IT 11 090
Examensarbete 45 hp
December 2011

Visual planning and verification of deep brain stimulation interventions

Elhassan M. Abdou

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Visual planning and verification of deep brain stimulation interventions

Elhassan M. Abdou

Deep Brain Stimulation (DBS) has resulted in a renaissance as an alternative way for treatment of parkinson's disease and essential tremor. Deep brain stimulation employ the use of high electric field to stimulate some brain centers. The electric field in the brain is generated from chronic implanted electrodes in the brain. The final position of the electrodes in the brain is specified by the aid of CT and MR scans for the patient's head before and after the operation. A study of electric field distribution in the brain is required to interpret and improve the action of DBS. In this master thesis project, Voreen was extended to visualize a multimodal volume of the CT and MR images. The MR volume was segmented to extract the brain from the skull in T1 weighted images. Some image processing techniques were developed to enhance the contrast of CT and MR images. In order to stimulate electric field in the brain, the neurologists are allowed to design and position the electrodes in the reconstructed volume. The electrodes and some pre-modeled electric fields can be visualized in the reconstructed volume and the slice views. A mesh generator was developed using delaunay tetrahedralization. The generated mesh can be sent to PDE solver to solve Laplace equation describing the distribution of electric field.

Handledare: Timo Ropinski
Ämnesgranskare: Robin Strand
Examinator: Anders Jansson
IT 11 090
Tryckt av: Reprocentralen ITC

Acknowledgements

First and foremost, I would like to thank my father and my mother who taught me everything I need to know and are making sure that I never loose my way. Thanks to my brother and sister who I hold dear for their support and for caring about me. Warm thanks to my lovely fiance for encouraging me and believing in me. My deep respect and appreciation to M. Elnaquib and Prof Yasser Kaddah for sculpting me and giving me a lot of their time to learn me how to develop my first program. Another thanks for my friends who supported me, stood by me in good and bad times. A special thanks to my supervisor Timo Ropinski for introducing me to the medical visualization world and giving me a chance to learn from his ideas and insights. I would like to show my deep appreciation for my examiner Robin Strand for his support and giving me time throughout this project. Also, sincere gratitude to the staff of Center of Image Analysis in Uppsala for teaching me and inspiring me with their insights and ideas. My deep appreciation for Michael Thune and Stefan Palsson for their guidance during the MSc program. Last but not least, I would like to express my deep respect to Uppsala University and Cairo University for giving me a chance to join their academic environment and helping me make some of my dreams come true.

Contents

1 Introduction **5**

1.1 Deep Brain Stimulation 5

1.1.1 Deep Brain Stimulation Operation 6

1.1.2 Target Localization 7

1.2 Aim Of The Thesis 8

2 Medical Image Data **9**

2.1 Computed Tomography(CT) 9

2.2 Magnetic Resonance Image (MRI) 10

2.3 Image Processing 10

2.3.1 Anisotropic Diffusion Filter 11

2.3.2 Contrast Enhancement 11

2.3.3 Histogram Equalization 12

3 Volume Visualization **14**

3.1 Ray Casting 14

3.1.1 Ray Set Up 16

3.1.2 Geometry Set Up 17

3.1.3 Compositing 17

4 Delaunay Triangulation **20**

4.1 Delaunay Triangulation 20

4.2 The Flipping Algorithm 21

4.3 Delaunay Tetrahedralization 24

5 Methods and Applications **25**

5.1 MRI Brain Segmentation 26

5.2 Multi-Volume Visualization 29

5.3 Field Modelling 30

5.4 Lead Modelling 32

5.5 Lead Slice Integration 35

5.6 Lead Volume Rendering 36
5.7 Tetrahedra Generation 39

6 Results 41
6.1 Full View Visualization 41
6.2 Meshing WorkSpace 44

7 Conclusion 50
7.1 Future Work 50

Chapter 1

Introduction

1.1 Deep Brain Stimulation

Deep Brain Stimulation (DBS) has resulted in a renaissance as an alternative way for treatment of Parkinson's Disease (PD) and essential tremor (ET). Benabid reported in 1987 [14], chronic stimulation on patients with contralateral thalamotomy, achieved partially tremor relief. Neverthless, patients with bilaterlateral thalamotomy showed greater relief from tremor. Benabid stated that the partial relief for stimulated patients was due to the limitations of producing electric pulses of higher frequency. He believed that 200Hz was the optimal frequency to stimulate ventralis intermdius (VIM). VIM is rarely used these days for PD treatment due to limitation in treatment of parkinson rigidity and bradykinesia. Benabid et al in [13] demonstrated the efficacy of SubThalamic Nucleus (STN) DBS in parkinsonian patients treatment and it is safer than VIM DBS. After one year of following up for twenty patients, Benabid et al. showed that 60% improved in Unified Parkinson Disease Rating Scale (UPDS) in compared to VIM DBS. Other studies [9], tried to stimulate Globus Pallidus interna (GPi) to treat PD, but it was not as promising as STN stimulation. Case studies showed that DBS can not remove motion disorder effects but dramatically reduces the effect of antiparkinson medications.

Federal Food and Drug Administration (FDA) accepted DBS as medical treatment option for Essential tremor and Parkinson's disease [10]. Neverthless, DBS was used in Europe a decade ago as a medical treatment for most movement disorders.

1.1.1 Deep Brain Stimulation Operation

In order to stimulate some regions in the brain, Deep Brain Stimulation techniques, employ the use of high frequency electric field in the brain. The electric field is generated through chronic implanted electrodes into the deep centres of the brain is depending on neurologic and psychiatric disorder.

The DBS system assembled from three components: the lead, the extension and the implanted pulse generator. A typical clinic DBS lead contains four electrodes for electric voltage generating. Electrodes are uniformly spaced from the distal end of the lead. A lead is a thin insulated wire which is introduced in deep brain centres through a small lesion in the skull of electric resistivity less than 100Ω The lead implant procedure will not be widely accepted without the introduction of *Stereotactic framework* to guarantee accuracy and facilitate surgical planning.

Stereotactic frame was first introduced for human use by Speigel and Wycis to fix the skull by ring and coordinates were determined using radioactive landmarks [24] [20]. Subsequent development of stereotactic frame improved target accuracy for electrode implanting (see figure 1.1). Medical image modalities blossomed during the 20th century, Computed Tomography and Magnetic Resonance Imaging marked major steps in operation planning. Stereotactic immobilized landmarks are exploited by automated algorithms to fuse data from previously mentioned modules [6] (Target localization and image modalities explained in Section 2 and 1.1.2). The extension is connect-

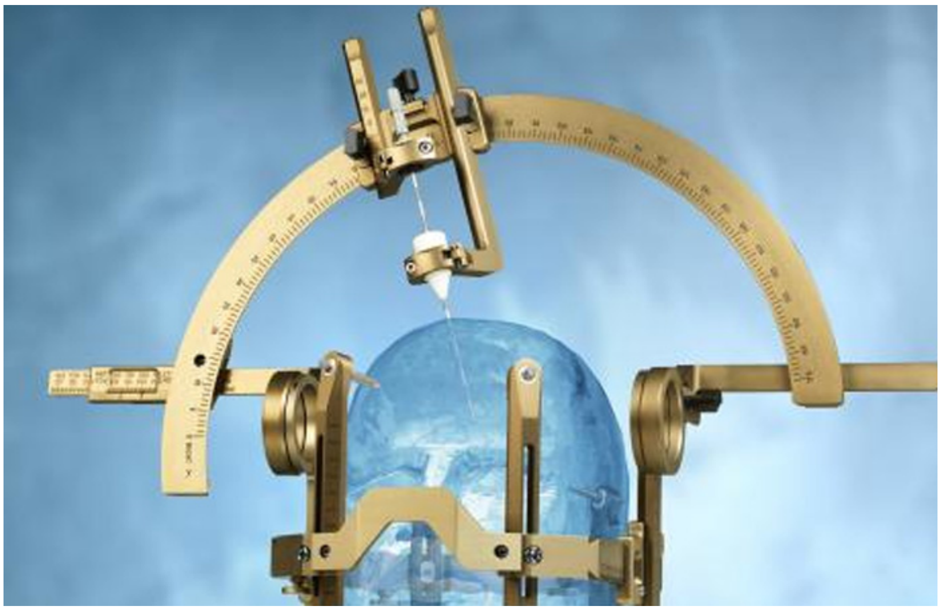


Figure 1.1: Leksell Stereotactic frame

ing end of the lead to the neuro-simulator generator [1]. The extension is an insulated wire implanted under scalp and neck skin, to the neuro - simulator generator which is usually placed in the clavicle or abdomen. Neuro-simulator is a battery based electric pulses generator.

1.1.2 Target Localization

Early hypotheses, claimed that electric pulses inhibited the brain centres activities, mimicking *thalamotoy* [1] or *Pallidotomy* [2]. Recent studies, showed that neurons activities nearby activated electrodes are decreased due to synaptic inhibition, while output from stimulated nucleus is increased by directly activating the axons.

The selection of target is influenced by major disable symptoms that will be alleviated from parkinsonian patient. The main comparison in this section is between GPi DBS and STN DBS for their efficacy in fixing most PD motion disorder. GPI DBS shows a dramatic relief from levodopa-induced dyskinesias. Also, UDPRS motor score is improved in range of 30-50% for bilateral GPi DBS. Clinical results stated significant improvements in tremor and posture gait. On the other hand, Bilateral STN DBS achieves 50-70% improvement for motor score in UPDRS. In contrast to GPI DBS, STN DBS shows sufficiently great relief in rigidity, akinesia and tremor. STN DBS improves gait disturbance and posture instability if the patient showed response to levodpa dose before surgery. In opposite to GPI DBS, STN DBS patients reduce the dose of levodpa to 50-60%, where in GPI DBS case it remains the same. In the long term, STN DBS is equivalent or superior to GPI DBS in most motion disorder except for dyskinesias, if the levodopa dose remains or reduces.

