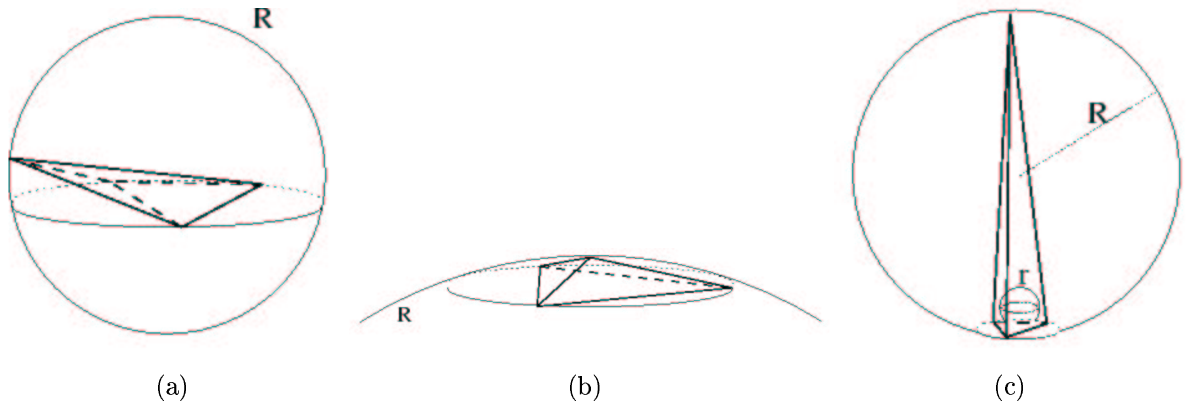


Figure 3.4: Bad shaped tetrahedra. (a) Silver, (b) cap and (c) needle.

In order to measure and characterise such differences in shape, researches in the past have proposed various measures of shape and geometrical quality [Parthasarathy, 1993; Dompierre *et al.*, 1998]. Liu and Joe [1994b] defined a "*tetrahedron shape measure* as a continuous function that evaluates the quality of a tetrahedron. It must be invariant under translation, rotation, reflection and uniform scaling of the tetrahedron. It must be maximum for the regular tetrahedron and it must be minimum other than for a degenerate tetrahedron. There is no local maximum other than the global maximum for a regular tetrahedron and there is no local minimum for a degenerate tetrahedron. For the ease of comparison, it should be scaled to the interval $[0, 1]$, being 1 for a regular tetrahedron and 0 for a degenerate tetrahedron."

A degenerate tetrahedron is a tetrahedron where the volume vanishes and some of the edges do not vanish, like the silver tetrahedron. We will described some of the must common measurements used for tetrahedron evaluation.

Volume : Let $T(t_0, t_1, t_2, t_3)$ be a non-degenerate tetrahedron and $t_i(x_i, y_i, z_i)$ the coordinates of vertex t_i. The volume of a tetrahedron is defined as:

$$V = \frac{1}{6} \begin{vmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{vmatrix}$$

Area : The area of a tetrahedron T is the sum of the surface areas of the triangular facets s_i.

$$S = \sum_{i=0}^{3} s_i$$

where s_i = area of the triangle (t_i, t_j, t_k).

Radius ratio: The radius ratio ρ of a tetrahedron T is defined to be $\rho = N\rho_{in}/\rho_{out}$ where ρ_{in} and ρ_{out} are the inradius and circumradius of T, respectively, and N is the dimension of the space.

$$\rho_{in} = 3V/\sum_{i=0}^{3} s_i$$

$$\rho_{out} = \frac{\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}}{24V}$$

$$\rho = 3\frac{\rho_{in}}{\rho_{out}} = \frac{216V^2}{\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}\sum_{i=0}^{3} s_i}$$

where a, b and c are the products of the lengths of opposite edges of T.

Edge ratio: The edge ratio r of a tetrahedron T is defined to be the ratio between the smallest edge over the largest edge of the tetrahedron.

$$r = \min_{0 \le i < j \le 3} l_{ij} / \max_{0 \le i < j \le 3} l_{ij}$$

where $l_{ij} = \|t_j - t_i\|$. This measure does not agree with the definition of shape measure since it fails detecting some degenerated tetrahedra as the silver.

Aspect ratio: The aspect ratio of two characteristic sizes can be normalised by other measure, but it cannot be a function of the circumsphere, of the smallest edge and of the minimum dihedral angle because degenerate tetrahedra may have non-null circumsphere, smallest edge or dihedral angle. There are many other aspect ratio measures that can be used but the most frequently used and introduced by the GAMMA project [Frey and George, 2000] is defined by the ratio between the inradius and the largest edge:

$$Q = \frac{12}{\sqrt{6}} \frac{\rho_{in}}{\max\limits_{0 \leq i < j \leq 3} l_{ij}}$$

Solid angle: A feasible shape measure to optimise a tetrahedron is based on tetrahedron angles. Tetrahedron shape measures based on the minimum of the solid angle θ_{min} and the minimum of the dihedral angle φ_{min} can be used. They are more complex to evaluate and more costly because of inverse trigonometric functions. The solid angle θ_i at a vertex t_i of the tetrahedron $T(t_0, t_1, t_2, t_3)$ is defined to be the surface area formed by projecting each point of the face not containing t_i to the unit sphere centered at t_i. The area of a unit sphere is 4π, the maximum solid angle for a positive tetrahedron is 2π in the case of a flat tetrahedron where a vertex sees half of the space. A large solid angle near 2π for T implies that T has a small solid angles. Solid angles are computed as:

$$\theta_{min} = \alpha \min_{0 \leq i \leq 3} \theta_i$$

$$\sin\left(\theta_i/2\right) = 12V \left(\prod_{j,k \neq i \, 0 \leq j < k \leq 3} \left((l_{ij} + l_{ik})^2 - l_{jk}^2 \right) \right)^{-1/2}$$

where $\alpha^{-1} = 6 \arcsin(\sqrt{3}/3) - \pi = 0.5512856$ is the value of the four solid angles for a regular tetrahedron ($\approx 30°$).

Dihedral angle: Each of the six edges of a tetrahedron is surrounded by two triangular faces. At a given edge, the dihedral angle between the two faces is the angle between the intersection of these faces and a plane perpendicular to the edge. For a positive tetrahedron, the dihedral angle is bounded by zero and π. It is equal to π minus the angle between the normals of the faces. The minimum dihedral angle is a tetrahedron a shape measure.

$$\varphi_{min} = \alpha \min_{0 \leq i < j \leq 3} \varphi_{ij} = \alpha \min_{0 \leq i < j \leq 3} (\pi - \arccos(n_{ij1} \cdot n_{ij2}))$$

where n_{ij1} and n_{ij2} are the two triangular faces adjacent to the edge ij and $\alpha^{-1} = \pi - arccos(-1/3) = 1.230959$ is the value of the six dihedral angles of the regular tetrahedron ($\approx 70°$).

Any shape measure that satisfy the definition of Liu and Joe [1994b] can be used by a mesh optimiser. The more a mesh is optimised with a given tetrahedron shape measure, the closer to the optimal mesh it is for any other tetrahedron shape measure. At the limit, if it were possible to mesh a domain with only equilateral tetrahedra, as it is in 2D, all mesh optimiser should converge to that mesh, whichever shape measure is used. The problem is that the optimal mesh does not exist in 3D. It is impossible to fit the space with regular tetrahedra, especially when we have complex structures like the ones we are working with. So the converge state is unknown and depends slightly of the tetrahedron shape measure used. For our ART evaluations we used the volume, aspect ratio and dihedral angles.

3.3 Evaluation of ART meshes

3.3.1 ART surface meshes

Since it is noticeable that ART surface mesh vertices are not on the real volume surface and the geometrical features presented in subsection 3.1.2 are based on meshes which vertices are almost on the surface, we decided to generate a brain surface mesh reference using the marching cube algorithm [Lorensen and Cline, 1987], from where we are sure vertices lie on the real surface within a very small discretisation error. Figure 3.5 left: shows a brain surface mesh generated with the marching cube algorithm, and right: the same mesh where only vertices and edges are shown.

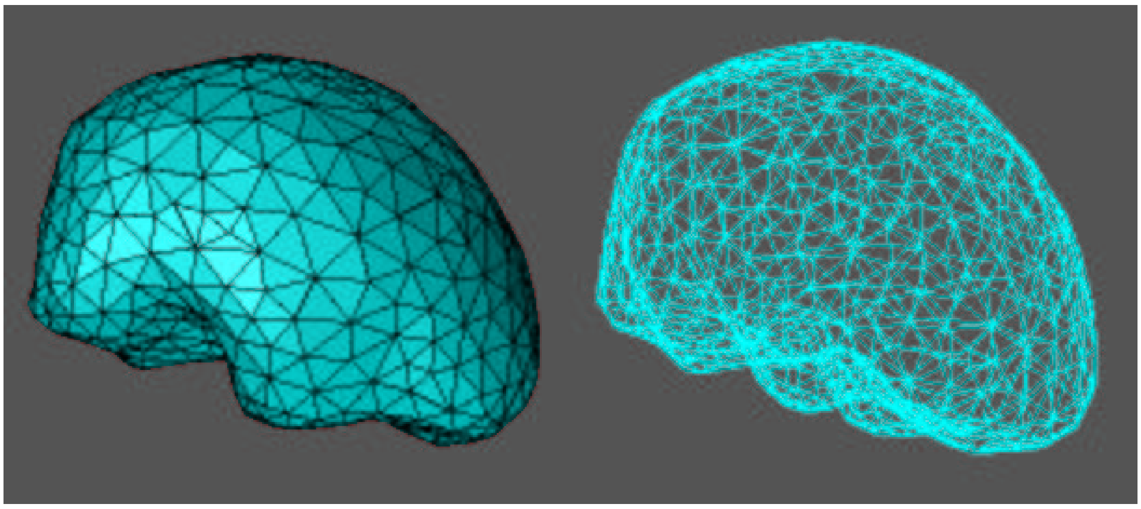


Figure 3.5: Surface of the brain volume. Left: Marching cube surface mesh, right: same where only vertices and edges are shown.

Figure 3.6 shows an example of ART surface meshes for $n = 4$. From left to right we have brain, skull and skin mesh tissues. Table 3.1 shows the values of the different parameter criteria described in sections 3.1.1 and 3.1.2. The first row of the table shows the reference mesh values. The description of how to execute the programs to obtain these evaluations are shown in Appendix C.

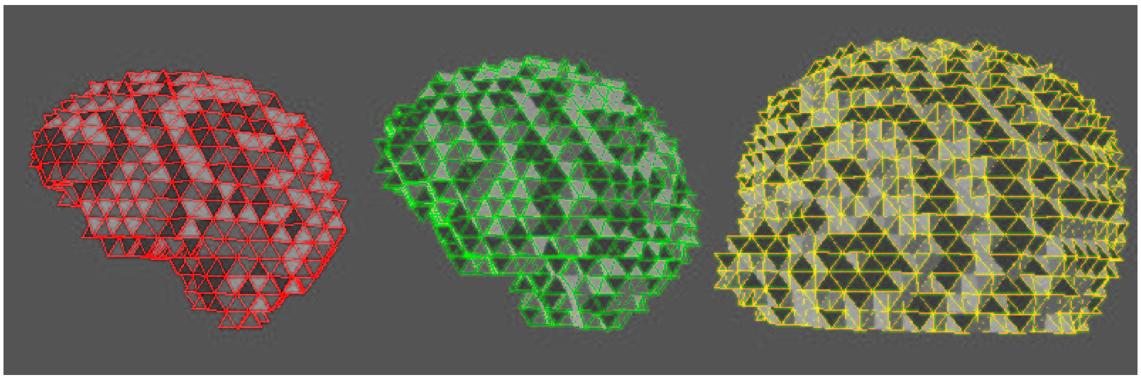


Figure 3.6: ART surface meshes. Left: brain surface mesh, middle: skull surface mesh, and right: skin surface mesh.

Mesh	**Distance** Mean (STD)	**Planarity** min (mean)	**Deviation** min (mean)	**Smooth.** min (mean)	**Quality** min (mean)	**# Elem.**
Refe.	1.27 (0.47)	0.88 (0.97)	0.79 (0.97)	0.89 (0.98)	0.24 (0.79)	9580
Brain	4.79 (3.75)	0.53 (0.62)	0.59 (0.86)	0.25 (0.36)	0.89 (0.89)	1704
Skull	14.44 (12.31)	0.45 (0.49)	0.59 (0.96)	0.25 (0.38)	0.89 (0.89)	29608
Skin	22.20 (19.54)	0.44 (0.52)	0.59 (0.94)	0.25 (0.35)	0.89 (0.89)	39452

Table 3.1:

Surface evaluation parameters. The first column shows the mesh type. The first row contains the reference mesh evaluation values. The rest of the rows contain brain, skull and skin mesh values, respectively. Distance is given in pixels, # Elem. refers to number of triangles per mesh. The other parameters are dimensionless.

Table 3.1 shows a mean distance value for the reference brain mesh of 1.27 ± 0.47 in comparison with the ART brain mesh with 4.79 ± 3.75 pixels which shows a larger mean value and specially a larger standard deviation than the reference mesh. The other distance values cannot be compared with the reference since they are not from the same tissue. Mean values of the rest of the geometrical features, planarity, deviation and smoothness are, for the reference mesh as expected, near to 1 since marching cube algorithm generates a good approximation of the surface. On the contrary, the shape quality mean value, for the same mesh, is lower (0.79) compared with those of all the ART meshes (0.89) which have a high quality since they are almost regular triangles.

There are two important aspects to notice: 1) although the distance values of skull and skin meshes can not be compared with the reference mesh mean value and standard deviation the former increase considerably compared with the ART brain mesh distance, and 2) the number of elements show in the last column of Table 3.1 increase as well dramatically from 1704 triangles for brain mesh to 39,452 triangles for skin mesh.

We cannot make any comment regarding the features of planarity, deviation and smoothness for ART meshes since these measures are defined only for meshes having vertices which lie, with a very small error, on the real surface.

Figure 3.7 shows the ART surfaces where only vertices and edges are shown. It is clear from this Figure, that only brain surface mesh is correctly extracted. Skull and skin surface meshes contain many vertices and edges inside which means that they do not constitute a representation of the real surface of the tissues. Therefore, we conclude that, ART surface meshes are not correctly extracted.

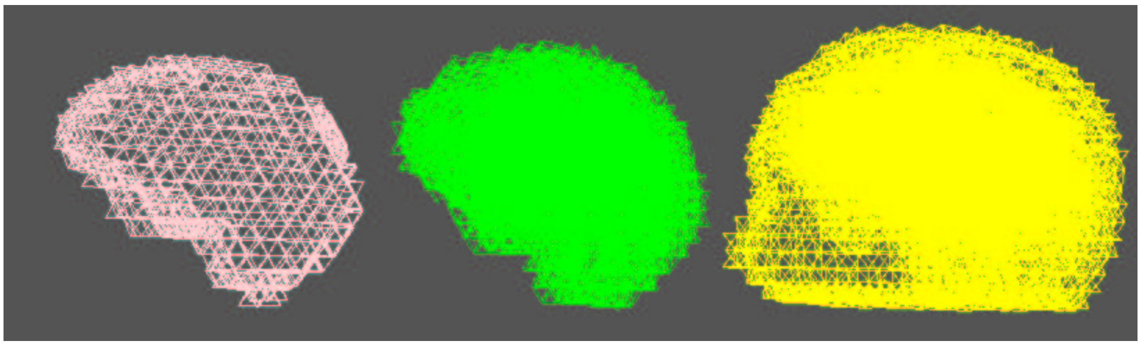


Figure 3.7: ART surface meshes with only vertices and edges shown. Left: brain surface mesh, middle: skull surface mesh and right: skin surface mesh.

3.3.2 ART volume meshes

In order to evaluate the ART volume meshes we calculated parameters such as volume, aspect ratio and dihedral angles, as described in section 3.2. Figure 3.8 shows an example of the ART volume meshes, form left to right: brain, skull, skin meshes and the geometrical constructor for ($n = 4$).