In other words, target selection is based on several factors, most disabling symptoms, response to medication and goals of therapy. For instance, if dyskinesias is a major disability then GPi will be the treatment plan. The treatment with GPi will follow up in parallel with levodpa medicine. On the other side, STN may be better for most PD motion disorder and with advantage of medical dose reduction as mentioned above [12].

[1]thalamotomy: invasive operation where a very tiny part in the brain is removed from the thalamus to block tremor

[2]Pallidotomy: invasive operation to destroy over active globus pallidus to alleviate rigidity

1.2 Aim Of The Thesis

Although, it is obvious that DBS inhibits the activities of neurons near by the brain centres for parkinsonian patients, more research is needed to interpret the action of DBS. Some brain centres are largely unclear for treatment of some other motion disorder. Various interdisciplinary methods and tools are used to visualize the effect of the electric field in the brain. In order to visualize and allocate targets several medical imaging modalities are used, that produce enormous amount of data describing different anatomical and functional structures of the brain. Mathematical modelling and partial differential equations solvers are employed for evaluating the electric potential distributions produced by the electrodes. We aim in this thesis, to develop a visualization tool for electric potential distribution in multi-modal volume. generated from reconstruction of Computed Tomography (CT) and Magnetic resonance imaging (MRI) for the brain and register them according to pregenerated matrix from the stereotactic frame points. We used the VOlume REndering ENgine `Voreen` [11] to develop multimodal visualization, image processing, volume handling and other visualization processes. In order to generate the electric field in an elegant way or exploring new brain centres, we allow the physician to interactively place the lead, specifying numbers of electrodes and visualize the electrodes position in different views and 2D slices. After designing the case study, the physician can send it to mesh generator that was developed in this thesis.

Chapter 2

Medical Image Data

Since the inception of Computerized Image modalities the demanding for *Computer Aided Surgery (CAS)* and simulation programs is increased. In order to use the large amount of data acquired from different modalities, volume de-noising, segmentation, volume fusion and visualization techniques are acquired by CAS. It is highly competitive to have one application that contains these techniques, but bundles of software are used to provide complete studies of brain anatomy and electric evaluation. The step of processing of the data is crucial to provide an elegant data to be visualized. Image segmentation aims to divide the volume into parts that strongly correlated to objects or areas in the available data, which strongly rely on image processing steps. This chapter reviews some computerized image modalities in Sections 2.1 and 2.2 that are used during the DBS operation and the electric field studying. In Section 2.3 we explain some image processing theories that are used for processing the data from previous modalities.

2.1 Computed Tomography(CT)

Computerized Tomography was introduced in 1972 as an advent of X-ray imaging. CT provides a series of cross sectional X-ray images. Most of tomographic techniques are depending on back projection algorithms in order to reconstruct a volume from the obtained sectional series. The construction of CT slice from the X-ray images, with different angles, resolves X-ray image problems; imaging structures that are parallel to the conventional X-ray beam, prevent scatter radiation from large X-ray beams and imaging structure with smaller difference.

The CT is equipped with X-ray emitters and detectors that rotates around the patient to construct cross sectional images from different angles. Because the physics of the CT and X-ray is the same, the image is constructed by electrons density. The CT improves the anatomical resolution by synthesis the image from different angles [3].

Because the X-ray image represent the electron density in a certain region, the CT is sometimes used for image soft tissues by injecting high contrast material in the blood. Although the CT can be used for soft tissue imaging, MRI is supreme for this task.

2.2 Magnetic Resonance Image (MRI)

Magnetic Resonance Imaging (MRI) is an image technique that encode electric pulses induced from the magnetic momentum of hydrogen atoms spinning in the human body. The physics behind MRI relies on the *Faraday law of electric induction*. Human soft tissue is rich contentment (water and fat) of hydrogen atoms that are spinning randomly. By applying a strong magnetic force it becomes aligned in the magnetic field direction. The magnetic field is in parallel to high conductivity coil. Pulses of radio frequency are applied to the aligned hydrogen atoms pushing it to a direction perpendicular to the coil. Once the pulses stopped, hydrogen atoms begin to return back to the alignment state. By Recording the relaxation time (time required to return back to the alignment state) the T1 Weighted MR image is constructed. MR can produce another kind of images called T2 Weighted images. Before the application of the radio pulses, the angular momentum of spinning hydrogen atoms are random, and it is aligned by a radio pulses application. T2 measures the decreasing of signals from angular momentum cancellation. There are predefined lookup table for both relaxation times, to predefine tissues that are scanned.

The MR image is influenced by several factors; the strength of stationary magnetic, the radio pulses signals, and the amount of spinning per cubic. The later factor is influenced by the environment of tissue and temperature [5],[4].

2.3 Image Processing

An object can be seen in the image because of it is contrast to the surrounding environment. The ultimate goal of image processing is to improve the object contrast (Signal) and inhibit noise. Image processing filtering in general should remove noise in regions of physical homogeneity, minimize information loss and enhance boundaries. This increases the Signal to Noise Ratio (SNR).

In this section some of the processing used in order to improve Signal to Noise Ratio (SNR) and visual appearance for visual analysis are shown.

2.3.1 Anisotropic Diffusion Filter

MR image are composed of grained regions separated by some edges. In order to preserve edges and smooth regions, a non linear anisotropic filter is applied to the image. Perona and Malik, introduced the use of iterative anisotropic diffusion filter to smooth the regions [22]. The diffusion filter depends on the heat diffusion equation in smoothing regions (flow of heat) and preserving strong edges (heat stops at boundary) as shown in the following equation:

$$\frac{\partial u(x,t)}{\partial t} = div(c(x,t) \bigtriangledown u(x,t)) \tag{2.1}$$

The diffusion strength is controlled by c(x,t). In this thesis we used the formula in equation 2.2 to smooth the MRI:

$$c(x,t) = \exp(- \left(\frac{| \bigtriangledown I(x,t)|}{K} \right)^2) \tag{2.2}$$

K is the constant that controls the diffusion strength and I(x,t) is intensity in x at time t. It can be easily shown from equations 2.1 and 2.2, that the maximum flow is at $\bigtriangledown I(x,t)$ equal to K and decrease when $\bigtriangledown I(x,t)$ moves far from K. Gerig et al, used this filter to enhance the MR images [19].

We applied this filter on every 2D slice with 8 neighbour connections. We used explicit schema to solve equation 2.1. In order to guarantee stability, the time step in 2D should not exceed 1/5 and 1/7 in 4 neighbourhoods and 8 neighbourhoods respectively. See [19] for numerical analysis.

2.3.2 Contrast Enhancement

The perceived MR images histogram were compressed in a very small region with intensity not exceeding 255. This makes the task of visualization and transfer function manipulation a tedious task. We enhanced the MRI data by a very primitive gray scale transformation to stretch the intensity distribution over the whole range. We applied a naive power function transformation on every voxel. The value of constants c and α in equation 2.3 are changed according to the case

$$s = cr^{\alpha} \tag{2.3}$$

The histogram of the transferred volume spreads out, making the transfer function manipulation for the volume easier and clearer for visual analysis.

2.3.3 Histogram Equalization

The CT images are characterized by very low contrast images for soft tissue. We tried to increase the contrast by flatten the CT's histogram. *Histogram equalization* is an automatic gray scale transformation technique that tries to flatten the histogram of the image. The resulted images are evenly distributed gray level and visually contrast enhanced. The input image of size N and gray scale $[p_0, p_k]$ is mapped using gray scale transformation function T(p) to an image of gray scale $[q_0, q_k]$. The histogram is a discrete probability density function. The histogram equalization tries to find a monotonic function $q = T(p)$. The monotonic property of T(p) implies

$$\sum_{i=0}^{k} G(q_i) = \sum_{i=0}^{k} H(p_i) \tag{2.4}$$

The continuous form of equation 2.4 is

$$\int_{q_0}^{q} G(s)\,\mathrm{d}s = \int_{p_0}^{p} H(s)\,\mathrm{d}s \tag{2.5}$$

The equalized histogram G(q) is an uniform probability density function of constant value

$$G(q_i) = \frac{N^2}{q_k - q_0} \tag{2.6}$$

The left hand side in 2.5 is replaced by the right hand side in equation 2.6

$$\int_{q_0}^{q} \frac{N^2}{q_k - q_0}\,\mathrm{d}s = \int_{p_0}^{p} H(s)\,\mathrm{d}s \tag{2.7}$$

The transfer function is derived from previous equation

$$q = T(p) = q_0 + \frac{q_k - q_0}{N^2} \int_{p_0}^{p} H(s)\,\mathrm{d}s \tag{2.8}$$

The previous equation is discretized using Riemann sum to the following equation

$$q = T(p) = q_0 + \frac{q_k - q_0}{N^2} \sum_{i=0}^{k} H(p_i) \tag{2.9}$$

The internal summation in the previous equation is called *cumulative histogram.*

The algorithm for finding the gray scale transformation *T(p)* from the previous equations can be found in [7] and is showed below.

1. let p be intensity counter, and G is the maximum gray level available
2. create an array H of length G
3. calculate the histogram of every slice
4. construct a cumulative histogram H_c:

$$H_c[0] = H[0] \tag{2.10}$$
$$H_c[p] = H_c[p-1] + H[p] \tag{2.11}$$

5. gray scale transformation function

$$T(p) = round\left(\frac{G-1}{size-of-image}H_c[p]\right) \tag{2.12}$$

6. apply voxel intensity to T(p)
7. for every slice in the volume return back to step 2.

This algorithm was developed to facilitate the manual fusion process of the CT and the MRI data. The equalized CT volume reveals some of the internal structure of the brain, like the fissure between the two hemispheres and some ventricles, allowing more landmarks selections. In some cases, it allows the user to easily select the value of bone being visualized by threshold value selection.