Since ART volume meshes are almost regular and generated from the same geometrical constructor all values of volume and range of shape quality and dihedral angles were the same. Volume values were $V = 97.42$, whereas ranges of aspect ratio were $1.17 \leq Q \leq 1.47$ and dihedral angle were $47.25° \leq \varphi \leq 106.32°$. Figure 3.9(a) shows

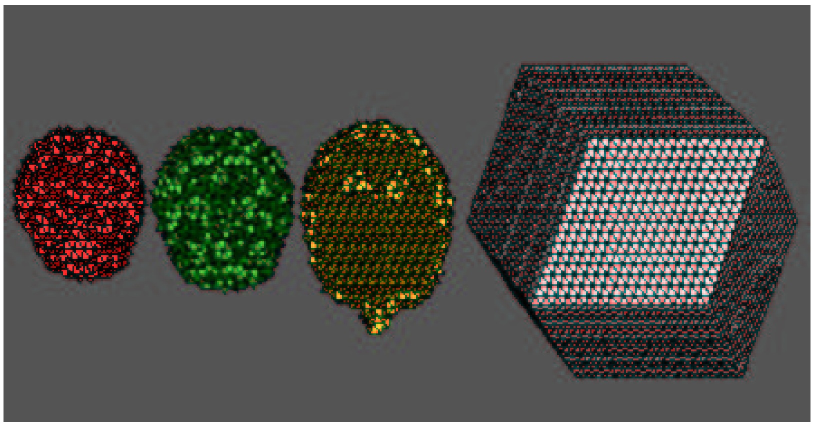
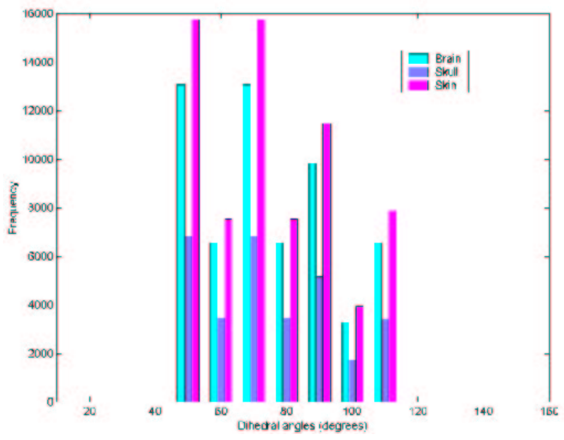


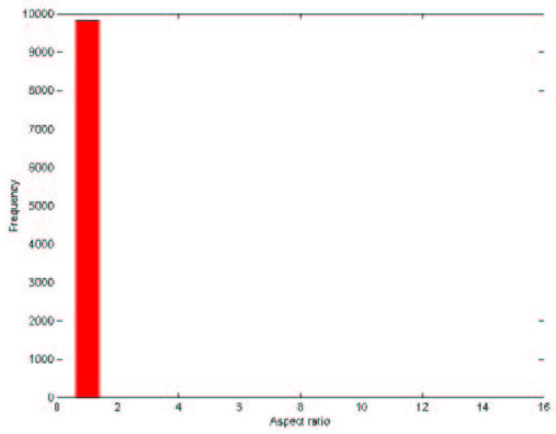
Figure 3.8: ART volume meshes. From left to right: brain, skull, skin meshes and the geometrical constructor ($n = 4$).

the distribution of dihedral angles and (b) the distribution of aspect ratio for all tissues (Appendix C shows the description of how to obtain these evaluations).

Distributions of Figure 3.9(a) have the same shape (and same range), they only vary in number of elements. All distributions have two important modes which are about $50°$ and $70°$ (being $\approx 70°$ for regular tetrahedron). All distributions of Q have a very small standard deviation and fall into the same bar of values as shown in Figure 3.9(b) (being $Q = 1$ for regular tetrahedron). These results are as expected, since we know ART meshes are almost regular.



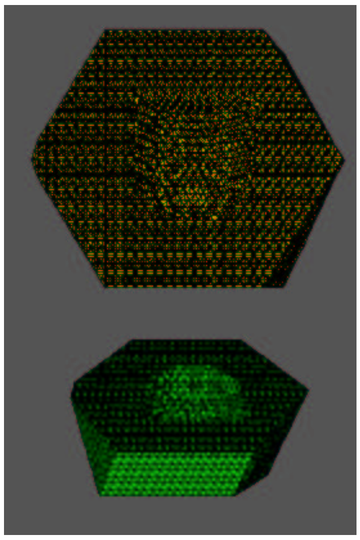
(a)



(b)

Figure 3.9: ART volume evaluation. (a) Dihedral angles and (b) aspect-ratio distributions.

One problem found was in terms of the number of tetrahedra of all tissues together and the number of tetrahedra of the geometrical constructor. Pescatore [2001] concluded that the optimum number of tetrahedra for an ART mesh was given for $n = 4$ (98,304 tetrahedra) since the number of tetrahedra in the mesh is related with the resolution and with numerical computation (in the case of this particular constructor), therefore increasing the number of tetrahedra and resolution to $n = 5$ was computational prohibitive. As can be seen from distributions in Figure 3.9(a) the number of tetrahedra per tissue varies being 9,835, 5,143 and 11,647 tetrahedra for brain, skull and skin meshes, respectively. These values make a total number of 26,625 tetrahedra, from 98,304 of the constructor, this means that the $\approx 73\%$ of the tetrahedra in the geometrical constructor are background. Figure 3.10(a) shows the geometrical constructor split into two with the hollow part where the tissues used to be. Figure 3.10(b) shows a zoom of the volume meshes of interest.


(a)

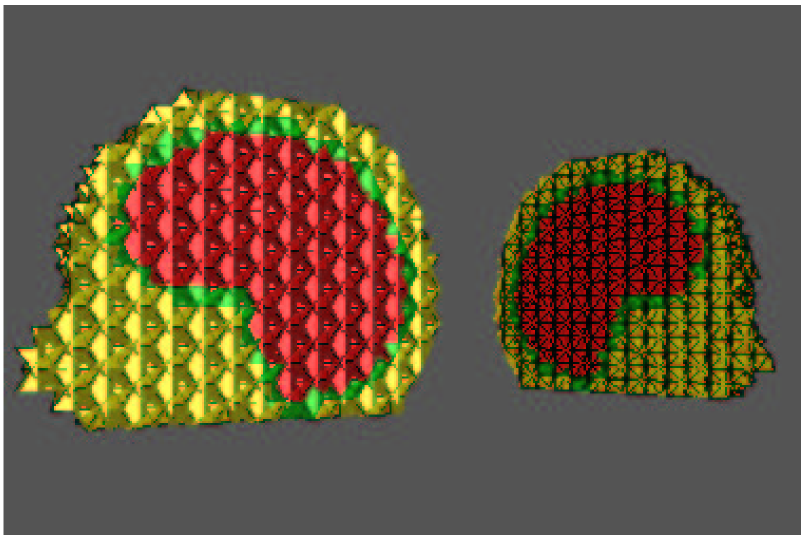

(b)

Figure 3.10: ART volume meshes.(a) Background and (b) a zoom of the head tissues.

3.4 Summary

We found that the surface ART meshes have a high quality shape since they are extracted from almost regular tetrahedra but surfaces do not describe properly the tissues of interest. Chamfer distance in ART brain mesh showed a mean distance of 4.79 pixels with a high standard deviation of 3.75 pixels. This measurement in the other two tissues does not have sense since surfaces were not properly extracted. Although the rest of the geometrical parameters described in section 3.1.2 cannot be applied to ART meshes, since vertices are not on the surface, these constitute useful tools for further surface analysis after the mesh refinement.

In volumetric ART meshes, we also found a very high shape quality and well-shaped dihedral angle distribution. Apart of the noticeable disproportion in number of tetrahedra, between tissues and background, it can be seen from Figure 3.10(a) that the tissues are not properly centered respect with to the geometrical constructor, this makes difficult the setting of the scale input parameter when the meshes are generated (see Appendix A for detail on how to run the ART mesh programs), therefore it is not possible to use this background in a effective way. This can be also related with the long time consuming of the process (about 45 min for $n = 4$, only for the homotopic labelling), since we suppose that the homotopic labelling runs over the whole geometrical constructor looking for head tissues. Since most of the tetrahedra are background, the program is processing a large part of the geometrical constructor which does not contain tissue, this is a misuse of processing time.

Chapter 4

State of the art on 3D mesh refinement

Mesh refinement is, generally, a technique used to improve on the one hand the geometrical mesh approximation when a worth visualisation is important, and on the other hand, the local mesh refinement which is critical to improve the efficiency to approximate the solution of partial differential equations in many modelling and simulation applications. Adaptivity of the mesh is particularly important in three dimensional modelling, such as MEG/EEG localisation, since the mesh size and computational cost grow very rapidly when a uniform refinement is used.

There are two main processes that undergo on local adaptive refinement: the refinement of a subset of elements based generally, in local error indicators, and the consistent transitions between refined and unrefined elements, the so called *mesh conformity.* In this section we will review the main algorithms that have been used on refinement in two and three dimensions, based on triangular and tetrahedral elements respectively.

In two dimensions, Bank and Sherman [1979] refine locally by subdividing certain triangles into four similar sub-triangles by connecting the mid-points of the sides of the "parent" triangle. The conformity of the mesh is ensured by approximate subdivision of adjacent elements. Some schemes are based on edge bisection, for example, the 4-T algorithm of Rivara [1987]: first the longest edge is bisected, and the mid-edge point is connected to the opposite vertex, and then the newly formed vertex is used to subdivide the initial triangle in four. In this algorithm, the angles of the triangles in a resulting locally refined grid are uniformly bounded away from 0 and π. Additional refinement of adjacent triangles is again necessary to ensure the conformity of the mesh, but this refinement is also made based on bisecting the longest edge in the Rivara algorithms.

Another approach of Rivara's algorithm [Rivara and Levin, 1992] is to bisect the triangle by the longest side and bisect as well as those triangles that do not conform until it converges, using a longest edge propagation path (LEPP), the difference between the above mentioned method (of the same author) and this, is that triangles are subdivided in two instead of four, only those triangles that need to conform the mesh are bisected again (for more detail see subsection 5.1.1).

Another related method is the "newest vertex bisection" introduced by Mitchell [1992]. The triangle edge to be bisected is identified without any computation. Only four similarity classes of triangles and only eight distinct angles are created by this method. Hence the important condition of being bounded away from 0 and π is again satisfied. Additional refinement is also necessary in this algorithm to ensure the conformity of the mesh. The newest bisection is equivalent to the 4-T Rivara algorithm in the case in which the bisected edge coincides with the longest-edge.

There are other schemes which introduce different transition elements to solve the problem of mesh conformity. Other techniques use constraints on the approximation rather than subdividing adjacent neighbour elements [Carey, 1976]. Ruppert [1995]'s algorithm produces Delaunay meshes with no small angles, using relatively few triangles and allowing the density of triangles to vary quickly over short distances. The basic idea of the algorithm is to maintain a Delaunay triangulation, making local improvements in order to remove the skinny triangles. Each improvement involve adding a new vertex to the triangulation and retriangulating.

After Ruppers's work, Shewchuk [1996a] introduce his algorithm (which implementation is called **Triangle**, `http://www-2.cs.cmu.edu/~quake/triangle.html`) for 2D mesh generation and construction of Delaunay triangulations, constrained Delaunay triangulations and Voronoi diagrams. His algorithm guarantees quality meshes having no small angles and are generated using Ruppert's Delaunay refinement algorithm. The features of the implementation of the algorithm include user-specified constraints on angles and triangle areas, user-specified holes and concavities, and the economical use of exact arithmetic to improve robustness (see subsection 5.2.1).

In three dimensions there are two main approaches for subdividing a single tetrahedron: octasection and bisection. Octasection methods simultaneously create eight descendants for each tetrahedron. Bey [1995] first connects the edges of each triangle face as in the two dimensional Bank refinement, then cuts off four sub-tetrahedra at the corners which are similar to the original one. The Bey's algorithm cuts the interior octahedron into four more sub-tetrahedra. Since this algorithm considers only the number of nodes at the mid-points of the edges, not the relative position of these nodes, for partial refinement of a tetrahedron there are $2^6 = 64$ possibilities. To handle the transition for mesh conformity, he considers four patterns that cover 25 of the 64 possibilities. The remaining 39 possibilities are refined to 8 sub-tetrahedra.

Methods based on bisection can also be devised easily to subdivide each tetrahedron in eight, but the primary stage consists in bisecting the tetrahedron in two. Bänsch [1991] presents an algorithm based on the selection of an edge as a "global refinement edge" in each tetrahedron, but imposes small perturbations of the coordinates of the nodes to avoid incompatibilities. The algorithm presented by Rivara and Levin [1992] is based on the longest edge bisection also in 3D. Mathematical proofs of the non-degeneracy of the grids created by this scheme have not yet been developed in 3D, although experiments suggested this property holds. Liu and Joe [1994a] and Liu and Joe [1995] present an algorithm (called QLRB) similar to that of Bänsch. They classify the tetrahedra in four types and set up the types of edges depending on the type of tetrahedron. The bisected edges are chosen without computation. This order depends on the assignation of types of the new edges. In this sense it can be said that the QLRB algorithm is the generalisation to 3D of the Mitchell algorithm. In addition, a shape measure is introduced. The number of similarity classes is proved to be bounded and therefore the meshes cannot degenerate.

Several similar studies are reported in the literature. For instance, a recursive approach is proposed by Kossaczkỳ [1994]. This algorithm imposes certain restrictions and preprocessing in the initial mesh . The 3D algorithm is equivalent to that given in [Bänsch, 1991]. Maubach [1995] developed an algorithm for n-simplicial grids generated by reflection. Although the algorithm is valid in any dimension and the number of similarity classes is bounded, it cannot be applied for a general tetrahedral grid. An additional closure refinement is needed to avoid incompatibilities. Mukherjee [1996] has presented an algorithm equivalent to [Bänsch, 1991] and [Liu and Joe, 1994a], and proves the equivalence with [Maubach, 1995].

The algorithm presented by Plaza *et al.* [2000] is also based on bisection. Although it shows similar behaviour to those cited above (Bänsch [1991]; Kossaczkỳ [1994]; Liu and Joe [1994a, 1995]; Mukherjee [1996]) the point of view is different since the three dimensional approach is now based on the two dimensional one applied to the skeleton of the triangulation. The two dimensional version is equivalent to the 4-T Rivara algorithm, and the three dimensional one is the generalisation to the three dimensions of the 4-T algorithm. The algorithm can be applied to any valid initial mesh without any restrictions on the shape of the tetrahedra, since it is based on a previous classification of the edges based on their length. Moreover, because of the underlying spatial recursion approach, these ideas can be extended to obtain local refinement algorithms in higher dimensions.

Recently, Molinari and Ortiz [2002] develop a 3D mesh refinement strategy based on Rivara's longest-edge propagation path (LEPP) by bisection algorithm. The algorithm targets for bisection an entire set of elements defined by a longest-edge incidence criterion. It guarantees that the aspect ratio of elements remains above a certain lower bound regardless of the number of applications of the algorithm. Although it is not theoretically prove, in practice has shown that it converges after a relatively small number of refinements (see subsection 5.1.2). In addition Molinari and Ortiz [2002] also develop a strategy for mesh coarsening, or unrefinement, which is based on the elimination of elements by edge-collapse. The authors use local retriangulation tools to optimise the quality of the mesh and it is also used to eliminate bad elements such as silvers. This algorithm is based on a parallel implementation made by De Cougny and Shephard [1999] (see subsection 5.1.3).

Finally, following the Delaunay constrained refinement strategy describe above for 2D [Shewchuk, 1996a], Shewchuk [1998] develop the extension of this strategy in 3D by maintaining a Delaunay or constrained Delaunay tetrahedrisation, which is refined by inserting carefully placed vertices until the mesh meets constraints on tetrahedra quality and size (volume). The algorithm generates meshes whose tetrahedra have circumradius-to-shortest-edge ratios no greater than the bound $B = 2$. Upon relaxing B to be greater than two, one can also guarantee good grading (see section 5.2.2).

In the present work we will follow and test the two main different strategies that have been applied in practice for modelling problems, in order to compare them and establish which of this two is more suitable to our particular problem of MEG/EEG localisation. These are: the subdivision refinement of ART meshes based on Rivara bisection algorithm ([Rivara and Levin, 1992; Molinari, 2000]) and the Delaunay constraint refinement strategy of Shewchuk [1998]. These topics will be discuss in more detail in Chapter 5.

4.1 Software available for mesh generation and refinement

During our research we did find a large scientific community dedicated to mesh generation problems and we also found a large amount of software develop for particular applications. Most of the modelling applications are in areas of computer flow dynamics (CDF), thermal, structural and environmental modellings, where geometries are generally more simples than head tissues.

We extract from the information compiled at `http://www.andrew.cmu.edu/user/sowen/software/tetrahedra.html` about software availability, particularly in the area of 3D tetrahedral mesh generation and refinement, a list of commercial and public domain software. Table 4.1 contains the commercial software characteristics, and Table 4.2 contains the features of free public domain software.

Name	**Company**	**Application**	**Algorithm**	**Platform**
ANSYS	ANSYS, Inc.	CFD,Thermal	Delaunay, Adv. Front	UNIX/Wind.
Design Space	ANSYS, Inc.	Structural, Thermal	Delaunay	Windows
Gambit	Fluent Inc.	CFD	Delaunay. Adv. Front	UNIX/Wind.
GMS	EMRC, BYU	Environmental	Delaunay	UNIX/Wind.
HOMARD	Electricité de France	Structural, CFD	Delaunay	UNIX/Wind.
ICEM CFD	ICEM CFD	Structural, CFD	Octree	UNIX/Wind.
MESH	ISE Zurich	Semiconductors Siml.	Octree	UNIX/Wind.
TGrid	Fluent Inc.	CFD	Delaunay, Adv. Front	UNIX/Wind.
TrueGrid	XYZ Sci. Apps., Inc.	CFD, Thermal	Delaunay	UNIX/Wind.

Table 4.1:

Commercial software for tetrahedral mesh generation and refinement.
Information extracted from:
`http://www.andrew.cmu.edu/user/sowen/software/tetrahedra.html`

Name	**Company**	**Functionality**	**Algorithm**	**Platform**	**Source Code**
COG	WIAS Berlin	Restricted	Delaunay	UNIX	Yes
Geompack90	ZCS, Inc.	Restricted	Delaunay	UNIX/Wind.	Yes
GMSH	E.P.Mont.U.Li.	Non-Rest.	Delaunay	UNIX/Wind.	Yes
VGRID	NASA	Non-Rest.	Advan.Front	UNIX	No
Tetgen	Berlios	Non-Rest.	Delaunay	UNIX	Yes

Table 4.2:

Public software for tetrahedral mesh generation and refinement.
Information extracted from:
`http://www.andrew.cmu.edu/user/sowen/software/tetrahedra.html`

Chapter 5

Refinement approaches studied in this work

Mesh refinement is an iterative process to determine where to place the mesh points and how to connect them. Popular refinement methods that have been implemented and used in numerical modelling include Delaunay refinement and the edge bisection methods, as it has been reviewed in last Chapter.