Chapter 3

Volume Visualization

Scientific visualization in general aims to create images that convey information of simple, complex and high dimensional data. This involves the use of graphics techniques to gain insight in the data and perceptual understanding of the data.

Medical data are huge data and usually equidistant sampled. Two or three dimensional samples are organized in rectilinear two or three dimensional grids respectively, see figure 3.1. Sophisticated techniques of computer graphics and image analysis are used in order to visualize the data. The aim of medical visualization is to aid the user to focus in specific data and to perceptual removing other data. Interactive user interface are used in medical visualization applications to control how data are visualized.

In this chapter, we explain the main techniques that are used to visualize the medical data in this thesis.

3.1 Ray Casting

Volumetric ray casting is a *Direct Volume Rendering (DVR)* technique, where every voxel in 3D data is used in visualization. DVR creates image from all the available data without a prior surface extraction. Every pixel color is computed by compositing color from every sample in the whole volume along the view direction. The compositing of color and transparency is computed from the optical model in the integral equation 3.1

$$I(D) = I_0 \exp^{-\int_{s_0}^{D} k(t)dt} + \int_{s_0}^{D} q(s) \exp^{-\int_{s}^{D} k(t)dt} ds \tag{3.1}$$

Equation 3.1 describes the accumulation of color at exit point D. I_0 is the initial intensity at point s_0. The $\exp^{-\int_{s_0}^{D} k(t)dt}$ is the decaying factor of the

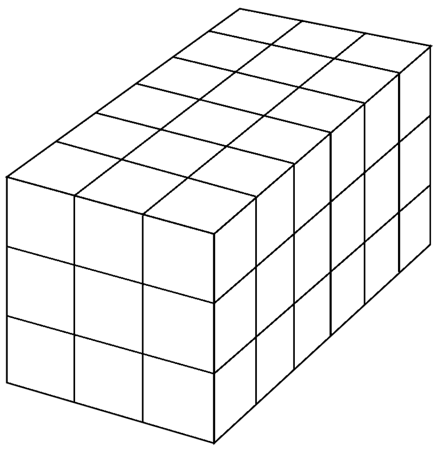


Figure 3.1: Regular Cartesian 3D grid

emitted energy along the path from s_0 to D. K(t) is an absorption factor. Q(s) is an active emitting point along the path of the ray. Q(s) energy decayed along the path by the decaying factor $\exp^{-\int_s^D k(t)dt}$.
The approximated discrete representation of equation 3.1 can be expressed by equation 3.2

$$I(D) = \sum_{i=0}^{n} c(x_i) A_i \prod_{j=1}^{i-1} (1 - A_j) \tag{3.2}$$

where $c(x_i)$ denotes the color at position x_i and A_i is the opacity at sample i.

Ray Cast sends a ray from every drawing pixel to the volumetric data and accumulate the color according to equation 3.2. Ray Cast is more preferred than the 2D texture or 3D texture based volume rendering for rays flexibility and full control of pixels compositing. The ray cast pseudo code can be written as,

```
1. Determine the volume entry and exit point
2. Calculate ray direction
3. While ray is inside volume
   Access data at current position
   Composite color and opacity
   Advance ray position
4. End
```

The following subsections explain how ray casting was implemented.

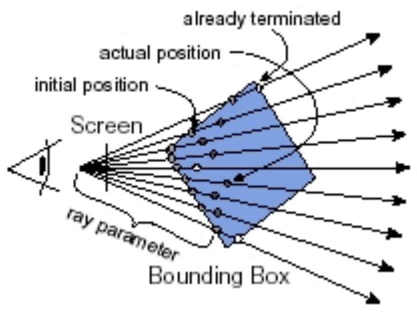


Figure 3.2: Ray Cast explanation, www.voreen.org

3.1.1 Ray Set Up

The viewing line is specified by three parameters; starting point, end point and unit vector. The line can be described by using the ray as a direction vector in the parametric equation 3.3

$$P_{line}(t) = Start + dir * t \tag{3.3}$$

One can use OpenGL capabilities of hardware interpolation to generate the start points and the end points of the rays. First the Eye coordinate matrix is adjusted after that a unit length cube [0,1]×[0,1]×[0,1] is drawn two times. In the first time, the back face is culled and rendered to a 2D color framebuffer. The coordinate of the vertices are the starting points. In the second render time, the front face is culled and rendered to another color framebuffer. These are the end points as shown in figure 3.3. By subtracting the starting from the end framebuffer we get the *dir* in equation 3.3.

Voreen was implemented in performance wise way. The volume bound box is used instead of the previous unit length box and the same previous algorithm is applied to it. The values of the framebuffers are the starting and ending values on the volume border exactly.

The Volume data is loaded to the GPU texture memory as a 3D texture. The ray is sampled by a user defined value. Data is accumulated along samples according to the compositing mode. If a ray sample does not coincide with voxel, the GPU provides tri-linear interpolation to calculate the sample value. The oversampling of the data may decrease the interactive response of the program.

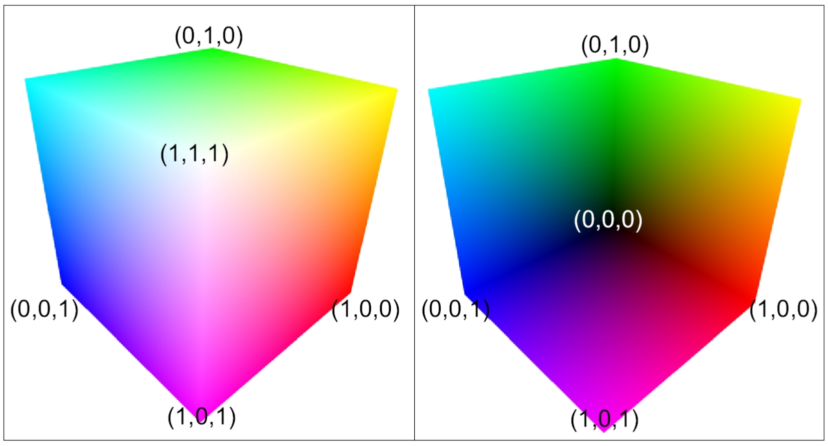


Figure 3.3: Entry and exit textures

3.1.2 Geometry Set Up

A *Proxy Geometry* is a geometry that is used to map the texture to its coordinate. Voreen maps the 3D sampler data to a cube of dimensions [-1,1]×[-1,1]×[-1,1]. If the 3D data is not equally dimensioned, Voreen maps the minimum dimension of 3D texture data to -1, maximum dimension to 1 and use these mapping values to map the other dimensions over the cube. For the single volume ray casting case, the entry and exit depth values are computed in the same texture space of the 3D volumetric data. The samples are queried from the 3D texture using Texture3D in glsl.

3.1.3 Compositing

The *Transfer function* maps the scalar values from 3D texture to color values (R,G,B,A). The *Classification* is the use of transfer function to classify volume according to its scalar values. There are two ways to transfer the scalar value to color, either by pre-interpolative classification or post-interpolative classification [17]. Pre-interpolate technique interpolates between RGBA values before interpolation of scalar data. Post-interpolate technique interpolates between scalar data values first and then applies the transferring function onto the interpolated data. It is shown in figure 3.4 that both techniques lead to different visual appearances.

Voreen was implemented to use the post-interpolative technique. The implementation of Post-interpolative classification is a straightforward texturing technique. 1D or 2D texture unit may be used for classification. The color values are stored in 1D texture memory which has the same resolution

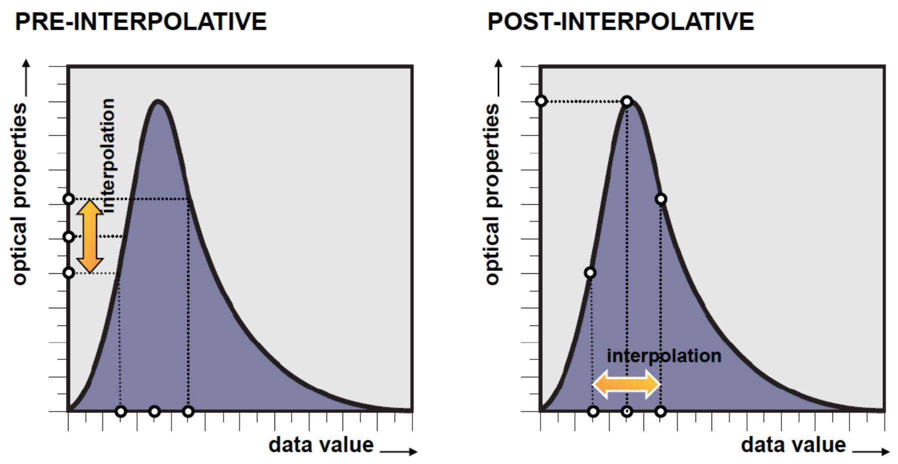


Figure 3.4: left: Pre-interpolative Transfer function, Right: Post interpolative Transfer function

as the scalar values resolution in the 3D texture. The rays are sampled and sampler3D in the fragment shader is queried to get data. The interpolated scalar data is used as fragment coordinate. The fragment coordinates are used to query the 1D texture to get the color and opacity of the scalar value.

A comparison between the two classification techniques can be found [17]. Several kinds of compositing can be used to construct the final image. Voreen has the following compositing techniques; Direct Volume Rendering (DVR), Maximum Intensity Projection (MIP),Iso Value (ISO), First Hit Point (FHP) and First Hit Normal (FHN). When applying DVR compositing, opacity correction is used so that the overall transformation of intensity is consistent [2]. The opacity correction is used for the opacity value modification using equation 3.4.

$$A = 1 - (1 - A_0)^{\frac{s_0}{s}} \tag{3.4}$$

A_0 is the opacity,
s_0 is sampling step size in texture and
s is sampling step size in world.