Delaunay refinement and edge bisection are similar in spirit. They need to begin with a triangular mesh for the input domain. Delaunay refinement uses normally a Delaunay triangulation as an input, and bisection refinement could use Delaunay or other well-shaped triangulation like the ART meshes. Given a shape measure condition (e.g. minimising the maximum angles or aspect ratio, or maximising the minimum angles), they proceed iteratively: search the element with the worst condition (or an estimation error below a tolerance) and add a local point to improve its condition. In 2D, Delaunay adds the circumcenter of the worst triangle, whereas bisection adds a point to the middle of the longest side. They have their own rules for finding the regions that are close to the boundary. Delaunay refinement generally demonstrates a better ratio and size quality, as well as better visual effect. Edge bisection uses a more local procedure for point placement and re-triangulation than Delaunay refinement. Moreover, 2D and 3D cases are slightly different since the extension to 3D methods are not trivial, as we will discuss in the next two sections.

However, since mesh adaptation is inextricably tied to the mechanics and physics of the problem, the study of mesh construction and mesh adaptation cannot be viewed only from a purely geometrical point of view, although the geometrical evaluation presented in Chapter 3 is the first step to achieve. In this Chapter we will described the main idea of each of the approaches and in the next Chapter we will deal with the geometrical comparisons.

5.1 Bisection refinement on 2D and 3D meshes

5.1.1 Rivara's bisection on 2D meshes

Definition 1. The *longest edge bisection* of a triangle t is the partition of the triangle by the mid-point of its longest edge and the opposite vertex. The neighbour of t is the triangle t^* which shares with t the longest edge of t. A point will be *non-conforming* if it is an interior point of the side of one triangle and a common vertex of the two other triangles (points 1 and 2 Figure 5.1(c))[Rivara and Levin, 1992].

Forward longest edge propagation path (LEPP):

Definition 2. For any triangle t_0 of any conforming triangulation T, the longest-edge propagation path of t_0 will be the ordered list of all the triangles $t_0, t_1, t_2, \ldots, t_{n-1}, t_n$, such that t_i is the neighbour triangle of t_{i-1}, by the longest side of t_{i-1}, for $i = 1, 2, \ldots, n$.

Proposition. For any unstructured, conforming, non-degenerate triangulation of any bounded 2-dimensional geometry Ω, the following properties hold:

1. The longest-edge propagation path $t_0, t_1, \ldots, t_n$ of any triangle t_0 is always finite.
2. The triangles $t_0, t_1, \ldots, t_n$ have strictly increasing longest side (if $n > 1$).
3. For the triangle t_n of the longest-edge propagation path of any triangle t_0, it holds that either:
   (a) t_n has its longest side along the boundary, and this is greater than the longest side of t_{n-1}, or
   (b) t_n and t_{n-1} share the same common longest side.

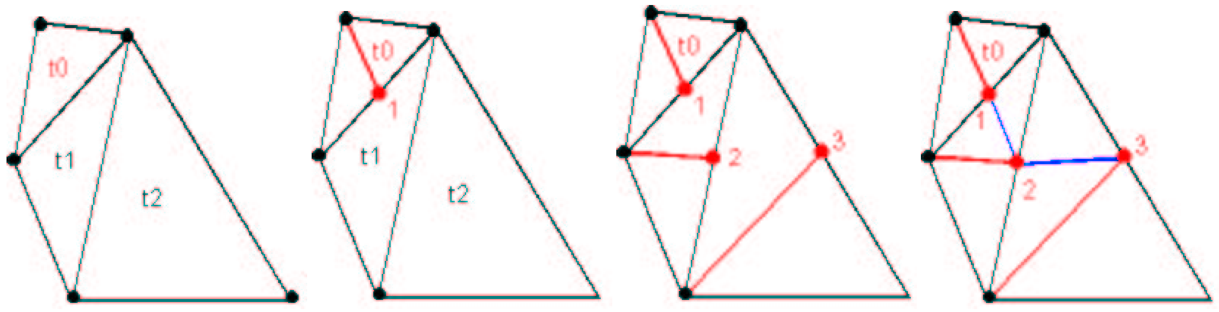
Definition 3. We shall say that two adjacent triangles t, t^* are a pair of terminal triangles if they share their respective (common) longest side.

Definition 4. We shall say that t is a terminal boundary triangle if its longest side is a segment of the boundary of Ω.

To illustrate the idea in terms of refinement propagation, consider the triangulation of Figure 5.1(a) where t_0 is the triangle to be refined. We refined t_0 and its neighbour and so on iteratively until the last two triangles share the same longest side. In this way the refinement propagation stops (Figure 5.1(c)). The same idea needs to be applied in order to conform the set of non-conforming points generated, in the inverse of the order in which they were created. Figure 5.1(d) shows the final refined triangulation.

Let be T a conforming triangulation, the algorithm can be schematically described as follows:

```
Longest-edge-bisection (T,t)
Perform a longest-edge bisection of t
Let P be the point generated
```



(a) (b) (c) (d)

Figure 5.1: Rivara's 2D forward propagation path bisection method (LEPP).

```
While P is non-conforming then do
   Find the neighbour t* of t (by the side containing P)
   Longest-edge-bisection (T,t)
```

Backward Longest edge propagation path:

Rivara [1996] introduced an improved of the latter algorithm and called it backward longest-edge refinement, which makes also use of the LEPP concept. The new algorithm produces the same triangulation without generating intermediate non-conforming points in the mesh which makes it easy to implement.

```
Backward-longest-edge-bisection (T,t)
While t remains without being bisected do
   Find the LEPP(t)
   If t*, the last triangle of the LEPP(t),
      is terminal boundary triangle, bisect t*
   Else bisect the (last) pair of terminal triangles of the LEPP(t).
```

For illustration Figure 5.2 shows the refinement of triangle t_0 over the initial triangulation in Figure 5.2(a) (with associated $LEPP(t_0) = t_0, t_1, t_2$). The triangulations (b) and (c) illustrate the first two steps of the Backward-longest-edge-bisection procedure. Triangulation (d) is the final mesh obtained. Note that the new vertices have been enumerated in the order they were created.

The repetitive use of the Backward-longest-edge-bisection procedure in order to refine t_0 and its descendants (triangles nested in t_0) tends to produce local quasi-equilateral triangulations.

Remarks [Rivara and Inostroza, 1997]:

1. Both algorithms are equivalent in the sense that they produce the same refined mesh at the end.

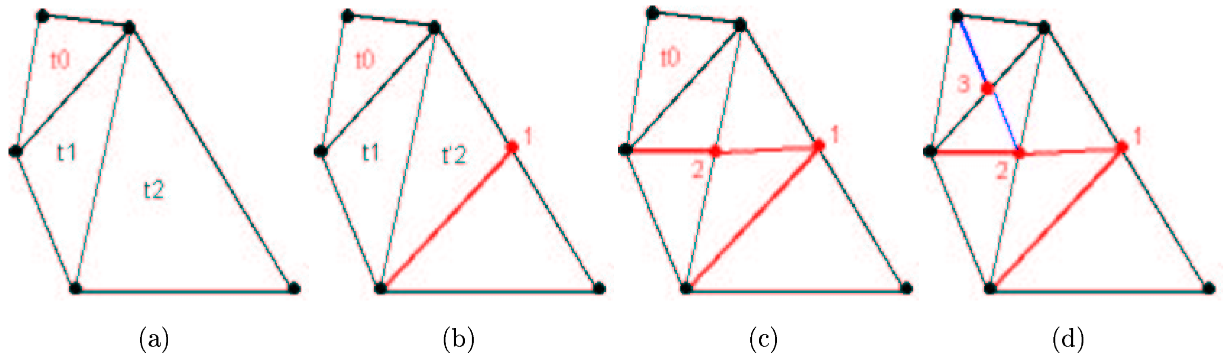


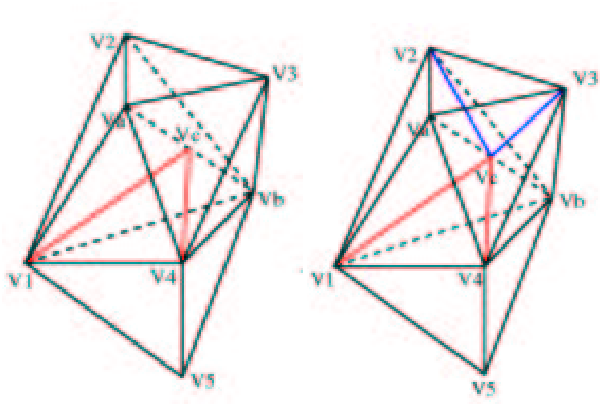
Figure 5.2: Rivara's 2D backward propagation path bisection method.

2. Both algorithms use the concept of LEPP introduced.
3. For any conforming triangulation T, the repetitive use of the Backward-longest-edge-bisection procedure in order to refine t_0 and its descendants (triangles nested in t_0) covers, in a monotonically increasing form, the area of t_0 with quasi-equilateral triangles.
4. The smallest angle α_t of any triangle t obtained through this process, satisfies that $\alpha_t \geq \alpha_0/2$, where α_0 is the smallest angle of t_0.
5. The algorithm guarantees the construction of good quality irregular and nested triangulations, with linear time complexity, provided that an initial good quality triangulation is used.
6. These ideas can be extended to 3D (see subsection 5.1.2).

The appealing properties of this bisection method is that it guarantees that the aspect ratio of the elements remains above certain lower bound regardless of the number of applications of the algorithm. And it also guarantees termination after a finite, typically small, number of element subdivisions.

5.1.2 Extension of Rivara's bisection on 3D meshes

Rivara and Levin [1992] also proposed the extension of the LEPP to 3D. Let the tetrahedron K_0 be the target for bisection. The tetrahedron $K_0 = \{V_1, V_4, V_b, V_a\}$ (Figure 5.3(a)), is bisected (by its longest edge) by a plane determined by the mid-point V_c of the longest edge $e = \{V_a, V_b\}$ in Figure 5.3(a), and the two vertices which are opposite to the bisected edge (V_1 and V_4 in Figure 5.3(a)). Accordingly, each element K^* sharing the non-conforming edge e is bisected in the same way (by its own longest edge) generating a set of new non-conforming edges (see Figure 5.3(b)). The elements sharing these non-conforming edges (in a ring around them) are in turn bisected in the same way, and so on, until propagation stops.



(a) (b)

Figure 5.3: Rivara's 3D bisection method.

The non-conforming edges are treated in the same way as described before, in the reverse order in which they were created in order to ensure the management of intermediate non-conforming edges. Thus, the 3D algorithm to refine tetrahedron K over a conforming tetrahedral mesh can be described as follow:

```
Tetrahedron refinement (K,T)
Perform a generalised bisection of K
(Let P be the point generated)
While P is non-conforming then do
   Find the neighbour set N_K of the tetrahedron K
   For each K* in N_T do
      Tetrahedron refinement (K*,T)
```

The neighbour set of K is the polyhedron formed by the union of all the tetrahedra that share with K its longest edge. Geometrically, the conforming process can be seen as the refinement of a set of polyhedral neighbour sets, where each has its axis longer than the preceding one in the recursion. This property ensures that the algorithm terminates with a conforming mesh in a finite number of steps. However, is has not been shown mathematically that the algorithm ensures that the tetrahedron constructed does not degenerate when the process proceeds to infinity. Rivara and Levin [1992] have reported numerical results that show that the algorithm is a reliable tool from a practical point of view.

The generation of the propagation path in 3D is not trivial. Molinari [2000] has suggested an algorithm to represent the propagation path in 3D using graphs which he called Longest-edge propagation graph (LEPG). Each vertex in the directed graph represents an element of the mesh. The vertices are linked by directed edges which are the longest edges of the elements from which they emanate. An example is shown in Figure 5.4 were the graph starts from tetrahedron K_1 which is adjacent by its longest edge e_1 to five other tetrahedra: K_2, K_3, K_4, K_5, and K_6.

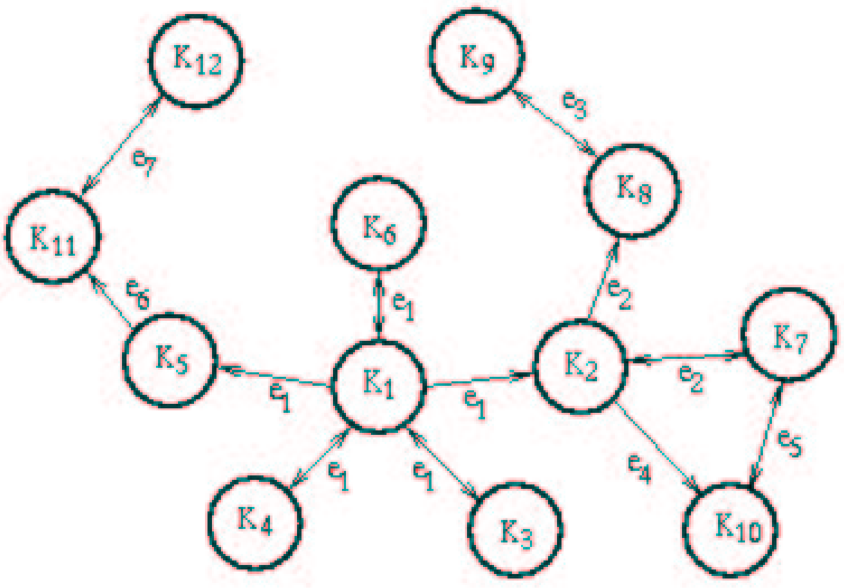


Figure 5.4: Longest-edge propagation graph (LEPG).

From this example we can see that K_2 has an edge e_2 which is larger than e_1. This edge is adjacent to two other tetrahedra, K_7 and K_8. The longest edge of K_7 is also e_2 and thus is a terminal vertex of the graph. Therefore, the edge joining the vertices K_2 and K_7 has two directions. K_7 is our first candidate for subdivision. K_8, on the other hand, has a larger edge e_3 which brings the algorithm to the terminal tetrahedron K_9. In the case of the vertex K_2, it has an edge e_4 shorter then e_2 but larger than e_1, thus the element K_{10} which is adjacent to K_2 along e_4 is a new member of the LEPG. K_{10} has another longest edge e_5, which belongs to K_7, a vertex that was previously encountered in the graph.

Suppose that the tetrahedron K_1 is the target to be refined. Then the LEPG is generated recursively by addition of new tetrahedra at the end of the path in accordance with the following rules:

1. Initialisation: $LEPG = \{K_1\}$.

2. Let $LEPG = \{K_1, K_2, \ldots, K_{i+1}\}$ after i iterations of the algorithm. Then the algorithm terminates if K_{i+1} is a terminal tetrahedron, *i.e.*, if the following condition is satisfied:

   - The longest edge of K_{i+1} is shared with K_i.

3. If $LEPG = \{K_1, K_2, \ldots, K_{i+1}\}$ and K_{i+1} is not a terminal tetrahedron, then the new path is $LEPG = \{K_1, K_2, \ldots, K_{i+1}, K_{i+2}\}$, where K_{i+2} is the tetrahedron incident to K_{i+1} along its longest edge.

4. If $LEPG = \{K_1, K_2, \ldots, K_{i+1}\}$ and K_{i+1} is a terminal tetrahedron, then K_{i+1} is bisected along its longest edge and the new path is $LEPG = \{K_1, K_2, \ldots, K_i\}$. This step results in the addition of new nodes to the mesh.

5. The previous rules are repeated until K_1 is bisected.

Figure 5.5 describes the order in which the vertices of Figure 5.4 are visited. Each line corresponds to a bisection of a tetrahedron.

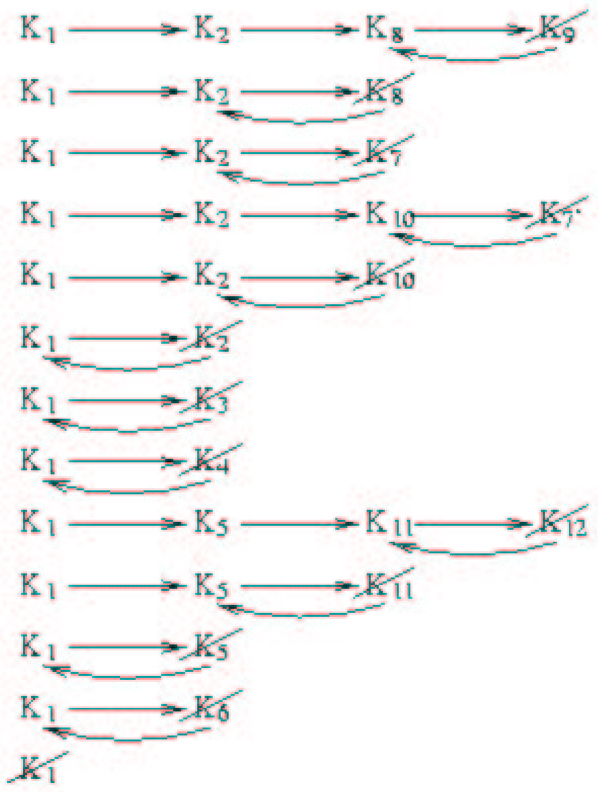


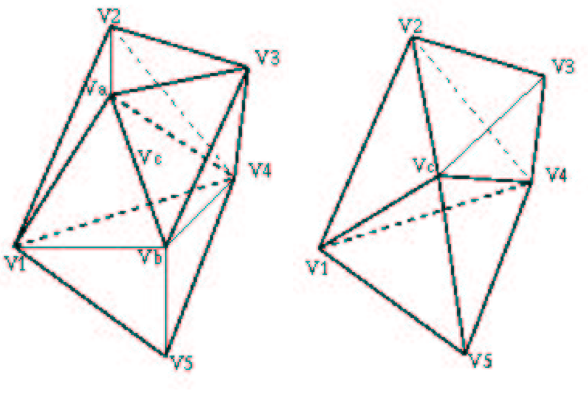
Figure 5.5: Bisection of tetrahedra, following the LEPG.