The ray accesses the data in a front to back order. The data is composited to the final image by using the following equations.

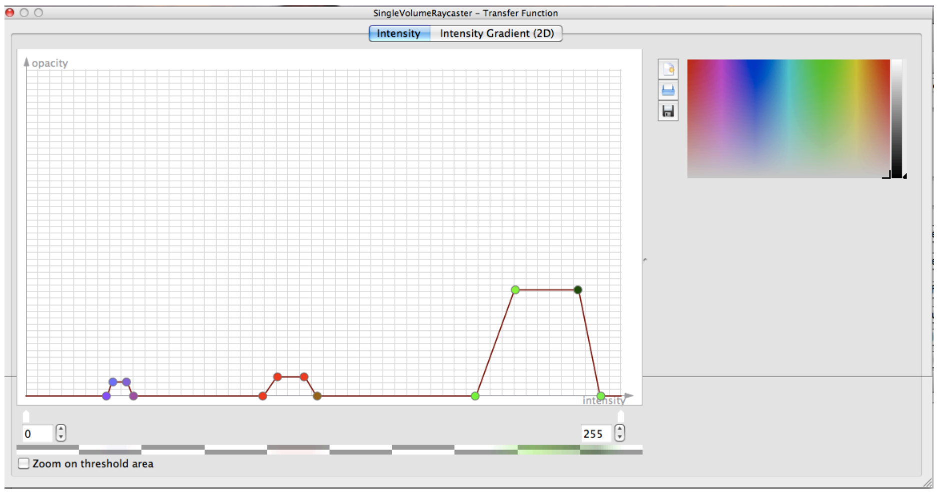


Figure 3.5: Post-interpolation from Voreen. Transfer function design is saved in 1D texture.

$$C^{out} = C^{in} + (1 - A^{in})A \times C \tag{3.5}$$
$$A^{out} = A^{in} + (1 - A^{in})A \tag{3.6}$$
$$\tag{3.7}$$

C^{out} is the observed color,
C^{in} is an intermediate color of previously composited color,
C is the current voxel,
A is the absorption of the current voxel,
A^{in} is an intermediate opacity value of previously composited opacity.

Chapter 4

Delaunay Triangulation

The breaking down of polygons to a set of non-intersecting triangles is called *triangulation*. The *meshing* processing is the process of converting a domain of points to a set of triangles or tetrahedra. The partial differential equations that governs many physical phenomena can not be solved analytically except for some simple cases. Thus, the complex domain needs to be broken down to simpler and flat elements. The differential equation is discretized over the sub-domains. This results in a linear system that can be solved on a computer. The goal of the meshing process is:

1. **Control the element size**: Smaller elements becoming larger and vice versa. The element size at boundaries or areas with high variations should be small enough to approximate fast changes. Large angles degrade the quality of the numerical method solution. Small angles will result an ill condition numerical systems. A small round off error degrade the accuracy of the systems. Ill condition converge slowly when solved by iterative method.

2. **Control total size**: Lower number of elements is preferred for faster computation.

In this chapter, the 2D delaunay triangulation is explained first then the extension to 3D delaunay tetrahedra is explained.

4.1 Delaunay Triangulation

The meshing process starts by having a set of points P. Then, a set of non interior intersecting triangles T connects the points to approximate the domain. The union of T completely fill the convex hull of P.

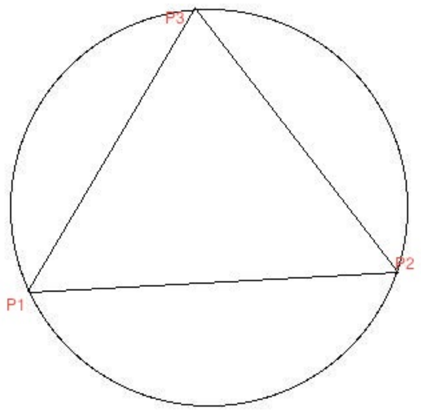


Figure 4.1: Empty circle: the delaunay triangle's vertices on the border of the circle

In 1934, Delaunay [15] defines *Delaunay Triangulation DT* of P as a plane graph. Any two points $p_i, p_j \in P$ form a delaunay edge if there is an empty circle passing through p_i and p_j. A circle is empty if there is no vertex in it, vertices are permitted to be on the border of the circle. In the same manner, a triangle is a delaunay triangle if there is an empty circumcircle that passes by the triangle's vertices and the triangle is in the plane graph as shown in figure 4.1.

Every edge of the convex hull of the set of vertices is a delaunay edge because it is always possible to find a circle that passes by only the two ending points as shown in figure 4.2.

The flipping algorithm is the simplest idea to compute the delaunay triangulation. It was developed on a fact if a triangle T is a delaunay then every edge in it is a delaunay edge.

4.2 The Flipping Algorithm

An edge is called *local delaunay* if the edge is on the convex hull of the domain or it is a delaunay in two triangles contain this edge. The flipping algorithm starts by arbitrary triangulation for P. It checks every edge if it is not local delaunay, it will be flipped. Flipping begins by removing the none local delaunay edge. The result is a quadrilateral. The other two opposite vertices are connected by flipping the edge as shown in figure 4.3.

Lemma If every edge in triangulation T is locally delaunay, T is a delaunay of points P.

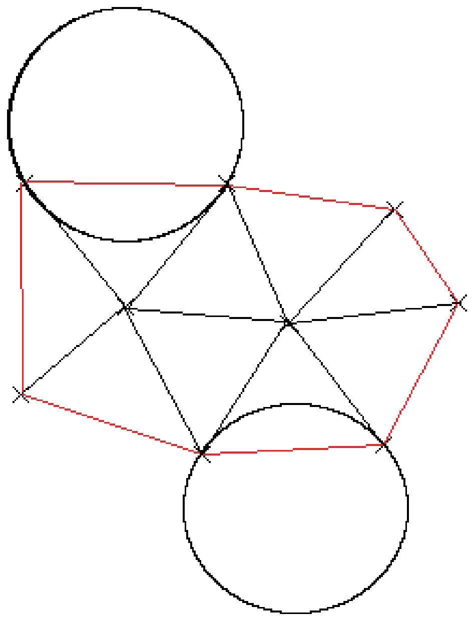


Figure 4.2: It is possible to find an empty circle that passes by the edge on the convex hull, so it is a local delaunay edge.

Proof Assume any arbitrary points p∈P and x∈ △ABC. The triangle ABC edges are local delaunay edges. If px segment crosses any edges of the local delaunay edges, p does not belong to △ABC. Assume another triangle ABD and $x^{'}$ is on px in △ABD. Because px' crosses one other local delaunay edge AB, p again is not in △ABD. Both △ABC and △ABD do not include p. So we can conclude that no points in p intrude any triangles in T.

The only condition that is needed to produce delaunay triangles, is that there are not any four vertices that are cocircular. This can not be guaranteed in medical images or structured grid in general. This can be solved by using a small perturbation in the vertices but this not a desirable solution.

Theorem Among all triangulation the vertex set, the delaunay triangulation maximize the minimum angle [21] and minimize the maximum circumcircle [16].

Unfortunately, the properties in the theorem above is not extended to higher dimension than two.

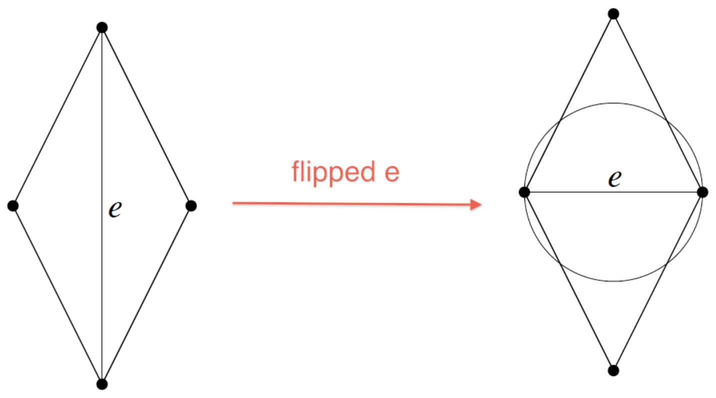


Figure 4.3: Flipping algorithm. Edge e is not local delaunay. Edge e is flipped and becomes a local delaunay.

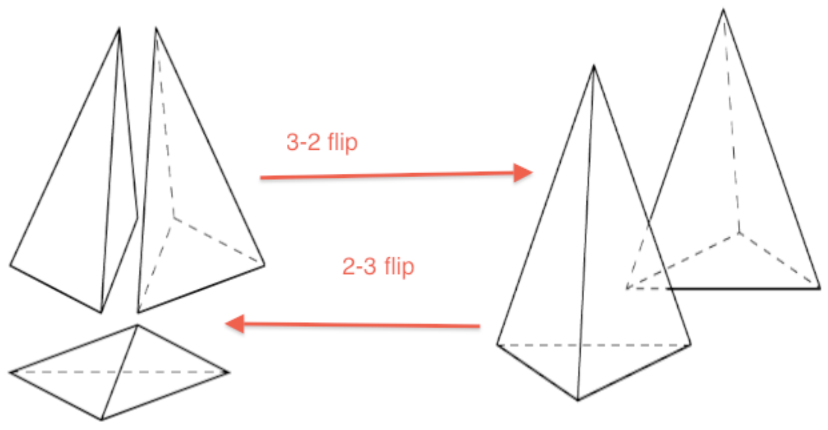


Figure 4.4: 2-3 and 3-2 flipping

4.3 Delaunay Tetrahedralization

The Delaunay tetrahedralization is a natural extension of the delaunay triangulation. A tetrahedron, triangle face or an edge is a delaunay if there is an empty sphere that passes by the vertices.