5.1.3 Derefinement or coarsening on 3D meshes

Coarsening is often useful, particularly in time-dependent computations, to unrefined regions of the mesh that were refined in previous steps of the computation, it also can be used to eliminate bad elements like silver tetrahedra. For the particular problem of MEG/EEG localisation can be used to eliminate tetrahedra were there is a low neuronal activity depending on the cognitive task being tested.

Some schemes that use subdivision patterns or bisection for refinement can also coarsen by reversing the refinement process. However, reversing the refinement cannot be done arbitrarily since some triangles or tetrahedra merged cannot still guarantee to form a conforming triangulation. Edge collapsing, which consists in "pulling" one end vertex to the other, along with the connected regions, can be used for coarsening purposes or with the idea of eliminating bad shape elements.

Molinari [2000], based on the work of De Cougny and Shephard [1999], proposed an edge collapsing algorithm to coarsen and/or eliminate bad tetrahedron from an already bisected refined mesh. Figure 5.6(a) illustrates the collapse operation on a simple three dimensional mesh. The initial mesh contains seven nodes and five tetrahedra: $t_1 = \{V_a, V_3, V_4, V_2\}$, $t_2 = \{V_1, V_a, V_4, V_2\}$, $t_3 = \{V_1, V_b, V_5, V_4\}$, $t_4 = \{V_1, V_b, V_4, V_a\}$ and $t_5 = \{V_a, V_b, V_4, V_3\}$.



(a) (b)

Figure 5.6: Edge collapse operation in 3D.

The collapse of edge $\{V_a, V_b\}$ onto the mid-point V_c eliminates the tetrahedra t_4 and t_5 and results in a coarser mesh consisting of six nodes and three tetrahedra t'_1, t'_2 and t'_3 (Figure 5.6(b)). These tetrahedra are deformed from t_1, t_2 and t_3 by the dragging of V_a and V_b towards the mid-point V_c. Particular care has to be taken for edges that belong to the boundaries or material interfaces, for example:

- the two vertices are inside the domain,
- one vertex is inside the domain, the other is on the boundary,
- the two vertices are on the boundary, but they are no corner vertices,
- the two vertices are on the boundary and one is a corner vertex, etc.

In three dimensions the number of possibilities is larger and are summarised in Table 5.1.

Edge	V_1	V_2	Operation
interior	interior	interior	collapse to mid-point
interior	boundary	interior	collapse to V_1
interior	boundary	boundary	no collapse
boundary face	interior face	interior face	collapse to mid-point
boundary face	boundary face	interior face	collapse to V_1
boundary face	boundary face	boundary face	no collapse
global edge	interior edge	interior edge	collapse to mid-point
global edge	boundary edge	interior edge	collapse to V_1
global edge	boundary edge	boundary edge	no collapse

Table 5.1: Possible cases for edge collapse in 3D [Molinari, 2000].

A boundary edge can be also identified as a global edge when the dihedral angle opposite to (subtended by) the boundary at the edge is under a given tolerance (like in a sharp edge) or over a given tolerance (like in a groove). A boundary vertex is identified as a global vertex if it is at the intersection of two global edges.

The operations in the table preserve the topology of the domain and sharp geometrical features of the boundary such as global edges and vertices, which depend on the choice of tolerance.

Molinari [2000] reported that some times the edge collapse operation worsens the aspect ratio of the elements, to avoid this he introduced and additional criterion based on the quality measure that Rivara and Levin [1992] used: minimum of the solid angles at its four vertices, which ranges from 0°, for silver, to 45° for a regular tetrahedron. Therefore, the edge collapse is performed if one or more tetrahedra incident on an edge are silvers, or if all the following conditions are met simultaneously:

1. The edge error indicator falls below the tolerance.
2. Edge collapse is geometrically possible (Table 5.1).
3. The average quality of the product tetrahedra exceeds the average quality of the original tetrahedra.

4. The quality of the worst product tetrahedra exceeds the quality of the worst original tetrahedron.
5. The total volume of the product tetrahedra is close to the total volume of the original tetrahedra.

5.2 Delaunay 3D mesh generation and refinement

One of the most popular techniques for point set and element generation for FEM applications are the Delaunay triangulation and the Delaunay refinement. In order to apply this method a definition of a piecewise linear system (PLS) in 2D, or a triangular surface in 3D is needed. In order to generate a polygonal based representation of the human cortical surfaces it is possible to apply methods based on surface reconstruction from *contours*, like the program NUAGES written by Geiger [1993], methods based on voxel reconstruction like the *marching cube* algorithm [Lorensen and Cline, 1987], or methods based on *cloud points* like Power Crust suggested by Amenta *et al.* [2001]. Surface reconstruction will not be discuss further here except to say that it has been extensively studied and some methods are a combinations of the mentioned above. Unfortunately the quality of the generated surface influences heavily in the quality of the 3D tetrahedrisation, when Delaunay tetrahedrisation methods are used.

5.2.1 Delaunay refinement in 2D

The idea can be summarised in adding Steiner points into the domain of the initial triangulation and an improved triangulation is generated. This procedure is repeated iteratively until a quality mesh is obtained.

Suppose $P = \{p_1, \ldots, p_n\}$ is a set of points in $\Re^d$. A simplex defined by $(d+1)$ affinely independent points from P, is a *Delaunay simplex* if the circumsphere of the simplex contains no point from P in its interior. The union of all Delaunay simplices forms the *Delaunay diagram* $DT(P)$. If P is not degenerate, then $DT(P)$ is a triangulation of the convex hull of P (Figure 5.7).

The geometric dual of the Delaunay diagram is the *Voronoi diagram*, which consists of a set of polyhedra $V_1, \ldots, V_n$, one for each point in P. V_i is called the *Voronoi cell* of p_i, and p_i is called the center of V_i. Geometrically, V_i is the set of points in $\Re^d$ whose distance to p_i is less than or equal to that of any other point in P.

One of the desired properties for a mesh that the Delaunay triangulation gives is that among all triangulations of a set of points in 2D, the Delaunay triangulation maximises the minimum angle. In any dimension, it always contains the nearest neighbour graph of the set of points, *i.e.* in the Delaunay triangulation, every point is directly connected with its nearest neighbours. The Delaunay triangulation also contains the minimum spanning tree connecting to the set of points.

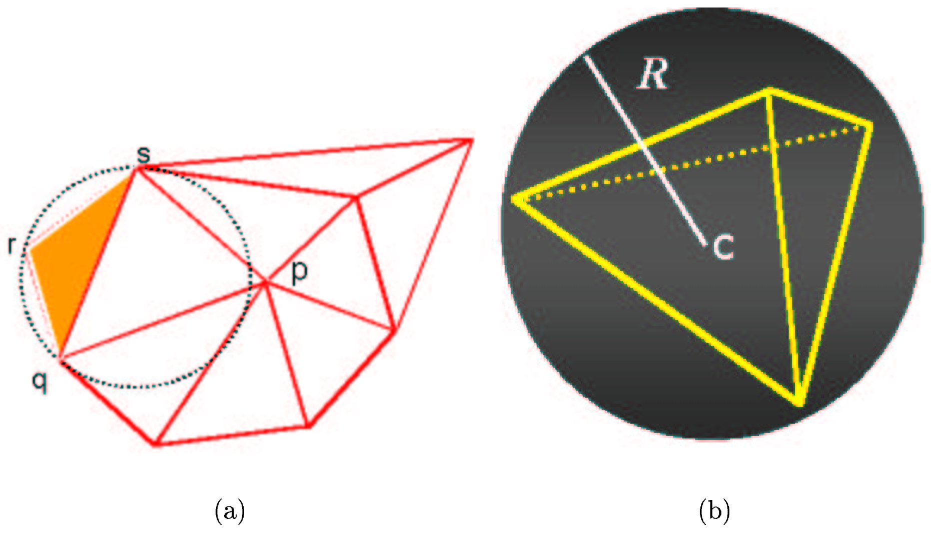


Figure 5.7: Delaunay definition. (a) Two dimension: circle $C(r)$ is empty if it contains no vertices, and (b) three dimension: circumsphere $B(c, R)$ is empty if it contains no vertices.

To generate the Delaunay-like triangulation for a polygonal domain Ω that retains the boundary, we would use the *constrained Delaunay triangulation*. Let P be the set of vertex points of Ω. Two points p and q in P are visible from each other if the line segment pq does not intersect with the interior of any boundary subdomain of Ω. A simplex of $d+1$ linear independent points is a *constrained* Delaunay simplex if its circumsphere contains no points from P in its interior and it is visible to any of these $d+1$ points. The union of all constrained Delaunay simplices forms the *constrained Delaunay diagram* $CDT(\Omega)$.

The Delaunay refinement first constructs $CDT(\Omega)$ and then systematically and adaptively adds Steiner points to improve the mesh quality. Ruppert [1995] developed a Delaunay refinement algorithm for 2D which generates well-shaped meshes:

```
Ruppert Iteration
Input A polygonal (PSLG) domain $\Omega$ in $\Re^2$, and a threshold parameter $\theta$ for the smallest angle.

  1. Let $T = CDT(\Omega)$.
  2. Let $S$ be the triangle in $T$ that has the smallest angle.
  3. While the angle of $S$ is less than $\theta$
     (a) Let $c$ be the circumcenter of $S$.
     (b) If $c$ is not contained in the diameter-circle of any boundary segment $T$, then add $c$ as a new Steiner point.
     (c) Else, add the middle point of that boundary segment as a new Steiner point.
     (d) Update the constrained Delaunay diagram and let $S$ be the simplex $T$ with the smallest angle.
```

Theorem 1. (Ruppert). For any $\theta \leq 20.7^\circ$ and for any polygonal domain Ω in $\Re^2$, the Ruppert Iteration terminates with a mesh where the smallest angle is at least θ. In addition, there is a constant C such as that the size of the mesh returned by the Ruppert Iteration is no more than C times the size of any mesh for Ω whose smallest angle is at least θ.

Shewchuk [1997] develops a GNU software called **Triangle** (`http://www.cs.cmu.edu/ quake/triangle.html`) which generates exact Delaunay triangulations, constrained Delaunay triangulations and quality conforming Delaunay triangulations. It works with a piecewise linear system in 2D. It implements the Ruppert's Delaunay refinement algorithm for 2D meshing. The implementation is very robust due the use of exact arithmetics [Shewchuk, 1996b]. Users can specify constrains on angles and triangle areas. The basic algorithms used in **Triangle** provide a guarantee on the aspect ratio and guarantee that the size of the mesh generated is at most a constant factor of the best possible. It is perhaps the most robust software for Delaunay triangulation and refinement in 2D. **Triangle 1.3** was released on July 1996, it is written in C and it runs on almost all Unix and Linux platforms. One can call the interface of **Triangle** from other programs.

5.2.2 Delaunay refinement in 3D

Mesh generation in 3D is inherently harder than in 2D. The input description of the domain is more complex; the meshes are usually larger; the boundary cases are more complicated; and the proof of correctness is more challenging. Ruppert's results can not be directly extended from 2D to 3D. The main reason is the following: Let M be a well-shaped Delaunay mesh in 3D with a set of points P. Let P' be a set of points obtained from P by a small random perturbation of P. Then, with high probability, the Delaunay triangulation of P' is not well-shaped.

For 3D Delaunay refinement, let S be a simplex in $\Re^d$ and let $shape(S)$ be a quality condition of S, i.e., the radius ratio, the radius-edge ratio, or the radius-shortest-edge ratio of S. Let $d' < d$. Then a d'-dimensional great circle of B is the intersection of B with a d'-dimensional hyperplane passing through the center of B. A simplex s is a boundary simplex in $CDT(\Omega)$ if it is contained in a boundary subdomain of Ω. Suppose s has dimension d'. Then the *diameter-circumsphere* of s is a $(d-1)$-sphere where s is circum-subscribed by only one of its d'-dimensional great circles.

The following is a basic scheme for the Delaunay refinement directly extended from the Ruppert Iteration. Some simple variants of this scheme can be used in any dimension. Let $shape(S)$ be a shape measure of a simplex S.

```
Scheme Delaunay Refinement
Input A polygonal (PLS) domain $\Omega$ in $\Re^d$, and a threshold param-
eter $\theta$ for the shape condition.

    1. Let $T = CDT(\Omega)$.
    2. Let $S$ be the simplex in $T$ with the worst $shape(S)$.
    3. While $shape(S)$ is less than $\theta$
        (a) Let $c$ be the circumcenter of $S$.
        (b) If $c$ is not contained in the diameter-circumsphere of any
            boundary simplex in $T$, then add $c$ as a new Steiner
            point.
        (c) Else, let $s$ be the boundary simplex with the small-
            est dimension such that $c$ is contained in the diameter-
            circumsphere of $s$, add the circum-center of $s$ as a Steiner
            point.
        (d) Update the constrained Delaunay diagram and let $S$ be
            the simplex $T$ with the worst $shape(S)$.
```

Ruppert's refinement is a special case of this scheme when θ is the smallest angle and $d = 2$. Unfortunately, Ruppert's theorem does not directly extends from 2D to 3D if the radius ratio is used; there is no constant that guarantees the termination of the Delaunay refinement procedure. The main problem is that the standard Delaunay refinement does not eliminate silvers.

Another approach to ensure the termination of the Delaunay refinement procedure is to use a shape condition that tolerates silvers. Miller *et al.* [1995] suggested the use of the radius-shortest-edge ratio (instead of the aspect ratio introduce by Frey and George [2000] and describe in section 3.2). They have shown that the radius-shortest-edge ratio is the strongest shape condition that tolerates silvers. We can apply this shape condition to the 3D Delaunay refinement (for $\Omega \in \Re^3$) changing the step 3 of the latter algorithm by "While radius-shortest-edge ratio of S is less than θ", the rest of the scheme is the same.

Recently, Shewchuk [1997] has shown that the above algorithm terminates with a properly chosen constant for the radius-shortest-edge ratio.

Theorem 2. (Shewchuk). For any $\theta \geq 2$, 3D Delaunay Refinement with Radius-Edge Ratio terminates. In addition, the size of the mesh produced is linear in the size of the "optimal" well-shaped mesh for the same domain. Some of the examples he showed in his paper shown dihedral angles bounded between 21° and 149°.

Based on Shewchuk [1997] theoretical work on 3D extension and **Triangle** software, Hang [2001] generates the extension of **Triangle** to 3D meshing GNU software that is called **Tetgen** (`http://tetgen.berlios.de/`), it is written in C++, version 1.0 was released on Agust 2001 and runs on almost all Unix and Linux platforms. **Tetgen** follows exactly the same input and output formats than **Triangle**, although the input can be also a surface mesh. Users can specified constraints on radius-shortest-edge ratio and tetrahedral volume (see Appendix D for technical details).

Tetgen is the software we are using in this work to generate the Delaunay constrained triangulations in 3D for our MRI volume tissues.

5.3 Summary

Since ART meshes are almost regular, we think that by applying the bisection method that maximises the minimum angle we can achieve a better shape quality for the end mesh than with Delaunay methods, since Delaunay rely first on the construction of a Delaunay surface before construct and refine the final tetrahedral mesh. The problem with the ART meshes is that since it is not based on the surface of the volume, the representation of the surfaces is not that exact. However, we can first apply the refinement only on the boundaries and interfaces between tissues and then take this output as our input for the refinement depending on the error estimation of the numerical model or, any other criterion for refinement.

Delaunay refinement has been applied on FEM modeling problems with very satisfactory results, although, at the present, silver tetrahedra can not be avoided. Nonetheless, with ART refinement meshes by bisection, we believe we will not have such a problem. Therefore we need first to evaluate both geometrical models in the numerical model to decide which one is preferable for the problem of MEG/EEG source localisation.

Chapter 6

Geometrical comparison of mesh refinements obtained by bisection and Delaunay approaches

Most of the work reported in the FEM literature deals with generation and refinement of Delaunay 3D meshes. The starting point for this project has been the refinement of almost regular meshes (ART) and the idea of bisection to keep as much as possible the quality of ART elements.

The main objective of this chapter is to test both approaches (described in Chapter 5) with our tissues of interest in order to evaluate their geometrical differences (those presented in Chapter 3) and yield the first conclusion in terms of geometrical quality. The second conclusion is related to the test of both geometrical models into the numerical FEM model, to finally yield the global conclusion of which geometrical model suits better the particular problem of MEG/EEG source localisation. This last test is, at the present, out of the scope of this report.

To achieve our goal we need to implement the Rivara and Levin [1992] bisection method for the refinement of ART geometrical model. To test the Delaunay refinement model, we have found a GNU software called **Tetgen** to generate and refine 3D meshes (see Appendix D and E, respectively, to find technical information about both programs).

In the forthcoming sections we will show the progress achieved at each stage.

6.1 Results on bisection refinement from ART volume meshes

At this stage we have implemented the Rivara and Levin [1992] 2D bisection applied to 3D surface meshes. That is, we have bisected surface mesh of triangular elements. Figure 6.1 shows an example of the algorithm applied to a sphere surface of 32 almost regular triangular elements.

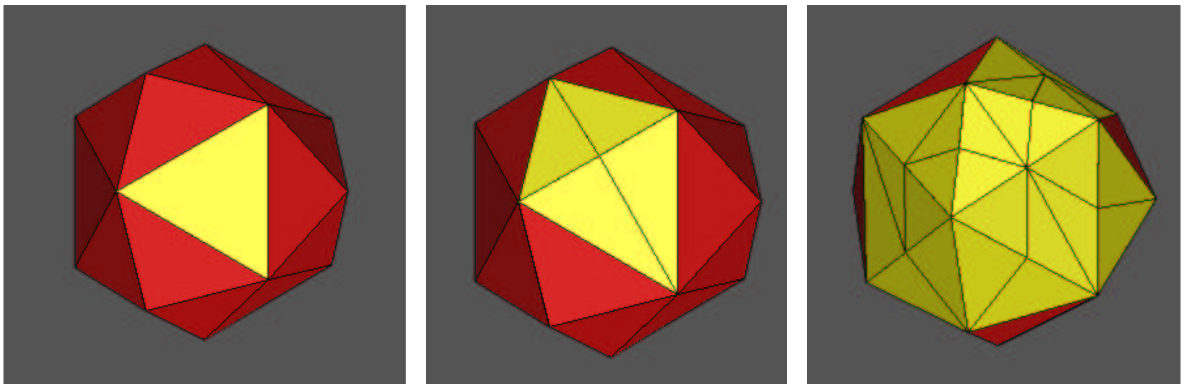


Figure 6.1: Rivara 2D bisection applied to 3D surface. (a) Original surface, (b) after one refinement and (c) after a second refinement. The light elements mark the elements to be refined

Table 6.1 shows the evaluation of the surface as the refinement is taken place. The first row shows the original sphere with 32 almost regular triangles. Planarity and deviation does not change with increasing refinement, whereas smoothness increases. On the other hand shape quality of elements decreases slightly with refinement.

Mesh	**Planarity**	**Deviation**	**Smooth.**	**Quality**	**# Elem.**
	min (mean)	min (mean)	min (mean)	min (mean)	
Orig.	0.5 (0.5)	1.0 (1.0)	0.87 (0.87)	0.79 (0.84)	32
Sphe.1	0.5 (0.5)	1.0 (1.0)	0.87 (0.88)	0.63 (0.82)	34
Sphe.2	0.5 (0.5)	1.0 (1.0)	0.87 (0.89)	0.46 (0.77)	48

Table 6.1:

Surface evaluation parameters. First column shows the mesh type: the first row contains the reference mesh. The the other rows contain first refined mesh and second refined mesh, respectively. # Elem. refers to number of triangles per mesh. The other parameters are dimensionless.

Since we are not able to work with the surface meshes extracted from ART generator, we applied the same algorithm to the reference surface show in Figure 3.5 obtained with the marching cube algorithm. Figure 6.2(a) shows the original mesh, (b) the mesh after the first refinement and (c) after the second refinement. The light triangles indicate the elements to be refined.

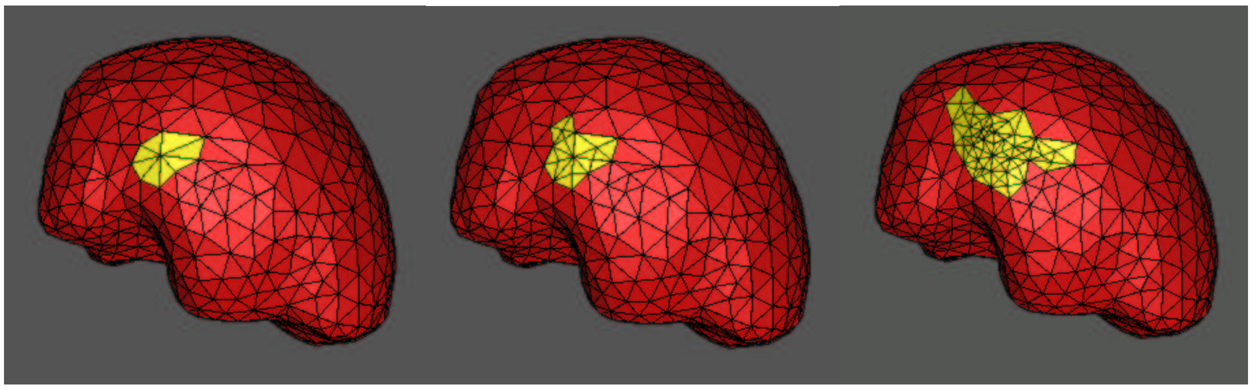


Figure 6.2: Rivara 2D bisection applied to 3D surface. (a) Original surface, (b) after one refinement and (c) after a second refinement. The light elements mark the elements to be refined

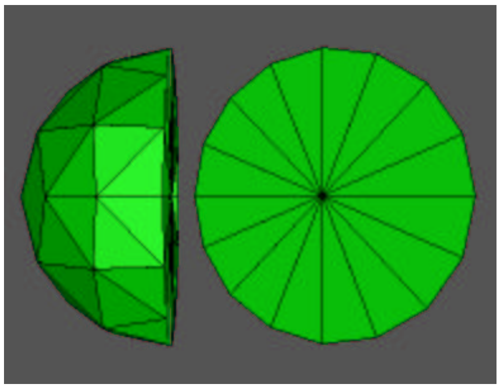
We do not show the table of quality measures since Rivara technique generates well-shaped mesh refinement if the initial mesh is also well-shaped, which is not the case of marching cube surface. This is a temporarily example. We need to generate almost regular surfaces (e.g. Delaunay surfaces) in order to show that the quality of the surfaces does not change dramatically after the bisection refinement. However, we consider that the best strategy for the particular MRI volumes we are working with will be to make the refinement directly from the ART volume mesh and, after this refinement, extract the surface. This is because, if we want to keep as well-shaped tetrahedra quality as possible during the refinement of ART meshes, it will be easer to adjust the whole volume mesh to the surface of the real volumes and then extract the surfaces, rather than do it in the inverse order.

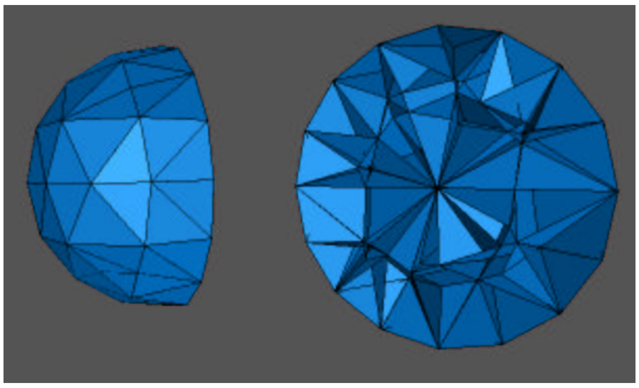
The extension of Rivara and Levin [1992] algorithm in 3D has been described in subsection 5.1.2, and the implementation is still in progress.

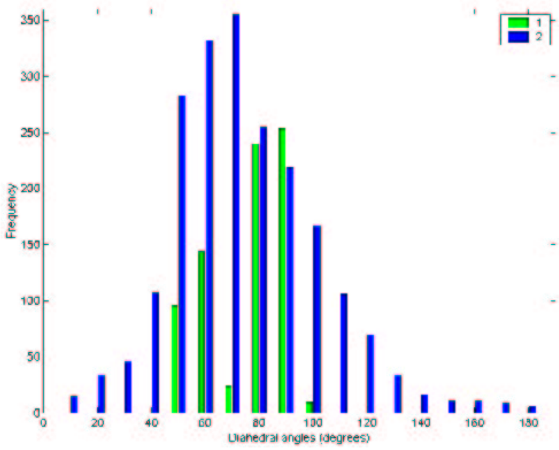
6.2 Results on Delaunay generation and refinement meshes from MRI volumes

One of the drawbacks of Delaunay 3D generation and refinement is that they are based on the volume surface mesh. This constraint is a drawback for our problem since the correctly construction of the volume mesh, in terms of quality, will depend on the quality of the surface mesh. There are two main approaches to generate a surface mesh from the MRI volumes: 1) based on contours [Geiger, 1993] and 2) based on cloud of points [Amenta *et al.*, 2001]. There has been several work into this area which is call "surface reconstruction", and to generate a well-shaped surface from a non-convex and complex volumes, such as head tissues, is not trivial. Furthermore, our problem involves the use of three different tissues in the same model which implies three different material and therefore boundaries between materials that must conform the complete head model.

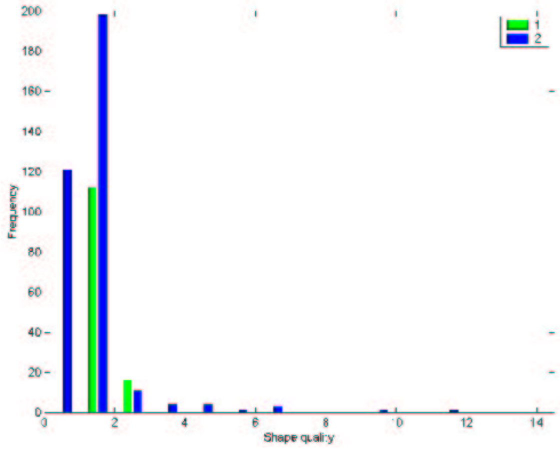
Our first approach to test the **Tetgen** software was to generate a sphere surface mesh generated with marching cube method with only one material as an input to **Tetgen**. Figure 6.3(a) shows the 3D Delaunay tetrahedrisation with 128 tetrahedra, (b) shows the same sphere but this time Dealunay constrained to a quality (1.14) and volume values (3000) given a 346 tetrahedral mesh.


(a)


(b)



(c)



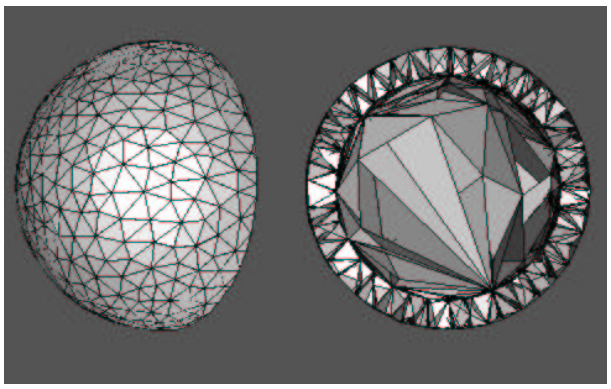
(d)

Figure 6.3: Delaunay 3D mesh with one material. (a) Delaunay tetrahedral mesh, (b) Delaunay constrained tetrahedral mesh, (Q and V), (c) dihedral angle distribution of (a) and (b), and (d) aspect-ratio distribution for (a) and (b).

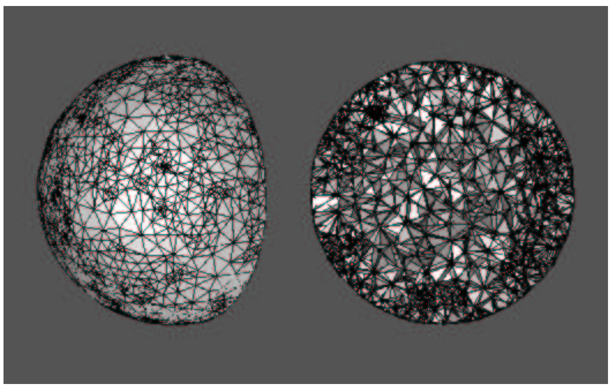
Figure 6.3(c) shows the dihedral distributions for both meshes. For mesh (a), angles range from $47.26° \leq \varphi \leq 90°$ with no bad tetrahedra, whereas for mesh (b) angles range from $2.73° \leq \varphi \leq 174.34°$ with 13 bad elements.

Figure 6.3(d) shows the shape quality distribution of both meshes, where mesh (a) ranges from $1.5 \leq Q \leq 2.55$ and for mesh (b) $1.12 \leq Q \leq 21.79$. Finally, both generation processes took $\approx 0.25sec$. Mesh (b) is the refined Delaunay tetrahedrisation which contains 13 bad elements with wider range of dihedral angles and some of its bad elements with very high radius-shortest-edge ratio.

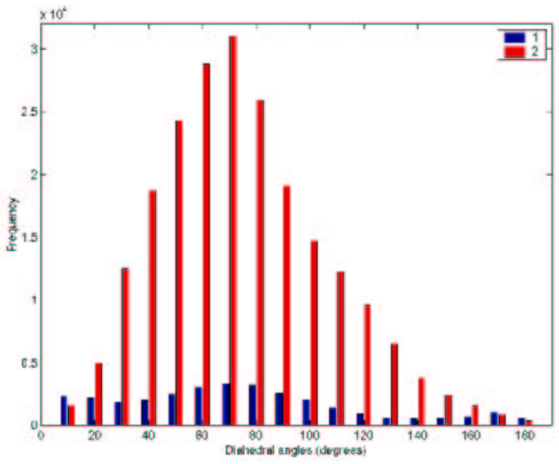
A different example is shown in Figure 6.4 where two different materials are considered.



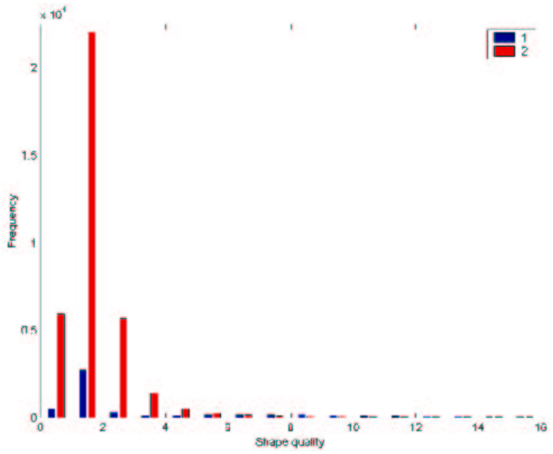
(a)



(b)



(c)



(d)

Figure 6.4: Delaunay 3D mesh with two materials. (a) Delaunay tetrahedral mesh, (b) Delaunay constrained tetrahedral mesh, (Q and V), (c) dihedral anggle distribution of (a) and (b), and (d) aspect-ratio distribution for (a) and (b).

The two concentric surface meshes were generated with the marching cube algortihm as an input for **Tetgen**. Figure 6.4(a) shows de Delaunay triangulation with 5,331 tetrahedra, and Figure 6.4(b) de Delaunay refined triangulation constrain to a quality (1.8) and volume values (100), with 36,538 tetrahedra. Figure 6.4(c) shows the distribution of dihedral angles for both meshes. For mesh (a) angles range from $0.12° \leq \varphi \leq 179.77°$ with 1,722 bad elements, whereas for mesh (b) angles range from $0.29° \leq \varphi \leq 179.17°$ with 2,625 bad elements.

Finally Figure 6.4(d) shows the distributions of the shape quality for both meshes, where mesh (a) has a range of $1.06 \leq Q \leq 446.9$ and for mesh (b) $1.02 \leq Q \leq 225.5$.

Although range of dihedral angles are similar in both cases, for mesh (b) it can be seen that quality shape shows a big mode below 2 and proportionally has less bad elements (2,625/36,538) than mesh (a) (1,722/5,331).

We can consider that Delaunay mesh refinement with a constrain to shape quality and volume can generate better shaped meshes. Although, the number of tetrahedra increases considerably since the algorithm inserts many Steiner points into the surface mesh in order to meet the constraints (mesh (b)). This may be not necessary to accurately represent the geometry. The processing times for these processes were $2.63sec$ for mesh (a) and $20.34sec$ for mesh (b).

6.3 Comparisons

At the present stage we do not have enough material to make the comparison with the MRI volumes. However from the latter examples, we can view some advantages and disadvantages of both methods.

On the one hand, ART meshes need an extremely long processing times, being $\approx 45min$ for a geometrical constructor of 98,304 tetrahedra where only 26,625 are the MRI tissues of interest. The surfaces or boundaries between tissues do not match with the real surfaces. Since we will need to refine first the boundaries to make them match, this process could increase the number of tetrahedra in the boundaries or tissue interfaces, and might not be really needed for the numerical modelling process.

On the other hand, since refinement will begin with a very well-shape mesh (ART), bisection model should give us a well-shaped refined mesh with no silver or other bad shaped elements.

For the Delaunay tetrahedrisation, the first drawback that we mentioned before is the need of a well-shaped (Delaunay e.g.) surface mesh which from MRI volume is not trivial to generate. Mesh element shape can be controlled in quality and volume which it is an advantage for the numerical computations, on the contrary, apparently for such a complex structures, silver and other bad shaped elements might be unavoidable.

Another disadvantage of 3D Delaunay approach is that in order to meet the constraints needed, the algorithm inserts Steiner points into the surface mesh (see Figure 6.4(b)) which makes the mesh increases in number of elements which once more could no be really needed for a useful representation of the volume geometry and could be undesirable from the computation point of view. One really excellent advantage of **Tetgen** implementation is that the processing times are much faster compared with those of ART generation meshes, due to the used of exact arithmetic with the geometric predicates develop by Shewchuk [1996a].

Chapter 7

Conclusion and Future work

Based on the work made at ENST by Dokladal *et al.* [2001] on MRI tissue segmentation and by Pescatore [2001] on almost regular tetrahedral (ART) meshes, the problem of mesh refinement has been address. The first aim of the project is to refine the current mesh in order to have a better geometrical representation of the tissue interfaces (brain, skull and scalp) extracted from MRI data. The next stage is to adapt the mesh locally, to a specific area of the tissues, in order to improve the accuracy of the computation and consequently the processing time for the localisation of the neurological activity.