The edge flip algorithm is extended from 2D to 3D to compute the tetrahedra. The 3D flipping starts by arbitrary hexahedrons with triangle faces. The flipping algorithm starts by arbitrary splitting of the hexahedron to two tetrahedra or three tetrahedra. There are two kinds of 3D flipping algorithm; 2-3 flip and 3-2 flip. The 2-3 flip converts two tetrahedra to three tetrahedra as shown in the figure 4.4. A tetrahedron with a small angle near by zero is called *kite* or *sliver* tetrahedron. It can be modified either by eliminating some faces or by perturbation for the vertices.

The flipping algorithm in 2D or 3D does not work if the quadrilateral or hexahaderon containing the edges or faces is not convex. A modified algorithm of flipping algorithm is used to overcome the problem the flipping algorithm get sometimes stuck in local optimum [21].

A delaunay mesh refinement modifies the tetrahedra produced in the previous discussed way to meet some constraints. The mesh refinement works on the produced mesh to remove the *sliver* or *kite* defect and control node's size by insertion some vertices. The insertion places of vertices are selected to conform the boundary approximation and improve mesh quality.

Chapter 5

Methods and Applications

In this thesis, some processors and shaders were developed to extend *Voreen's* functionality for visualizing and processing the CT and MR volumes. We assume that the two volumes are registered. The MR volume is first processed to segment the brain out from the skull. After loading the two volumes, the MRI suffers from low contrast, so it is processed by using a power transformation to enhance the contrast. A multi-volume ray caster was developed to visualize both volumes. The lead was modelled using *OpenGL* and quadric objects from *GLU*, which was integrated in different views supported by Voreen. Volume data are perturbed to a certain value in the lead area, to be segmented easily by using a simple threshold algorithm or surface extractor for mesh generation. In this thesis, modules were developed as extensions to Voreen. In this chapter, some terminologies used by Voreen frequently are explained. A *Processor* is a group of classes or a class that does a specific process. It may have an input and an output. The processor inherits its functionality from various classes depending on the processor's functionality. Each processor overloads the *process* function from the parents classes. The process carries out the main functionality of the processor. A clear explanation to add new modules and processors implementation can be found in the tutorials in [11].

The next sections give a deep explanation about the algorithms that are used to establish this thesis work and integration of the solution with Voreen. We start first with the preprocessing stage. After this we explain the visualization stage and modelling. Then we add the module to modify the volume for the lead insertion. Finally, the meshing module is added to generate a mesh of the brain.

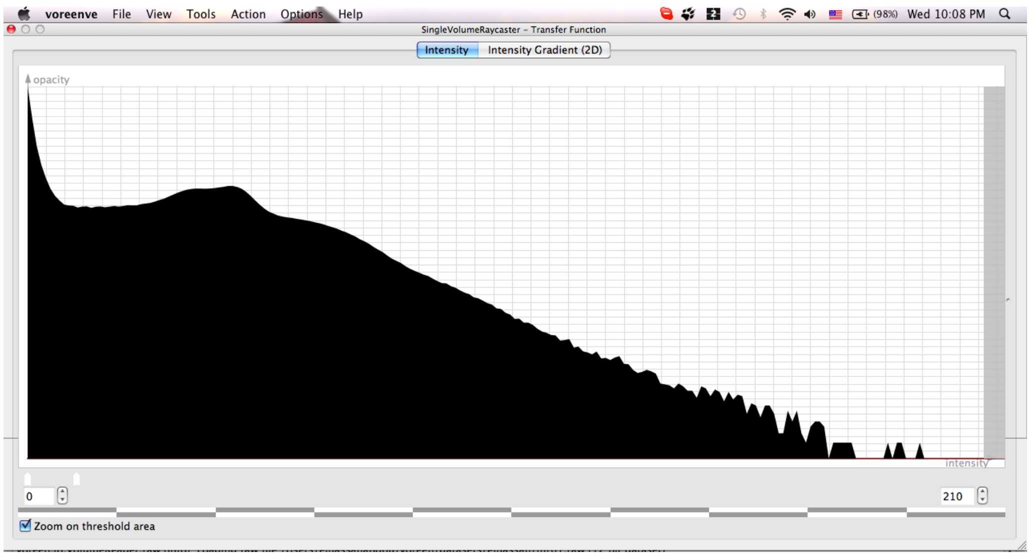


Figure 5.1: T1 MRI histogram

5.1 MRI Brain Segmentation

In this section, we proposed a 3D mathematical morphology method to segment a brain from the skull in 3D MRI. This method uses prior anatomical structure information to be converted morphological method. The algorithm works on T1 weighted MRI of the brain. The algorithm can work on images of different acquisition sequences like; MPR3D, SPGR and inversion. The algorithm that is described here, can be found in [25]. A typical T1 weighted MRI histogram is shown in figure 5.1. It may be stretched in different ways, but it has the same shape as in figure 5.1. The developed algorithm can be divided in two different stages. The first stage converts the gray valued volume to binary volume by thresholding the images. The second stage applies the 3D morphology operations and image filling to generate a brain mask. In [25] a supervised learning technique was used to generate threshold values. The output of this supervised technique, is statistical values as shown in the table 5.1.

The mean values of the brain organs are arranged as following:
$\eta_{air} \leq \eta_{skull} \leq \eta_{CSF} \leq \eta_{muscle} \leq \eta_{gray} \leq \eta_{white} \leq \eta_{fat} \leq \eta_{blood}$

The brain can be defined as an elliptical shape that is surrounded by a dark circle(skull) and thin bright circle(skin). The threshold values depend on the mean and standard deviation of the brain structure. The structure elements are balls and are approximated by using all close neighbours to a

structure	η	σ
putamen	66.3	5.75
lateral Ventricle	16.0	5.11
caudate nuclei	62.0	4.98
white matter	79.8	4.54
muscle	57.8	7.48

Table 5.1: Brain organ statistical values

cube. The distance function is a 3D version of the chessboard distance and is denoted by D^{26}. The size of the balls are measured in millimeters that can be easily converted to voxels. The voxel size is known in advance, which is used to convert the cubes to voxels. The following steps explain the application of the algorithm on the 3D MR brain image, before the insertion of the leads. The result of every step is presented in figure 5.3. The original image is shown first and the final image is shown at the end.

1. Thresholding the image to eliminate as much as possible from the brain surroundings. The lower value of threshold removes the CSF and the skull. The higher threshold value discards higher density materials (eyes, vessels and fat). The output from this step is a binary image representing the brain and skin. The threshold values were selected in the same manner as mentioned in [25]:

   $$\text{Threshold}_1 = \eta_{vent} - \sigma_{vent} \tag{5.1}$$
   $$\text{Threshold}_2 = \eta_{white} + 2 * \sigma_{white} \tag{5.2}$$

   These values can be seen on the MRI histogram in figure 5.2. The threshold values are crucial for the following steps to work perfectly. It is difficult to choose the threshold value due to the fact that it depends on the contrast acquisition of the MR images. Nevertheless, equations 5.1 and 5.2, give a good start to adapt the values. Only narrow connections between the brain and skin are acceptable in the output image from this step.

2. The aim of this step is to remove the narrow connection from the previous step between the brain and the skin. We erode the image with cube (2mm), which in our case is approximately 3 voxels.

3. A 2D filling algorithm is applied to every slice in the image to fill the holes in the connected components.

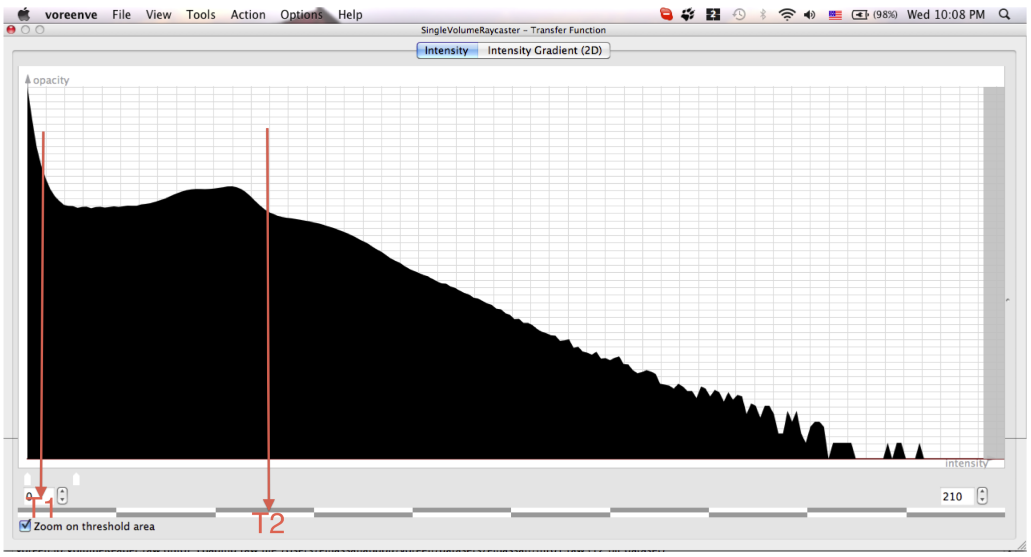


Figure 5.2: Threshold values

4. Big connections between the brain, the eyes and the ears are suppressed. This is implemented by eroding the output volume from the previous step with a cube (5 mm= 8 voxels).

5. The aim of the previous step is to generate connected components that are separated from each other. The biggest connected component is marked and the other components are discarded.

6. Due to the successive erosions in the previous steps, some of the parts of the brain are eliminated. By applying conditional dilation on the result from the first step and using the marker from previous step, the brain is reconstructed. First, we close the previous obtained marker with a structure element, bigger than the sum of the previous erosion structure elements. After this we get the minimum voxel values from both volumes.

7. A 2D filling algorithm is applied to every slice to fill the missing ventricles inside the brain. The brain mask is now complete.

8. The final step is to assign the gray value back to the resulted mask.

The Matlab methods developed above can be integrated in Voreen through *Matlab* processor.

5.2 Multi-Volume Visualization

The main output from this thesis is to give an insight of a CT and MR volumes in one scene. The volume is produced from the reconstruction process, helps to understand the brain structure and how the electric potential is distributed in it. In this section, shader algorithms used to visualize the MRI and CT data are explained. The shader algorithm present here is a modification of the primitive single ray casting algorithm mentioned in Chapter 3. CT volumes are used to visualize the external shell of the brain i.e. the skin and skull. The previous result of the segmented volume will be used instead of the original MRI data.

The texture data needs to be mapped to the proxy geometry. Voreen is supported with a *MultiVolumeProxyGeometry* processor to generate the proxy geometry for multi-volume input data. The output from this processor is a geometry list that was implemented in *MeshListGeometry* class. The *MeshListGeometry* class is used to encapsulate the geometry information of the generated proxy cubes. For every input volume to the *MultiVolumeProxyGeometry* processor, a proxy cube is added to the mesh list. The proxy geometry for every volume is generated in the same way as mentioned in Chapter 3.

The generated mesh list is used by *MeshEntryExitPoints* to estimate the entry points, exit points and ray's direction as mentioned in Chapter 3. A slight modification was applied to the previous algorithm to render the depth cube. First, the exit points are computed. The depth buffer is cleared by 1.0 and the depth function is set to GL_GREATER. After that, the front face is set to be culled. Then, the whole cubes in the input mesh List are rendered. The output from this is a 2D color buffer of the exit values of the ray. Finally, the render target is deactivated. Next, the entry points are computed. The depth function test is set to GL_LESS and the back face is set to be culled. After that, all of the cubes in the mesh list are rendered again. The output from this is a 2D color buffer representing the entry points. Voreen defines the proxy geometry as the *world coordinates*. So the ray is defined in the world coordinates. A *DBSraycast* processor was developed to reconstruct the CT and MR volumes. It accepts four input volumes and entry and exit buffers. The process function in the dbsraycast processor is called to render the two volumes. It starts by assigning texture units for the transfer functions and assigning texture units for the activated volume ports. The 3D textures are bind to the activated texture units. The vertex shader is a normal *pass through* vertex shader. The fragment shader takes from the processor the entry and exit color buffers through 2D sampler. The volume structure gives a complete description of the volume geometry i.e. the voxel dimensions, the voxel size, the volume cube size, number of color channels and the volume transformation matrix. The volume structure is defined in mod_sampler3d.frag and is included in dbsraycast.frag. The camera position in the world coordinates and light set up are defined in mod_shading.frag. This fragment file is also included in dbsraycast.frag. The ray algorithm is proceeding as a normal ray casting algorithm with modifications in querying the texture. Every ray sample position is given in the world coordinates. The world coordinates are converted to texture by the `worldtotex` method. The computing of the texture coordinate is started by multiplying the ray world position with the inverse of the matrix that transforms the proxy cube. Then it computes the position in the volume using the following equation 5.3:

$$\text{Texcoord} = ((\text{physicalPos} - \text{offset}) * \text{dataspacing}^{-1}) * \text{dataDimension}^{-1} \quad (5.3)$$

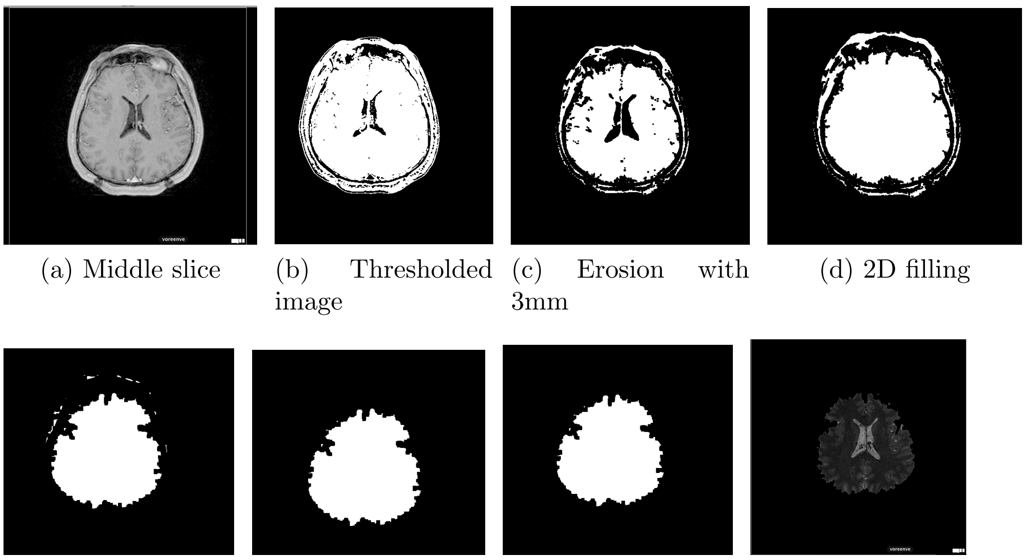
The resulting Texcoord from equation 5.3 is used as the texture coordinate parameter in `texture3D` or `texture` functions to query sampler3D in the fragment shader. For the single value volume, the gray value is saved in the alpha channel.

The *DBSraycast* assumes the first volume to be the MRI and the second volume to be the CT. This constraint was introduced for bone thresholding in the CT data, preventing acquiring color from CT and MRI voxels in the same position and the geometry manipulation with the bone. The CT volume can show some soft tissue during volume reconstruction. We are interesting in the brain voxels of the MRI and skull of the CT. The shader works under the assumption that the brain is segmented from the skull and the two volumes are registered. It checks if the MRI voxel is greater than zero in the current ray position, it will contribute in compositing in this step. Otherwise the CT voxel contributes in compositing in this ray step. The neurologist needs to understand the electric field distribution in the brain not in the skull. The developed shader is supported with a brain visualize mode, which discards the CT voxel from the starting of the lead until the end of the lead. This gives a better view of the electric field in the brain.

The compositing of the final color is done by Direct Volume Rendering to construct the final image.

5.3 Field Modelling

The electric field generated by the implanted lead is modelled by a Laplace equation. In equation 5.4, C represents the electric conductivity and V is the



(a) Middle slice (b) Thresholded image (c) Erosion with 3mm (d) 2D filling

(e) Erosion with 5mm (f) Conditional Dilation (g) 2D filling in every direction (h) Gray value assign