Since our work is based on ART meshes, the first task that we carry out was the geometrical evaluation of them. We found that the surface ART meshes have a high quality shape since they are extracted from almost regular tetrahedra but surfaces do not describe properly the tissues of interest. Chamfer distance in ART brain mesh showed a mean distance of 4.79 pixels with a high standard deviation of 3.75 pixels. This measurement in the other two tissues does not have sense since surfaces were not properly extracted. Although the rest of the geometrical parameters described in section 3.1.2 cannot be applied to ART meshes, since vertices are not on the surface, these constitute useful tools for further surface analysis after the mesh refinement.

In volumetric ART meshes, we also found a very high shape quality and well-shaped dihedral angle distribution. Apart of the noticeable disproportion in number of tetrahedra, between tissues and background, it can be seen from Figure 3.10(a) that the tissues are not properly centred respect with to the geometrical constructor, this makes difficult the setting of the scale input parameter when the meshes are generated, therefore it is not possible to use this background in a effective way. This can be also related with the long time consuming of the process (about $45min$ for $n = 4$, only for the homotopic labelling), since we suppose that the homotopic labelling runs over the whole geometrical constructor looking for head tissues. Since most of the tetrahedra are background, the program is processing a large part of the geometrical constructor which does not contain tissue, this is a misuse of processing time.

From the review of literature of mesh refinement made in Chapter 4 we found that most of the work that have been done related with tetrahedral mesh generation for FEM modelling purposes are based on constrained Delaunay methods [George, 1997; Shewchuk, 1997], and refinement of triangular and tetrahedral meshes based on bisections [Liu and Joe, 1995; Rivara, 1997; Plaza *et al.*, 2000] or other subdivision techniques [Bey, 1995; Mukherjee, 1996] from an already generated mesh. For some of these techniques there exists already a public domain software to generate tetrahedral meshes based on the surface mesh of the volumes.

Therefore, after mesh evaluation, we decided to investigate both strategies of mesh refinement and make used of the public domain software that already exist (see Chapter 5).

- In terms of surface and volumetric refinement by bisection methods, since it was not possible to use the surface meshes extracted from ART volumetric meshes, we temporarily generate surface mesh per tissue using the marching cube algorithm [Lorensen and Cline, 1987] trying to get as close as possible the same triangle area as the ART triangles. We implemented the bisection mesh refinement of surface [Rivara, 1997], as shown in Figures 6.1 and 6.2. This algorithm will be extended to tetrahedral subdivision and applied first on those tetrahedral that share tissue in order to smooth and to decrease the error in the surfaces. This refinement technique can be used as well in any part of the tissues as necessary when meshes run under the numerical simulation.

- In the case of Dealunay refinement techniques we used the software called **Tetgen** (`http://tetgen.berlios.de/`). Our first approach to test the **Tetgen** software was to generate a sphere surface mesh using the marching cube method with only one material as an input (see Figure 6.3). A different example is shown in Figure 6.4 where two different materials are considered. The two concentric surface meshes were generated with the marching cube algorithm as an input for **Tetgen**. From these examples we notice that bad elements cannot be avoided.

The main objective of this exercise is to compare, for the particular case of EEG and MEG source localisation problem, which of these two geometrical models are more feasible from the geometrical point of view. At the present stage we do not have enough material to make the comparison with the MRI volumes. However from the above examples, we can view some advantages and disadvantages of both methods.

On the one hand, ART meshes need an extremely long processing times, being $\approx 45min$ for a geometrical constructor of 98,304 tetrahedra where only 26,625 are the MRI tissues of interest. The surfaces or boundaries between tissues do not match with the real surfaces. Since we will need to refine first the boundaries to make them match, this process could increase the number of tetrahedra in the boundaries or tissue interfaces, and might not be really needed for the numerical modelling process.

On the other hand, since refinement will begin with a very well-shape mesh (ART), bisection model should give us a well-shaped refined mesh with no silver or other bad shaped elements.

For the Delaunay tetrahedrisation, the first drawback that we have is the need of a well-shaped (Delaunay e.g.) surface mesh which from MRI volume is not trivial to generate. Mesh element shape can be controlled in quality and volume which it is an advantage for the numerical computations, on the contrary, apparently for such a complex structures, silver and other bad shaped elements might be unavoidable.

Another disadvantage of 3D Delaunay approach is that in order to meet the constraints needed, the algorithm inserts Steiner points into the surface mesh (see Figure 6.4(b)) which makes the mesh increases in number of elements which once more could no be really needed for a useful representation of the volume geometry and could be undesirable from the computation point of view. One really excellent advantage of **Tetgen** implementation is that the processing times are much faster ($\approx 20.34sec$ for 36,538 tetrahedron) compared with those of ART generation meshes, due to the used of exact arithmetic with the geometric predicates develop by Shewchuk [1996a].

Summary of approaches. Since ART meshes are almost regular, we think that by applying the bisection method that maximises the minimum angle we can achieve a better shape quality for the ended mesh than with Delaunay methods, since Delaunay rely first on the construction of a Delaunay surface before construct and refine the final tetrahedral mesh. The problem with the ART meshes is that since it is not based on the surface of the volume, the representation of the surfaces is not that exact. However, we can first apply the refinement only on the boundaries and interfaces between tissues and then take this output as our input for the refinement of tissue areas depending on the error estimation of the numerical model or, any other criterion for refinement.

Delaunay refinement has been applied on FEM modelling problems with very satisfactory results, although, at the present, silver tetrahedra can not be avoided. Nonetheless, with ART refinement meshes by bisection, we believe we will not have such a problem. Therefore we need first to evaluate both geometrical models in the numerical model to decide which one is preferable for the problem of MEG/EEG source localisation.

For the future work we have to accomplish the following tasks:

- Extend the Rivara's bisection algorithm in 3D, to made both the adaptation of the mesh to the boundaries of the tissues and to refine areas were more activity is expected.

- Generate Delaunay surfaces from MRI volumes in order to use them as an input for **Tetgen**. Adapt the 3 surface meshes into a single mesh indicating the difference in material (tissues) as an input for **Tetgen** to generate the refine Delaunay meshes with the 3 tissues integrated.

- Test both geometrical models in the numerical model to decide which one is preferable for the problem of MEG/EEG source localisation.

Appendix A. MRI segmentation programs at ENST.

```
This appendix provides a short description of the MRI segmentation
procedure.

DIRECTORY OF FILES: ~/martinz/PROJECT_ENST/MRI_SEGMENTATION/Petr/
DATE: August 2002.

This software was developed under the framework of the COMOBIO
project. The software has been jointly developed by P. Dokladal
and R. Urtasun [2001].

1. Contents
#####################

The directory

PROGS - contains the sources of the programmes.

bin   - contains the binaries and the links to some binaries (mostly
    situated in PROGS).

COMOBIO - contains the report, presentation slides, images,
    animations, etc.

resultats/COMOBIO_LOCAL - contains the input data and the results (only
                          for the correctly segmented following images:
              hapdey
              Lyon
              PITIE_SALPETRIERE/COU
              PITIE_SALPETRIERE/GAU
```

```
          PITIE_SALPETRIERE/IM1
          PITIE_SALPETRIERE/MAZ
          PITIE_SALPETRIERE/MRDC )


##########################
2. Program summary
##########################

Compiling the programs may be a pain, sorry for that. It's not in the
development state of a commercial application. Development of other
programs is actually still in progress by the time of writing this.

The programs are dependent of these libraries : LEDA, tivoli. Some
programs will not compile without LEDA. It has been used to
implement the waiting lists. LEDA is available for free from
http://www.mpi-sb.mpg.de/LEDA. Tivoli is internal image processing
software library of the Department of Image Processing by ENST,
Paris.

The programs will compile with the Makefile utility after adjusting
the necessary flags and environment variables.

2.1 Script programs used for the segmentation
#############################################

auto_*.sh - script programs that are used to segment the MRI's.
    These programs are called from auto_atlas.

auto_atlas.sh - the script which calls the others one by one. It
    shouldn't be run manually, since it need's special arguments.

go* - a set of script programs used to start the
    segmentation. They can be used to send the segmentation to
    another Unix-based machine. See the details lower.

2.2 Binaries (or little script utilities)
##############################################

add_border.sh - script used to add an empty border around the input
    image. The hilmi's fr3d needs it.
```

```
ascii2image - converts an ASCII file to an .ima file

caca - tree-based implementation of h-minima, permitting the
    extraction of n markers identified as the n basins of maximum dynamics
    of the mask function.

cc_tree - segmentation program performing the \lambda-thinning,
    \lambda-thickening or \lambda-skeletonization. Extracts a
    simple connected components by iterative simple points
    manipulations.

ima2pgm,
pgm2ima,
eps2jpeg.sh - data format converter scripts

fuse_cervelet - utilitary program used in auto_tc_cerv.sh. Not to be
    called manually.

geodist - geodesic (mask conditioned) distance calculation.

get_marker,
get_params.sh,
order_labeled_objets,
t_getClusters,
t_robust - utilitary programs. Not to be called manually.

lpe_reconstr - reconstruction based on the watershed. Takes a set of
    markers and a mask image. IMPLEMENTATION UNFINISHED: Doesn't
    use the distance on the plateau.

make_pub.sh,
make_view.sh - utilitary scripts used to create 3D sections with loox


###############################################
3. How to run the segmentation on one image
###############################################

Before running the segmentation, you must manually prepare one
argument files called "ATLAS_PARAMS". See the auto_atlas.sh for
```

details.

While the segmentation is supposed entirely automated, it is not 100% liable. The user must stop the execution of the script at two places to check the intermediate results. This is done by placing the "exit" command on the right place in the auto_atlas.sh script. If earlier stages of the auto_atlas.sh script are not to be re-executed, they must be commented out manually. We have preferred some ergonomic inconvenience but SIMPLICITY at this stage of development to excessive complexity of the scripts.

The auto_atlas.sh script is run in this way:

1) create a directory to store the input data, argument files, output of the scripts and the intermediate results. Go to this directory.

2) create the control file ATLAS_PARAMS (see details in auto_atlas.sh)

3) run "go machine_name" where "machine_name" is a name of the Unix machine where it will be executed. (.rhost file must be set on the root directory to enable the remote shell execution - see some Unix manual). Otherwise run: "auto_segment.sh ‘cat ATLAS_PARAMS‘" (inverse quotes)

4) the first execution will attempt to generate automatically the "arguments" file containing the segmentation parameters. The arguments file contains ten ASCII numbers. These are the means and std deviations for five classes representing the air+bone, cerebrospinal fluid, grey matter, white matter and fat in the input image. Please
check manually that these correspond roughly. In 40% of cases the csf is not found. If these are not correct, you must proceed manually to generate them.

5) Place an "exit" command after the "Segmentation du cerveau" section and run the segmentation for the second time. As soon as it stops, check the *_brain_mask file. The object should delimit correctly the encephalon. If it remains connected to the rest of the head, lower the mean value or the std deviation (third and fourth number if the arguments file) of the cerebrospinal fluid. If it is incomplete then increase the mean (ninth number in the

```
arguments file) of the fat.

6) Now, place the "exit" command after the "Obtention du tronc
cerebral et du cervelet" section. You may comment out the
extraction of the encephalon to prevent it to be recalculated.
When it finishes, check the *_tc and *_cervelet files. They should
contain the brain stem and the cerebellum, respectively.
Otherwise, you must modify either the manually-chose position of
the slice delimiting the upper end of the brain stem or the
cortex-intensity mean value.

7) Comment out the previous sections and continue the
segmention. Chances are that everything works fine now.

8) The intermediate results are stored in the current directory
(supposed this directory is mounted at the same place).
The output files are as follow:

Section: ``Segmentation des hemispheres'' *_hemisph_mask, *_hemisph
Section: ``Calcul de l'appartenance au cortex'' *_cortex
Section: ``Obtention du contour de la tete'' *_head_mask_lisse,
         *_head_mask_lisse_inv, *_head_mask, *_head_mask_inv
Section: ``Obtention de la peau'' *_peau
Section: ``Obtention de la graisse'' *_graisse
Section: ``Obtention du crane'' *_IrmEntree, *_crane, *_crane_focalise,
         *_crane_focus_limit
Section: ``Obtention des muscles'' *_muscles
Section: ``Obtention de l'image de contours et d'etiquettes a fin d'evaluer''
         *_labels, *_contours

##############################################
3. How to run the segmentation on all images
##############################################

You can run the segmentation of several images at a time. This is
useful for the development of new methods that must tested on a
serie of images. Use the go_all.sh script to create and send the
processes on other machines. This script doesn't take parameters,
it must be modified by the user. To kill the dispatched
processes, use kill_all.sh (must also be modified to use correct
names). Otherwise, use go_signac_go.sh to have signac do all the
```

```
images one by one.

###############################################

To contact the author: Peter.DOKLADAL@cmm.ensmp.fr
```

Appendix B. ART mesh generation at ENST.

Pescatore [2001] developed a library called TIC where we can find all the programs used to generate the ART meshes.

B-1 Sequence to generate the ART meshes

There are two main steps, 1) generate the geometrical constructor and 2) obtain the surface and volume ART homotopic labelled meshes.

1. GEOMETRICAL CONSTRUCTOR:

Program: `TicFuchsTetra.cpp`

Directory: ~martinz/PROJECT_ENST/PROGS/Tic/bin

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Input parameters for `TicFuchsTetra`:

```
TicFuchsTetra -h

-l  <lambda>                default = 1/3
-c  <condition> (0 to 3)    default = 0
-t  <type> of connectivity
-n  <number> of tetrahedra  default = 100,000
-o  <output> the mesh of the geometrical constructor
```

Example: `TicFuchsTetra -t 24 -o mesh_n4`
Output files: `mesh_n4.tetra, mesh_n4.ntetra`

2. ART HOMOTOPIC LABELLED MESHES:

Program: `TicFuchsMixel.cpp`

Directory: ~martinz/PROJECT_ENST/PROGS/Tic/bin

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Input parameters for `TicFuchsMixel`:

```
TicFuchsMixel -h

-i  <input image>    (format .ima)
-d  <debugger>       It generates de evolution files of the process
                     Debug files can be converted into .ima with TicTetra2Gis
-t  <topology>       Use for topology of sphere
-m  <mesh>           Geometrical constructor file
-o  <output>         Name of the output mesh file
-it <iterations>     For smooth, 0 = no smoothing or number of iterations
                     for Laplacian smooth. Default = 0
-f  <type of model>  When it is used it takes the central tetrahedron to begin
                     the homotopic labelling. If it is not use it takes other
                     model which is not working properly.
-t1 <threshold 1>    Tissue proportion per tetrahedron (optimum 0.8)
-t2 <threshold 2>    Tissue proportion per neighbour by face (optimum 0.6)
-c  <case>           0 = Non homotopic labelling, only tissue proportion
                     1 = Homotopic labelling, only for the maximum volume tissue
                     2 = Homotopic labelling for all tissues
-s  <scale>          In mm. Scale factor to expand the geometrical constructor
                     to the size of the volume image.
```

Example: `TicFuchsMixel -i MRI_segmented -t -m mesh_n4 -o tissue_s160 -f -t1 0.8 -t2 0.6 -c 2 -s 160`

Output files: `tissue_s160.mtetra`, the surface meshes: `tissue_s160_1.tri`, `tissue_s160_2.tri`, `tissue_s160_3.tri`.

If the option -it is used then: `tissue_s160_smooth_1.tri`, `tissue_s160_smooth_2.tri`, `tissue_s160_smooth_3.tri`

B-2 Mesh file formats

```
Volume image: GIS format

2 files: *.dim -> header and  *.ima -> binary file

*.dim contains:

Nx Ny Nz (image dimension)
-type    (S16: signed short, U16: unsigned short, U8: unsigned char)
-dx      (voxel size in (mm) dx,dy,dz)
-dy
-dz

For example:
256 256 106
s16
0.937
0.937
1.500

Surface mesh format TRI: *.tri

- number of vertex
x y z     coordinates of vertex follow by
nx ny nz  normal coordinates
- number of triangles vertex of triangles

For example:
- 21891
123.496 123.211 4.75914 -0.108704 0.604082 -0.789473
125.105 123.589 4.86635 -0.224496 0.542998 -0.809169
123.727 124.042 5.54772 -0.116728 0.571328 -0.812379
 - 43778 43778 43778
0 9395 9501
0 9501 2
0 2 1
0 1 9396
```

```
Tetrahedral mesh: 3 files: *.ntetra *.tetra *.mtetra

File *.tetra

- number vertex
x y z coordinates

- number of tetrahedron
list of vertices of the tetrahedron (th tetrahedra are oriented with the normals
at the exterior of the faces)

For example:
- 18128
-2 -1.41421 -0.816497
-1 -1.41421 -0.816497
1.11022e-16 -1.41421 -0.816497
1 -1.41421 -0.816497
-1.66667 -0.471405 -0.816497
-0.666667 -0.471405 -0.816497
0.333333 -0.471405 -0.816497
1.33333 -0.471405 -0.816497
-1.33333 0.471405 -0.816497
-0.333333 0.471405 -0.816497
0.666667 0.471405 -0.816497
1.66667 0.471405 -0.81

- 98304 98304 98304
21 2608 2610 2611
2608 512 2609 2612
2610 2609 514 2613
2611 2612 2613 515

File *.ntetra : the neighbours by vertex in the tetrahedral mesh

- number of tetrahedron
size of the list of neighbours, neighbour\_1 ... neighbour\_n

For example:
- 24
23 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
23 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

```
23 0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19......

File *.mtetra: label (or material) of each tetrahedron

format: - number of labels label1 ... labeln label composition

For example:
 - 4
 0 50 100 255    (0=backfround, 50=brain, 100=skull, 255=skin)
 0 100 0 0
 0 100 0 0
 0 100 0 0
 0 100 0 0
```

Appendix C. Evaluation and miscellaneous programs.

C-1 Surface mesh evaluation

PROGRAM FOR DISTANCE: `evalMeshDist.c`

Directory: ~martinz/PROJECT_ENST/PROGS/Evaluation/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `brain_bin.ima, MCbrain.tri`

HELP:

```
18) martinz@kahlo % ~martinz/PROJECT_ENST/PROGS/Evaluation/evalMeshDist -h
evalMeshDist: makes the evaluation of the distance between the original binary
surface of the volume (.ima) and the vertices of its surface mesh (.tri)
using Chamfer distances. Inputs are the original binary volume image
(.ima, only U8BIT) and the surface mesh file (.tri). The output is a ordered ASCII
vector with the inside or outside distance of each vertex to the surface (.dat).

evalMeshDist: input argument missing
Usage: evalMeshDist -i|nput| <image (Tivoli)>
                   [-m|esh input mesh TRI format|]
                   [-o|utput| <array with differences (ASCII)>]
                   [-h|elp|]
```

EXAMPLE:

```
15) martinz@kahlo % evalMeshDist -i brain_bin -m MCbrain -o diff

reading input image achieved
distance (from 962916 voxels) completed
```

```
Inverting...
distance (from 5983900 voxels) completed
writing the result achieved
Open mesh file: MCbrain.tri
Num vertex: 4792, Num triangles: 9580
Open mesh file: diff.dat
Median= 1.000 ; Mean= 1.275 ; std= 0.478; Maxima= 7
the end (13.0s)
```

Output files names: `diff.dat`, `distance_map.ima`

```
16) martinz@kahlo % ls
MCbrain.tri         brain_bin.ima      distance_map.dim
brain_bin.dim       diff.dat           distance_map.ima
```

PROGRAM FOR GEOMETRY: `evalMeshGeom.c`

Directory: ~martinz/PROJECT_ENST/PROGS/Evaluation/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `MCbrain.tri`

EXAMPLE:

```
49) martinz@kahlo % evalMeshGeom MCbrain.tri

Geometric feature analysis:

         SHAPE QUALITY
         Minimum    : 0.246002 (triangle #2099)
         Mean       : 0.790570
         Stan. dev. : 0.115592

         PLANARITY
         Minimum    : 0.908845 (triangle #4757)
         Mean       : 0.985701
         Stan. dev. : 0.012872

         SMOOTHNESS
         Minimum    : 0.933956 (triangle #2951)
         Mean       : 0.993996
         Stan. dev. : 0.006319

         DEVIATION
         Minimum    : 0.649510 (triangle #4762)
         Mean       : 0.893026
         Stan. dev. : 0.051207

THE END.
```

C-2 Volume mesh evaluation

PROGRAM VOLUME MESH: `evalTetMesh.cpp`

Directory: ~martinz/PROJECT_ENST/PROGS/Evaluation/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `brain180.tetra`

EXAMPLE:

```
109) martinz@kahlo % evalTetMesh -m brain180.tetra

Evaluating TetMesh file: brain180.tetra
18128 vertices, 27496 triangles,  6874 tetrahedra

Mesh quality statistics:

  Smallest volume:                97.421    |  Largest volume:            97.423
  Smallest shape ratio:           1.1793    |  Largest shape ratio:       1.4317
  Smallest area:                  39.553    |  Largest area:              53.693
  Shortest edge:                  8.2216    |  Longest edge:              12.323

  Smallest dihedral:         47.254    |  Largest dihedral:         106.32

  Dihedral Angle histogram:
        0 -  10 degrees:         0     |     90 - 100 degrees:      2290
       10 -  20 degrees:         0     |    100 - 110 degrees:      4589
       20 -  30 degrees:         0     |    110 - 120 degrees:         0
       30 -  40 degrees:         0     |    120 - 130 degrees:         0
       40 -  50 degrees:      9178     |    130 - 140 degrees:         0
       50 -  60 degrees:      4570     |    140 - 150 degrees:         0
```

```
      60 -  70 degrees:      9178    |    150 - 160 degrees:        0
      70 -  80 degrees:      4570    |    160 - 170 degrees:        0
      80 -  90 degrees:      6869    |    170 - 180 degrees:        0

  Shape quality histogram:
     0.50 - 1.50 :      6874    |    8.50 - 9.50 :            0
     1.50 - 2.50 :         0    |    9.50 - 10.50 :           0
     2.50 - 3.50 :         0    |    10.50 - 11.50 :          0
     3.50 - 4.50 :         0    |    11.50 - 12.50 :          0
     4.50 - 5.50 :         0    |    12.50 - 13.50 :          0
     5.50 - 6.50 :         0    |    13.50 - 14.50 :          0
     6.50 - 7.50 :         0    |    14.50 - 15.50 :          0
     7.50 - 8.50 :         0    |    15.50 - 16.50 :          0

  Shape histogram:

    Silver:  0
    Needle:  0
    Spindle: 0
    Wedge:   0
    Cap:     0

  There are 0 bad elements among 6874 elements.
```

C-3 Miscellaneous

To display the meshes:

The file meshes are display using Geomview GNU/GPL, which was written at the Geometry Center at the University of Minnesota between 1992 and 1996 (`http://www.geomview.org/`).

PROGRAM TO DISPLAY MESH: `geomview`

Directory: ~martinz/PROJECT_ENST/PROGS/Geomview/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `dodec.off, sphere.off`

In order to display the meshes with geomview, the files have to be converted to the *.OFF format:

```
#"dodec.off":
OFF
20 12 30
        1.214124 0.000000 1.589309
        0.375185 1.154701 1.589309
        -0.982247 0.713644 1.589309
        -0.982247 -0.713644 1.589309
        0.375185 -1.154701 1.589309
        1.964494 0.000000 0.375185
        0.607062 1.868345 0.375185
        -1.589309 1.154701 0.375185
        -1.589309 -1.154701 0.375185
        0.607062 -1.868345 0.375185
        1.589309 1.154701 -0.375185
        -0.607062 1.868345 -0.375185
        -1.964494 0.000000 -0.375185
        -0.607062 -1.868345 -0.375185
        1.589309 -1.154701 -0.375185
        0.982247 0.713644 -1.589309
        -0.375185 1.154701 -1.589309
        -1.214124 0.000000 -1.589309
        -0.375185 -1.154701 -1.589309
        0.982247 -0.713644 -1.589309
        5 0 1 2 3 4
        5 0 5 10 6 1
        5 1 6 11 7 2
        5 2 7 12 8 3
        5 3 8 13 9 4
        5 4 9 14 5 0
        5 15 10 5 14 19
        5 16 11 6 10 15
        5 17 12 7 11 16
        5 18 13 8 12 17
        5 19 14 9 13 18
        5 19 18 17 16 15
```

The "OFF" header tells us it's a polylist file. The second line in the file tells us that there are 20 vertices, 12 faces, and 30 edges. (The OOGL libraries presently don't use the edges value, so you can just use 0 if you don't happen know the number of edges.) The next 20 lines give a list of vertices. The last 12 lines specify the faces: the first number is the number of vertices in that face. Since our polyhedron happens to be regular, all faces have the same number of vertices (in this case, 5). The rest of the numbers on the line are indices into the above list of vertices. After these numbers a RGB colour (1 0 0 = red) can be added as well as shown in the next example:

```
# This file represent an sphere with triangle elements.
# ‘‘sphere.off’’
OFF
18 32 0
 56.24  0.000  0.000
 0.000  56.24  0.000
-56.24  0.000  0.000
 0.000 -56.24  0.000
 0.000  0.000  56.24
 0.000  0.000 -56.24
 0.000  39.77  39.77
 39.77  39.77  0.000
 39.77  0.000  39.77
-39.77  0.000  39.77
-39.77  39.77  0.000
 0.000 -39.77  39.77
-39.77 -39.77  0.000
 39.77 -39.77  0.000
 0.000  39.77 -39.77
 39.77  0.000 -39.77
-39.77  0.000 -39.77
 0.000 -39.77 -39.77
3 8 6 7 1.00 1.00 0.00
3 6 9 10 1.00 0.00 0.00
3 9 11 12 1.00 0.00 0.00
3 11 8 13 1.00 0.00 0.00
3 7 14 15 1.00 0.00 0.00
3 10 16 14 1.00 0.00 0.00
3 12 17 16 1.00 0.00 0.00
3 13 15 17 1.00 0.00 0.00
3 0 8 7 1.00 0.00 0.00
3 4 6 8 1.00 0.00 0.00
3 1 7 6 1.00 0.00 0.00
3 1 6 10 1.00 0.00 0.00
3 4 9 6 1.00 0.00 0.00
3 2 10 9 1.00 0.00 0.00
3 2 9 12 1.00 0.00 0.00
3 4 11 9 1.00 0.00 0.00
3 3 12 11 1.00 0.00 0.00
```

```
3 3 11 13 1.00 0.00 0.00
3 4 8 11 1.00 0.00 0.00
3 0 13 8 1.00 0.00 0.00
3 0 7 15 1.00 0.00 0.00
3 1 14 7 1.00 0.00 0.00
3 5 15 14 1.00 0.00 0.00
3 1 10 14 1.00 0.00 0.00
3 2 16 10 1.00 0.00 0.00
3 5 14 16 1.00 0.00 0.00
3 2 12 16 1.00 0.00 0.00
3 3 17 12 1.00 0.00 0.00
3 5 16 17 1.00 0.00 0.00
3 3 13 17 1.00 0.00 0.00
3 0 15 13 1.00 0.00 0.00
3 5 17 15 1.00 0.00 0.00
```

To execute Geomview you only have to type as follow:

`geomview sphere.off`

The next section will present some miscellaneous software mainly to convert formats between the ones use by the TIC library (see Appendix B for detail about these formats) and the format OFF.

To convert formats:

PROGRAM: `J2tetra.cpp`

Description: Extract from files *.mtetra and *.tetra (from the TIC library) the mesh of the selected tissue (only one tissue each time).

Directory: ~martinz/PROJECT_ENST/PROGS/Utils/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `tissue180.mtetra, tissue180.tetra`

Use: J2tetra file.tetra file.mtetra level

where file.tetra=geometry constructor, file.mtetra=labels per tetrahedron
and level =1 for brain, =2 for skull and =3 for head

EXAMPLE:

```
J2Tetra tissue180.tetra tissue180.mtetra 2
writing data.tetra
number of vertex: 18128, number of tetrahedra: 3487
```

The `data.tetra` file corresponds only to the tetrahedral mesh of the skull.

PROGRAM: `Tri2Off.c`

Description: Converts the TRI format of the TIC library into OFF files to use for display.

Directory: ~martinz/PROJECT_ENST/PROGS/Utils/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `MCbrain.tri`

Use: Tri2Off -m file.tri

where -m <mesh> file.tri, the output will have the same and name extension *.off

EXAMPLE:

```
Tri2Off -m MCbrain
Open mesh file: MCbrain.tri
Num vertex: 4792, Num triangles: 9580
Writing MCbrain.off
the end (0.4s)
```

PROGRAM: `Tetra2Off.c`

Description: Converts the TETRA format of the TIC library into OFF files to use for display.

Directory: ~martinz/PROJECT_ENST/PROGS/Utils/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `brain180.tetra`

Use: Tetra2Off -m file.tetra

where -m <mesh> file.tetra, the output will have the same name and extension *.off

EXAMPLE:

```
Tetra2Off -m brain180.tetra
Num vertex: 18128, Num triangles: 27496
Writing brain180.tetra.off
the end (1.2s)
```

PROGRAM: `Tri2Mesh.c`

Description: Converts the surface TRI format of the TIC library into SMESH surface files to use as an input for Tetgen (see Appendix E)

Directory: ~martinz/PROJECT_ENST/PROGS/Utils/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `MCbrain.tri`

Use: Tri2Mesh -m file.tri

where -m <mesh> file.tri, the output will have the same name and extension *.smesh

EXAMPLE:

```
Tri2Mesh -m MCbrain
Open mesh file: MCbrain.tri
Num vertex: 4792, Num triangles: 9580
Writing MCbrain.smesh
the end (0.5s)
```

Appendix D. Bisect mesh refinement programs.

As we mentioned in Chapter 6, the bisection algorithm of Rivara and Levin [1992] is at the present implemented only for 2D version, although it is applied to 3D mesh surfaces, the bisection is made in the triangle elements of these surfaces the same as the algorithm is designed in 2D. These scripts are implemented using Matlab, future versions including 3D bisection will be implemented in C.

Since the graphical interface (GUI) of Matlab 6.0 makes the process very slow and we do not need it now, we recommend that you run Matlab without the GUI:

```
matlab -nojvm -nosplash
```

PROGRAM 2DMESH BISECTION: `refine_mesh.m`

Directory: ~martinz/PROJECT_ENST/PROGS/Rivara2D/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `sphere.tri, brain_sm.tri`

EXAMPLES:

Figure 6.1(a) corresponds to the file sphere.tri, triangle number 1 is marked in light color only for visual purpose.

Figure 6.1(b) is generated as follows:

```
ref_trig=refine_mesh('sphere.tri',1);

Riding file sphere.tri ...

18 vertices, 32 triangles
MAIN: marking  the mesh....done.  [Nv=18,Nt=32]  [time=0]
MAIN: refining the mesh....done.  [Nv=19,Nt=34]  [time=0.06]
Writing file sphere1.off
Writing file sphere1.tri
```

The program refine_mesh has two input parameters, the TRI mesh file and a vector with the indices of the triangles to be refined, in this case only number 1. The output is the mesh already refined in formats TRI and OFF with the same name and an increase counter, another output is the vector of the indices of the triangles refined in this run. This is with the purpose to run iteratively the same program as many times as wanted, and to continue the refinement in the same area if it is needed. Nv = number of vertices and Nt = number of triangles.

Figure 6.1(c) is generated as follow:

```
ref_trig2=refine_mesh('sphere1.tri',ref_trig);

Riding file sphere1.tri ...
19 vertices, 34 triangles
MAIN: marking  the mesh....done.  [Nv=19,Nt=34]  [time=0]
MAIN: refining the mesh....done.  [Nv=26,Nt=48]  [time=0.44]
Writing file sphere2.off
Writing file sphere2.tri
```

In this example we refined again the triangles refined in the last run, which indices are save in ref_trig, instead of only the number one.

Finally, and following the same idea, Figure 6.2(a) corresponds to the file brain_sm.tri. Figure 6.2(b) is generated by:

```
rt=refine_mesh('brain_sm.tri',1);

Riding file brain_sm.tri ...
2693 vertices, 998 triangles MAIN:
marking  the mesh....done.  [Nv=2693,Nt=998]  [time=0.33] MAIN:
refining the mesh....done.  [Nv=2697,Nt=1006]  [time=1.09]
Writing file brain_sm1.off
Writing file brain_sm1.tri
```

And Figure 6.2(c) is generated by:

```
rt2=refine_mesh('brain_sm1.tri',rt);

Riding file brain_sm1.tri ...
2697 vertices, 1006 triangles MAIN:
marking  the mesh....done.  [Nv=2697,Nt=1006]  [time=0.33] MAIN:
refining the mesh....done.  [Nv=2707,Nt=1026]  [time=1.75]
Writing file brain_sm2.off
Writing file brain_sm2.tri
```

At the moment in order to refined the mesh, the triangles to be refined have to be marked by hand (through the vector of indices). This decision should be made depending on the needs of the simulation problem, for example, by an error tolerance or other criteria.

Appendix E. TETGEN Constrained Delaunay mesh generation.

Tetgen is a 3D quality tetrahedral mesh generator and Delaunay triangulator. It is based on the Delaunay method. The goal of Tetgen is to generate tetrahedral meshes for arbitrary three-dimensional domains that are adapted to various problems of scientific computing, e.g. Computational Fluid Dynamics, Computational Structural Mechanics, Computational Electro-Magnetic, Thermal problems and so on.

It is mainly based on the PhD thesis of Jonathan Shewchuk [1997] and on the software develop by the same person called TRIANGLE which generates 2D meshes (http://www-2.cs.cmu.edu/∼quake/triangle.html). Tetgen follows the same design philosophy as TRIANGLE.

Tetgen can be used as stand alone program, or as C++ library to incorporate into another program. It is available in free GNU source code status, and it can run on any computer with a C++ compiler, e.g. cc/gcc under Unix/Linux and bcc32/msvc under Windows98/NT/2000.

Tetgen can be obtained via http from: <http://tetgen.berlios.de> Author's address: <sihang@mail.berlios.de>

Tetgen generates exact Delaunay tetrahedrisations, constrained (conforming) Delaunay tetrahedrisations, and quality conforming Delaunay tetrehedrisations. The latter only generates an almost good mesh with bounded radius-edge ratio. If no command line switches are specified, your .node input file will be read, and the Delaunay tetrahedrisation will be returned in .node and .ele output files (for details on file formats used by Tetgen look at <http://tetgen.berlios.de/>, or at ∼martinz/PROJECT_ENST/PROGS/Tetgen/UserManual.pdf). In this appendix we will describe only the format we use in our examples (*.smesh).

The command syntax is:

```
tetgen [-pq__a__AfcngBPNEIOXzS_T__CQVh] input_file

Underscores indicate that numbers may optionally follow certain switches;
do not leave any space between a switch and its numeric parameter.
input_file must be a file with extension .node, or extension .poly if the
```

```
-p switch is used. The formats of these files are described below.