Figure 5.3: Brain segmentation steps

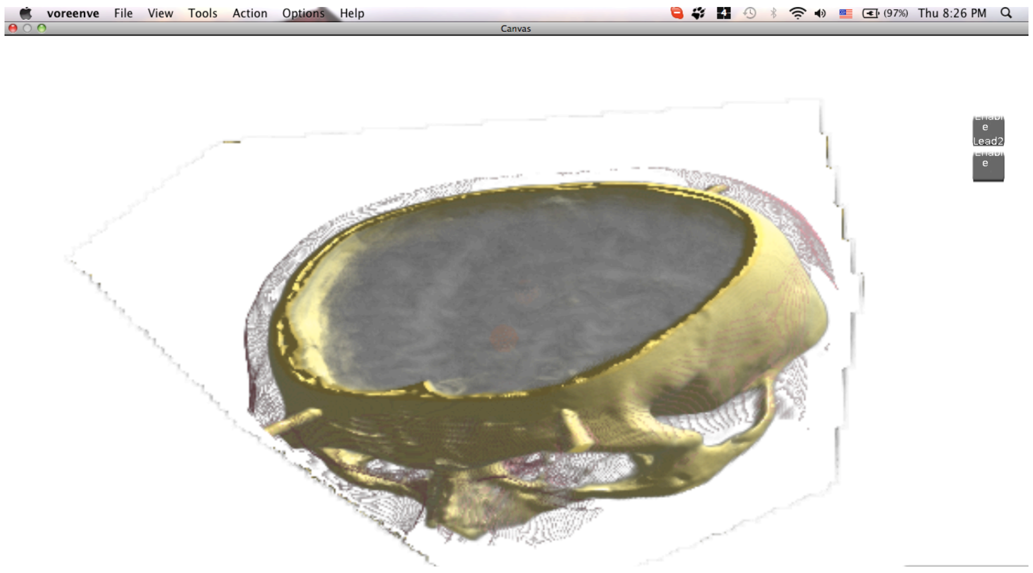


Figure 5.4: Volume reconstruction of CT and MRI

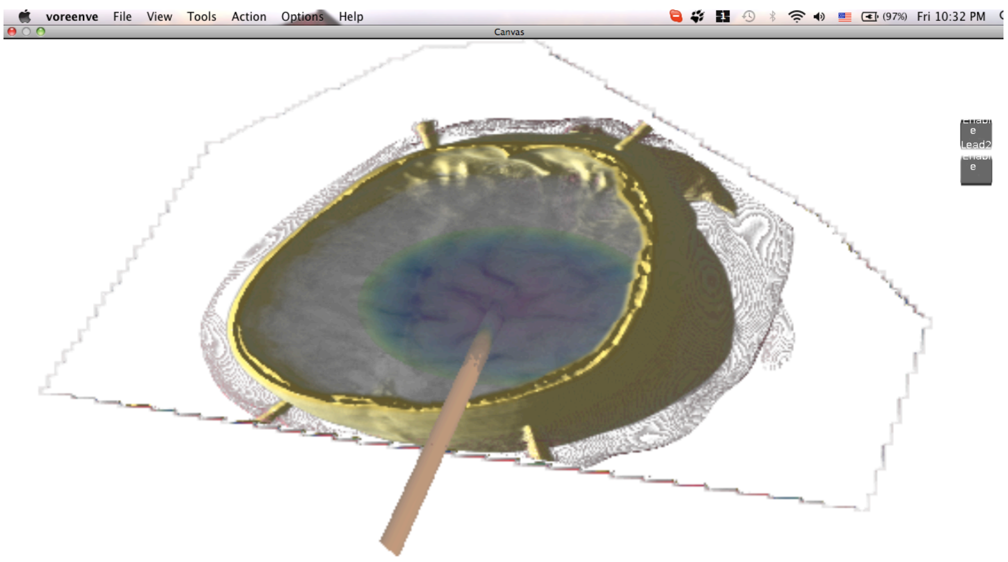


Figure 5.5: Spherical representation of the electric field

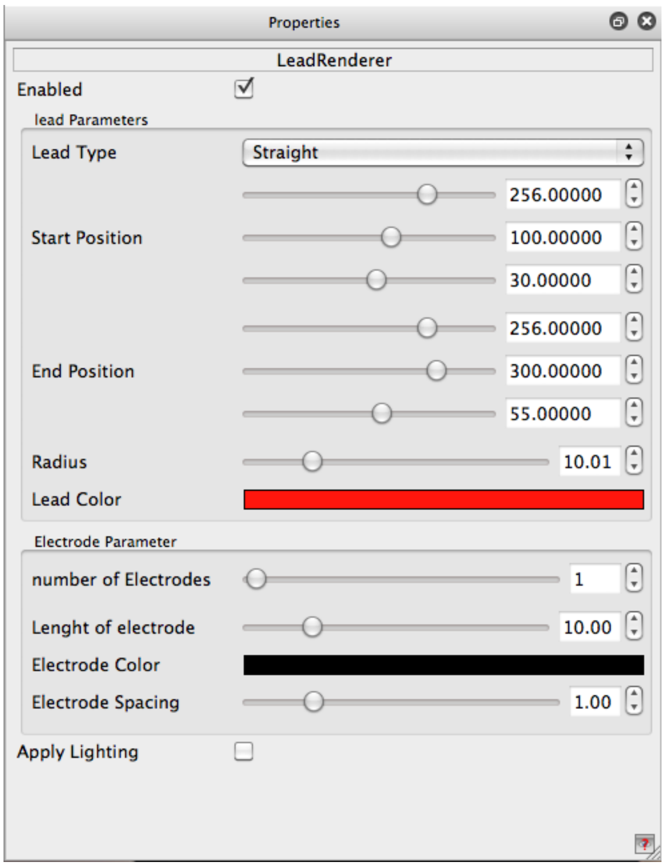
potential difference. Solving the differential equation is beyond the scope of this thesis. A full explanation about solving the Laplacian equation in homogeneous and non homogeneous medium can be found in [18]. The method of solving in [18] can be solved analytically in an homogeneous medium.

$$\nabla C \nabla V = 0 \tag{5.4}$$

In the early stage of the developing this thesis, we assumed that we will receive a parametric equation describing the potential distribution in the brain. Two geometric shapes were developed. These shapes are assigned the color from the transfer function representing the distance from the center of electrodes. A small processor was implemented that contains the parametric equation of an ellipse and a sphere. The equation type is specified by a string option in the *fieldequationprocessor*. The field equation string option is connected to the dbsraycast via linking property in the Voreen workspace. The shader is recompiled whenever the string option is changed. A string describing the equation, is compiled in the shader using the generateheader function in the dbsraycast processor.

5.4 Lead Modelling

The lead specification in this thesis was acquired from Medtronic model 3387 and 3389 manual [1]. The lead geometric properties can be described as a



Properties

LeadRenderer

Enabled ✓

lead Parameters

Lead Type	Straight
Start Position	256.00000
	100.00000
	30.00000
End Position	256.00000
	300.00000
	55.00000
Radius	10.01
Lead Color	🟥

Electrode Parameter

number of Electrodes	1
Lenght of electrode	10.00
Electrode Color	⬛
Electrode Spacing	1.00

Apply Lighting ☐

Figure 5.6: Lead specification interface

cylinder of specified starting and end points covered by a thin sheath. The lead is supported with four cylindrical electrodes that can be activated as mentioned in Section 1.1.1. If an electrode is activated, it will be added to geometric representation of the cylinder. See figure 5.6.

A *LeadRenderer* processor was developed to render the geometric specification specified in the GUI interface. The *LeadRenderer* is inherited from the *GeometryRendererBase*. It uses the input volume data to compute the mapping of the lead geometric specifications to the world coordinates. The output of this processor is sent to a *RenderProcessor* through the *GenericCoProcessor<PortGeometryRendererBase>*. It carries out the rendering to the activated framebuffer. The *LeadRenderer* is dedicated to model the lead and the electrodes in the Render function. The user provides the processor with the starting and ending points of the lead via the UI in the voxel coordinates. The process starts by multiplying the starting and end points by a transformation matrix to be transformed to the world coordinates. The

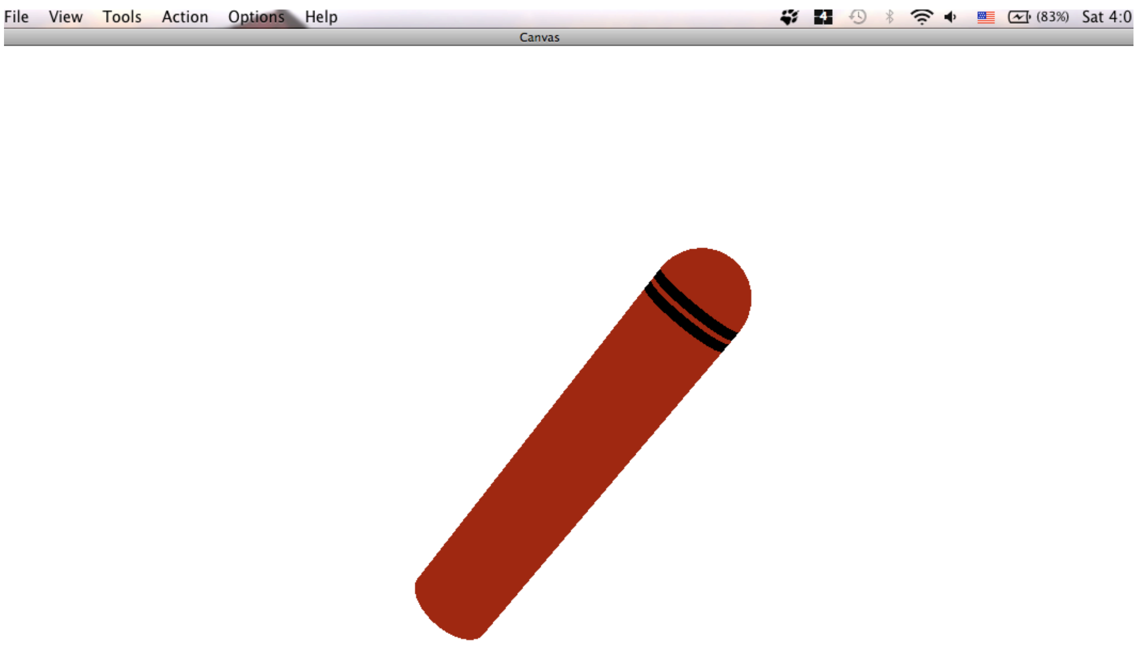


Figure 5.7: A lead with two activated electrodes

radius is multiplied by a scaling factor, which is the first element in the *voxel to world matrix*. In the next steps, all of the computation will be carried out in the world coordinates. The lead is rendered by the GLUCylinder. A rotation matrix is computed first, to rotate the cylinder axis to be aligned to the Z-axis. The lead is drawn first to the first electrode. The user specifies the electrode's spacing and electrode's length, which will be converted to the world coordinate in the same manner as the lead radius scaling. The electrode spacing is drawn by the lead quadric object. The lead end is a hemisphere with a radius equal to the radius of the lead. The hemisphere is drawn as triangle strips of the same color of lead. These triangle strips are compiled to a *glList*. This list is saved in the server side to be rendered. The list is rendered after applying the rotation matrix to align the axis of the lead with the Z-axis and transform the model to the center of the hemisphere. If the radius is changed, the compiled list will be changed. Another GLUObject is created for the electrode's characteristics. The electrode starts from the center of the hemisphere. The geometric characteristics and number of the activated electrodes can be adjusted from the UI. See figure 5.7

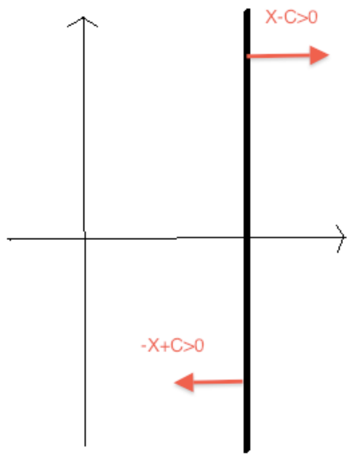


Figure 5.8: Clipping plane equation

5.5 Lead Slice Integration

The *SliceViewer* in Voreen shows three different slice views of the volume data; coronal, sagital and axial views. The lead geometry must be integrated in the slice view to convey complete information about the lead. A *SliceGeometryProcessor* was developed to accomplish the geometry view mixing. The *OpenGL* is supported by additional clipping planes to clip out the objects inside the viewing frustum. The *OpenGL* specification states that there should be at least six arbitrary clipping planes. The *SliceGeometryProcessor* activates two of the arbitrary planes to be used for clipping the geometry around the slice. Any transformation after activating the clipping planes will not effect the clipping plane but it will effect only the drawing object after this. The *OpenGL* will render only fragments that are in positive direction of the plane equation. See figure 5.8.