Command Line Switches:

    -p  Reads a Piecewise Linear Complex (.poly file), which can specify
        points, facets, holes, and regional attributes and volume
        constrains.  Will generate a constrained (conforming) Delaunay
        tetrahedrisation fitting the input; or, if -q, or -a is used, a
        conforming Delaunay tetrahedrisation. If -p is not used, Tetgen
        reads a .node file by default.
    -q  Quality mesh generation by Jonathan Shewchuk's[1997] Delaunay
        refinement algorithm. Adds points to the mesh to ensure that all
        tetrahedra are bounded in specified radius-edge ratio. The default
        radius-edge ratio used is 2, Shewchuk proved the refinement algorithm
        is theoretically guarantee to terminate (assuming no small input
        angle in the PLC). An alternative minimum radius-edge ratio may be
        specified after the 'q'. In practice, the algorithm often succeeds
        for minimum radius-edge ratio to 1.1.
    -a  Imposes a maximum tetrahedra volume. If a number follows the ‘a’, no
        tetrahedron will be generated whose volume is larger than that number.
        If no number is specified, a .poly file specifies a number of maximum
        volume constraint. A .poly file can optionally contain a volume
        constraint for each facet-bounded region, thereby enforcing tetrahedron
        densities in a first triangulation.  You can impose both a fixed volume
        constraint and a varying volume constraint by invoking the -a switch
        twice, once with and once without a number following.  Each volume
        specified may include a decimal point.
    -A  Assigns an additional attribute to each tetrahedron that identifies
        what face-bounded region each tetrahedron belongs to.  Attributes
        are assigned to regions by the .poly file.  If a region is not
        explicitly marked by the .poly file, tetrahedra in that region are
        assigned an attribute of zero.  The -A switch has an effect only
        when the -p switch is used.
    -f  Outputs (to a .face file) a list of faces of the tetrahedrisation.
    -c  Outputs (to a .face file) a list of convex hull faces of the point
        set. Note: if you use -f and -c switches together, Tetgen only
        generates a convex hull faces file for you.
    -n  Outputs (to a .neigh file) a list of tetrahedra neighboring each
        tetrahedron.
    -g  Outputs the volume mesh to (.ele.gid) file and surface mesh to
        (.face.gid) file for viewing mesh result with Gid software.
    Outputs an Object File Format (.off) file, suitable for
    viewing with the Geometry Center's Geomview package.
-B  No boundary markers in the output .node, .poly, and .face output
    files.
-P  No output .poly file. Saves disk space.
-N  No output .node file. Saves disk space.
-E  No output .ele file. Saves disk space.
-I  No iteration numbers. Suppresses the output of .node and .poly
    files, so your input files won't be overwritten.  (If your input is
    a .poly file only, a .node file will be written.) Shouldn't be used
    with the -q, or -a switch if you are using a .node file for input,
    because no .node file will be written, so there will be no record
    of any added points.
-O  No holes.  Ignores the holes in the .poly file.
-X  No exact arithmetic.  Normally, Tetgen uses approximate floating-point
    arithmetic for certain tests. In some circumstances the inexact tests
    are not accurate enough. Exact arithmetic may increase the robustness
    of the algorithms, despite floating-point round off error. Use exact
    arithmetic will cause a small lower in speed but create the possibility
    valid mesh when approximate arithmetic failed. The exact arithmetic
    package used is written by Jonathan Shewchuk [1996].
-z  Numbers all items starting from zero (rather than one).  Note that
    this switch is normally overridden by the value used to number the
    first point of the input .node or .poly file.  However, this switch
    is useful when calling Tetgen from another program.
-S  Specifies the maximum number of Steiner points (points that are not
    in the input, but are added to meet the constrains of minimum
    radius-edge ratio and maximum volume). The default is to allow an
    unlimited number. If you specify this switch with no number after it,
    the limit is set to zero.  Tetgen always adds points at segment
    intersections, even if it needs to use more points than the limit
    you set. Tetgen by default inserts segments by splitting, it
    always adds enough points to ensure that all the segments appear in
    the tetrahedrisation, again ignoring the limit.  Be forewarned that
    the -S switch may result in a conforming tetrahedrisation that is not
    truly Delaunay, because Tetgen may be forced to stop adding
    points when the mesh is in a state where a segment or subface is
    non-Delaunay and needs to be split. If so, Tetgen will print a warning.
-T  Specifies the tolerance for round-to-zero. Default Tetgen use a
    tolerance 1e-12.
-C  Check the consistency of the final mesh.  Uses exact arithmetic for
```

```
    checking, even if the -X switch is used.  Useful if you suspect
    Tetgen is buggy.
-Q  Quiet: Suppresses all explanation of what Tetgen is doing, unless
    an error occurs.
-V  Verbose: Gives detailed information about what Tetgen is doing.
    Add more ‘V’s for increasing amount of detail.  ‘-V’ gives
    information on algorithmic progress and more detailed statistics.
    ‘-VV’ gives point-by-point details, and will print so much that
    Triangle will run much more slowly.  ‘-VVV’ gives information only
    a debugger could love.
-h  Help:  Displays a detail descriptions on Command Line Switches.
```

For the particular case of generating volume meshes of the head tissues, we need to generate first the surface meshes of the volumes before running Tetgen.

Once this surfaces are generated, they need to be in .smesh format in order to be use as an input for Tetgen. We will describe briefly the .smesh format and show the examples of how to use Tetgen in order to obtain the volume meshes shown in Chapter 6 (see Appendix C to find a program to change surface .TRI format to .SMESH format).

.smesh file format

```
First line: <# of points> <dimension (must be 3)> <# of attributes>
<# of boundary markers (0 or 1)>

Following lines: <point #> <x> <y> <z> [attributes] [boundary marker]

One line: <# of facets> <# of boundary markers (0 or 1)>
Following lines: <# of facet’s vertices> <endpoint1> <endpoint2>...
[boundary marker]

One line: <# of holes>
Following lines: <hole #> <x> <y> <z>

Optional line: <# of regional attributes and/or volume constraints>
Optional following lines: <constraint #> <x> <y> <z> <attrib> <max volume>
```

EXAMPLE: Sample .smesh file describing the surface of a cylinder.

```
# cylinder.smesh file

# Nodes section
# 640 points in 3D, no attributes, no boundary marker.
640 3 0 0
1 109.833312988281 -16.779541015625 0
2 105.925262451172 15.4060974121094 0
3 94.4282531738281 45.7212219238281 0
4 76.0104370117187 72.4040374755859 0
5 51.7422180175781 93.90380859375 0 ...

# Facets section
# 1204 facets, no boundary markers
1204 0
3 495 584 520
3 561 543 554
3 2 27 1
3 28 27 2
3 29 2 3
3 29 28 2
3 4 29 3
3 30 29 4
3 31 4 5
3 31 30 4 ...

# Holes section
# There is no hole in cylinder 0

# Region section
# No region be defined 0
```

EXAMPLE: To generate the volume mesh if this cylinder using Tetgen.

```
tetgen -p cylinder.smesh
```

Reads the surface mesh file from cylinder.smesh file, compute constrained (conforming) Delaunay tetrahedralisation of this surface mesh and write results to object.1.node and object.1.ele. To generate the output to be able to display with Geomview then:

`tetgen -pg cylinder.smesh`.

How to generate the examples shown in Figures 6.3 and 6.4?

PROGRAM: `tetgen.cpp`

Directory: ~martinz/PROJECT_ENST/PROGS/Tetgen/bin/

Sample files directory: ~martinz/PROJECT_ENST/EXAMPLES/

Sample files names: `one_sphere.smesh, two_.smesh`

In Figure 6.3 we use only one sphere surface (only one material or tissue), whereas in Figure 6.4 we use two concentric surface spheres (two different material or tissues).

Figure 6.3(a) shows Delaunay tetrahedral mesh generated as:

```
tetgen -qpgVC one_sphere.smesh
```

where `one_sphere.smesh` is the surface mesh of a sphere, `-q` generates a mesh with the default value of radius-edge ratio (2), `-p` to read instead the .poly file the .smesh file, and generates a conforming Delaunay mesh (using together with -q), `-g` generates the output file .off to display with Geomview and a .gid file to display with Gid, `-V` verbose and `-C` checks for consistency. It generates the following output files:

```
one_sphere.1.node
one_sphere.1.ele
one_sphere.1.ele.gid
one_sphere.1.face.gid
one_sphere.1.off
```

Figure 6.3(b) shows a constrain Delaunay tetrahedral mesh generated as:

```
tetgen -q1.14a3000gVC one_sphere.1.node
```

where `one_sphere.1.node` is the .node file generated from the last example, `-q` generates a mesh with a value of radius-edge ratio of 1.14, `-a` imposes a maximum tetrahedron volume of 3000, `-g` generates the output file .off to display with Geomview and a .gid file to display with Gid, `-V` verbose and `-C` checks for consistency. It generates the following output files:

```
one_sphere.2.node
one_sphere.2.ele
one_sphere.2.ele.gid
one_sphere.2.face.gid
one_sphere.2.off
```

Figure 6.4(a) shows Delaunay tetrahedral mesh generated as:

```
tetgen -gVC two_sphere.smesh
```

where `two_sphere.smesh` is the surface meshes of two concentric spheres, it generates a Delaunay mesh for both material using the default values, `-g` generates the output file .off to display with Geomview and a .gid file to display with Gid, `-V` verbose and `-C` checks for consistency. It generates the following output files:

```
two_sphere.1.node
two_sphere.1.ele
two_sphere.1.ele.gid
two_sphere.1.face.gid
two_sphere.1.off
```

Figure 6.4(b) shows a constrain Delaunay tetrahedral mesh generated as:

```
tetgen -pq1.8a100A1a100A0XgVC two_sphere.smesh
```

where `two_sphere.smesh` is the surface mesh of two concentric spheres, `-q` generates a mesh with a value of radius-edge ratio of 1.18 for both materials, `-a` imposes a maximum tetrahedron volume of 100, `A1` for the attribute 1 or material 1, `-a` imposes a maximum tetrahedron volume of 100, `A0` for the attribute 0 or material 0, `X` use exact arithmetic, `-g` generates the output file .off to display with Geomview and a .gid file to display with Gid, `-V` verbose and `-C` checks for consistency. It generates the following output files:

```
two_sphere.1.node
two_sphere.1.ele
two_sphere.1.ele.gid
two_sphere.1.face.gid
two_sphere.1.off
```

For any other application of `Tetgen`, please consult the user manual at:
`~martinz/PROJECT_ENST/PROGS/Tetgen/UserManual.pdf`

Bibliography

Amenta, N., Choi, S., and Kolluri, R. (2001). The power crust. In *ACM Symposium on Solid Modeling and Applications*, Ann Arbor.

Bank, R. and Sherman, A. (1979). The use of adaptive grid refinement for badly behaved elliptic partial differential equations. In R. Vichnevetsky and R. Stepleman, editors, *Advances in computer Methods for Partial Differential Equations*, volume 3, pages 33–39, IMACS, New Brunswick.

Bänsch, E. (1991). Local mesh refinement in 2 and 3 dimensions. *Impact Comput. Sci. Engin.*, **3**, 181–191.

Bey, J. (1995). Tetrahedral grid refinement. *Computing*, **55**, 355–378.

Borgefors, G. (1984). Distance transformation in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, **27**, 121–145.

Borgefors, G. (1986). Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, **34**, 344–371.

Carey, G. (1976). A mesh refinement scheme for finite element computations. *Comput. Methods Appl. Mech. Engin.*, **7**(1), 93–105.

Cuffin, B. (1996). EEG localization accuracy improvements using realistically shaped head models. *IEEE Trans. on Biomedical Engineering*, **43**, 229–303.

De Cougny, H. and Shephard, M. (1999). Parallel refinement and coarsening of tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, **46**, 1101–1125.

Dokladal, P., Urtasun, R., Bloch, I., and Garnero, L. (2001). Segmentation of 3D Head MR Images using Morphological Reconstruction under Constraints and Automatic Selection of Markers. In *IEEE International Conference on Image Processing, ICIP 2001*, volume III, pages 1075–1078, Thessalonique, Greece.

Dompierre, J., Labbé, P., Guibault, F., and Camarero, R. (1998). Proposal of benchmarks for 3d unstructured tetrahedral mesh optimization. In *7th International Meshing Roundtable*, Dearborn, Michigan.

Frey, J. and Borouchaki, H. (1998). Geometric evaluation of finite element surface meshes. *Finite Elements in Analysis and Design*, **31**, 33–53.

Frey, P. and George, P. (2000). *Mesh Generation. Application to finite elements.* Hermes.

Fuchs, A. (1998). Automatic grid generation with almost regular Delaunay tetrahedra. In S. N. Lab., editor, *7th International Meshing Roundtable*, pages 133–148.

Geiger, B. (1993). *Three-dimensional Modelling of Human Organs and its Application to Diagnostic and Surgical Planning.* Ph.D. thesis, INRIA, Sophia Antipolis.

George, P. (1997). Improvement on Delaunay based 3D automatic mesh generation. *Finite Elements in Analysis and Design*, **25**, 297–317.

Hang, S. (2001). *Tetgen. The Quality Tetrahedral Mesh Generator. Version 1.0 User's Manual.*

Kossaczkỳ, I. (1994). A recursive approach to local mesh refinement in two and three dimensions. *J. Comput. Appl. Math.*, **55**, 275–288.

Liu, A. and Joe, B. (1994a). On shape of tetrahedra from bisection. *Math. Comp.*, **63**, 141–154.

Liu, A. and Joe, B. (1994b). Relationship between tetrahedron shape measures. *Bit*, **34**, 268–287.

Liu, A. and Joe, B. (1995). Quality local refinement of tetrahedral meshes based on bisection. *SIAM J. Sci. Comput.*, **16**, 1269–1291.

Liu, A., Belliveau, J., and Dale, A. (1998). Spatiotemporal imaging of human brain activity using functional MRI constrained magnetoencephalography data: Monte carlo simulations. *Proc. Natl. Acad. Sci. USA*, **95**, 8945–8950.

Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proc. of SIGGRAPH)*, **21**(3), 163–169.

Marin, G., Guerin, C., Baillet, S., Garnero, L., and Meunier, G. (1998). Influence of skull anisotropy for the forward and inverse problem in EEG: simulation studies using FEM on realistic head models. *Human Brain Mapping*, **6**, 250–269.

Maubach, J. (1995). Local bisection refinement for n-simplicial grids generated by reflection. *SIAM J. Sci. Statist. Comput.*, **16**, 210–227.

Miller, G., Talmor, D., Teng, S., and Walkington, N. (1995). A Delaunay based numerical method for three dimensions: generation, formulation and partition. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 683–692.

Mitchell, W. (1992). Optimal multilevel iterative methods for adaptive grids. *SIAM J. Sci. Statist. Comput.*, **13**, 146–167.

Molinari, J. and Ortiz, M. (2002). Three dimensional adaptive meshing by subdivision and edge-collapse in finite-deformation dynamic-plasticity problems with application to adiabatic shear banding. *Int. J. Numer. Meth. Engi.*, **53**, 1101–1126.

Molinari, J.-F. R. (2000). *Three-dimensional finite element analysis of impact damage an erosion of metallic targets.* Ph.D. thesis, California Institute of Technology (CALTEC), Pasadena, California, USA.

Mosher, J., Leahy, R., and Lewis, P. (1999). EEG and MEG: forward solutions for inverse methods. *IEEE Trans. on Biomedical Engineering*, **46**, 245–259.

Mukherjee, A. (1996). *An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity.* Ph.D. thesis, Pennsylvania State University, University Park, PA, USA.

Parthasarathy, V. (1993). A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design*, **15**, 255–261.

Pescatore, J. (2001). *Maillages Homotopiques Tétraédriques des Tissus de la Tête pour le calcul du problème direct en magnétoélectro-encéphalographie.* Ph.D. thesis, Ecole Nationale Supérieure des Télécommunications., Paris, France. ENST 2001 E 034.

Phillips, J., Leahy, R., Mosher, J., and Timsari, B. (1997). Imaging neural activity using MEG and EEG. *IEEE Engineering in Medicine and Biology*, pages 34–42.

Plaza, A., Padrón, M., and Carey, G. (2000). A 3d refinement/derefinement algorithm for solving evolution problems. *Applied Numerical Mathematics*, **32**, 401–418.

Rivara, M. (1987). A grid generator based on 4-triangles conforming mesh refinement algorithms. *Internat. J. Numer. Methods Engen.*, **24**, 1343–1354.

Rivara, M. (1996). New mathematical tools and techniques for refinement and/or improvement of unstructured triangulations. *5th International Meshing Roundtable*, pages 77–86. Sandia National Laboratories.

Rivara, M. (1997). New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *Inter. J. Numer, Methods Engin.*, **40**, 3313–3324.

Rivara, M. and Inostroza, P. (1997). Using longest-side bisection techniques for the automatic refinement of Delaunay triangulations. *International Journal for Numerical Methods in Engineering*, **40**, 581–597.

Rivara, M. and Levin, C. (1992). A 3-d refinement algorithm suitable for adaptive and multi-grid techniques. *Communications in Applied Numerical Methods*, **8**, 281–290.

Ruppert, J. (1995). A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, **18**(3), 548–585.

Shewchuk, J. (1996a). Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *ACM First Workshop on Applied Computational Geometry*, pages 124–133, Philadelphia, PA.

Shewchuk, J. (1998). Tetrahedral mesh generation by Delaunay refinement. In *ACM Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, USA.

Shewchuk, J. R. (1996b). Robust adaptive floating-point geometric predicates. In *ACM 12th Sym. on Comput. Geometry*.

Shewchuk, J. R. (1997). *Delaunay Refinement Mesh Generation*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University., Pittsburg, Pennsylvania, USA.