In the *SliceGeometryProcessor*, the plane equations type is specified according to the slice view mode. The slice view mode in this processor is controlled by the *SliceViewer* processor. The position of the clipping planes are set according to the view slices mode in the *SliceViewer* processor. The `process` function in the *SliceGeometryProcessor*, computes the relative position of the clipping planes with respect to the proxy cube, i.e. the world coordinates. The *SliceGeometryProcessor* sets the project mode to the orthographic mode and sets the camera viewing direction to be in the same direction of the slice alignment direction. The vertex shader used in the *SliceGeometryProcessor* is a normal pass through shader with the gl_ClipVertex

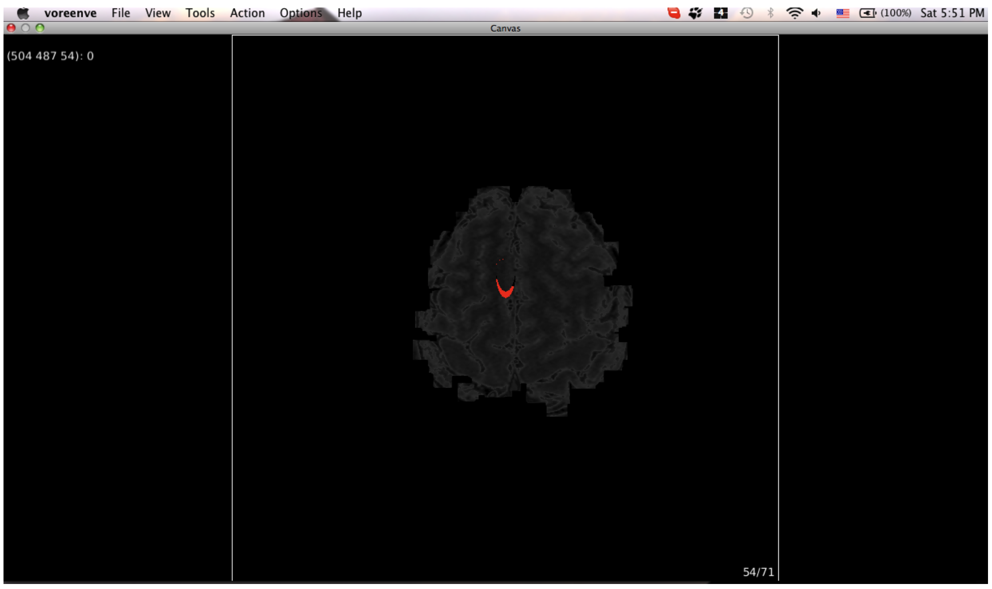


Figure 5.9: Axial view of MRI with the lead position in the slice

with a value of a gl_vertex multiplied by the gl_ModelView Matrix. Figures 5.9 and 5.10 shows the geometry rendering with a texture slice.

5.6 Lead Volume Rendering

This thesis is developed to be easily integrated with other frameworks to solve the Laplace equation 5.4. After investigating several mesh generators and some Partial Differential Equation (PDE) solvers, it is found that these frameworks are common in some steps. The PDE solver needs the volume to be meshed to be able to solve the PDE. In this module, the volume data is perturbed to label the leads and the brain. The module changes the voxels value in the volume that intersects with lead to predefined constant value. The *volumeelectrode* processor was implemented to achieve this.

The process starts by creating a new volume of the same type as the input volume. A similar UI systems for the one showed in figure 5.6 is used to get the geometrical description of the lead. The process enables the addition of four leads but it can be easily extended to add more leads. The lead's start and end points are given in the voxel coordinates and are transferred in the same manner as shown in Section 5.4. The process function investigates every voxel in the volume and checks if it is a lead candidate or not. The investigation starts by getting the voxel world coordinates position. Then, the voxel is projected to the lead's axis line. In order to compute the

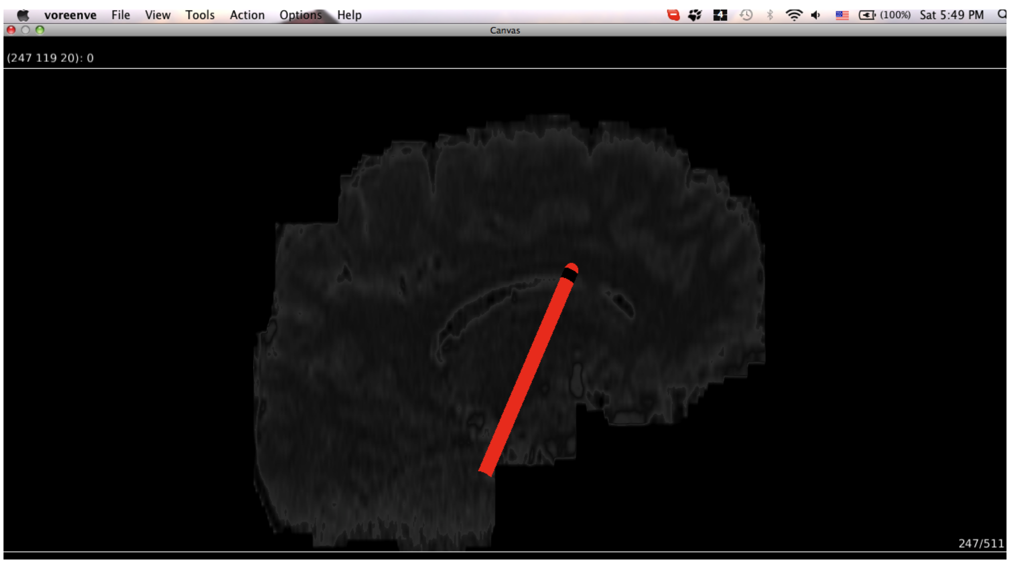


Figure 5.10: Sagital view with a lead position in the slice

projection of the point on the line, a line is constructed between a point on the axis line and voxel position. The length of the projected line should be zero, i.e. the line is perpendicular to the axis. The point that leads to the line projection can be described by parametric equation 5.5. See figure 5.11.

$$P = P1 + U(P2 - P1) \tag{5.5}$$

The parametric variable can be computed to make the length of the projected line equal to zero through the equation 5.6.

$$U = \frac{(P_0 - P_1) \cdot (P_2 - P_1)}{\| P_2 - P_1 \|^2} \tag{5.6}$$

The parametric variable should be in range [0, 1] in order to be on line of the axis. The accepted candidate is investigated to make sure if the length of this point to the voxel position is within the radius length. The accepted voxel is set its value in the output volume to predefined value and rejected candidate is set its value from the input. The other leads uses the same explained algorithm.

The output volume can be used by the computational geometry engine as shown in the next section . The mesh generation is explained in Chapter 4.

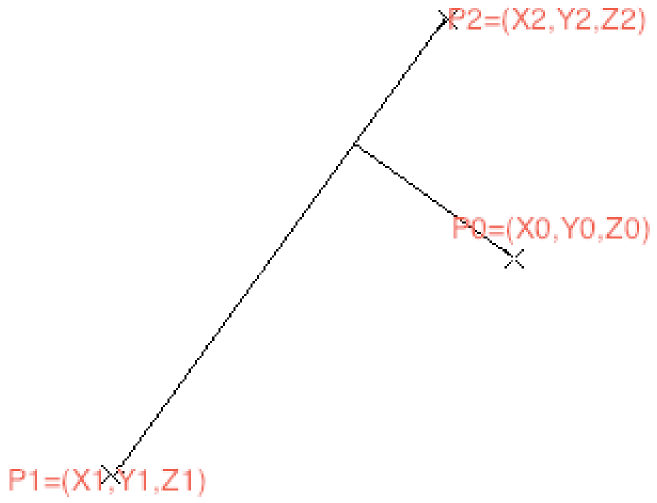


Figure 5.11: Line representation

5.7 Tetrahedra Generation

The Computational Geometry Algorithm Library *CGAL*[1] is an open source library implemented in C++. It provides data structures and algorithms for computational geometry. The *CGAL* hides the computational complexity by using generic interfaces for the data structure and functions. In the *VolumeElectrode* processor, a 3D mesh generator is added. The output from this processor is a .mesh file as well as the volume with labelled voxels to be rendered. The mesh engine used from the *CGAL* can be found in [23]. The engine is based on the delaunay triangulation algorithm. It is designed to guarantee that the user's mesh criteria will meet before termination.

The *VRNImage3* class was developed to construct the input domain in the format to be used by the *Mesh_Domain_3* class. The *Mesh_Domain_3* class needs the data description and one dimensional data pointer to be encapsulated in *_image* data structure.

The input data is passed to the *Mesh_Domain_3* class by the *VRNImage3* class. The input domain is assumed to be a pure 3D complex. A 3D complex can be defined as a set of points or surface patches that are pairwise disjoint and each boundary is union of the surface patches or the points. A pure means that a volume is described by these points or surface patches. The *Mesh_Domain_3* class tests the membership of every point to check whether or not it belongs to a certain domain. It also conveys the input data to the meshing algorithm *make_mesh_3*.

The meshing criteria controls the delaunay refinement process. It is specified through the *Mesh_criteria_3* class. The meshing criteria specifies the size and shape of the final tetrahedra and the triangles on the surface patches. The final mesh is controlled by the following mesh criteria:

1. **Cell-radius edge ratio**: the ratio between the tetrahedron's circumradius and the length of the shortest edge.

2. **Cell_size**: governs the size of the mesh of the resulted tetrahedra.

The resulted faces are controlled by

1. **Surface patch size and angle**.

2. **Surface patch distance**: controls the surface subdivision.

3. **Surface topology**: makes sure that every vertex of the surface is located on the surface patch.

[1]www.cgal.org

The 3D output is described in the data structure, *Mesh_complex_3_in_triangulation_3*. The output are 3D tetrahedra approximating the input domain.

Chapter 6

Results

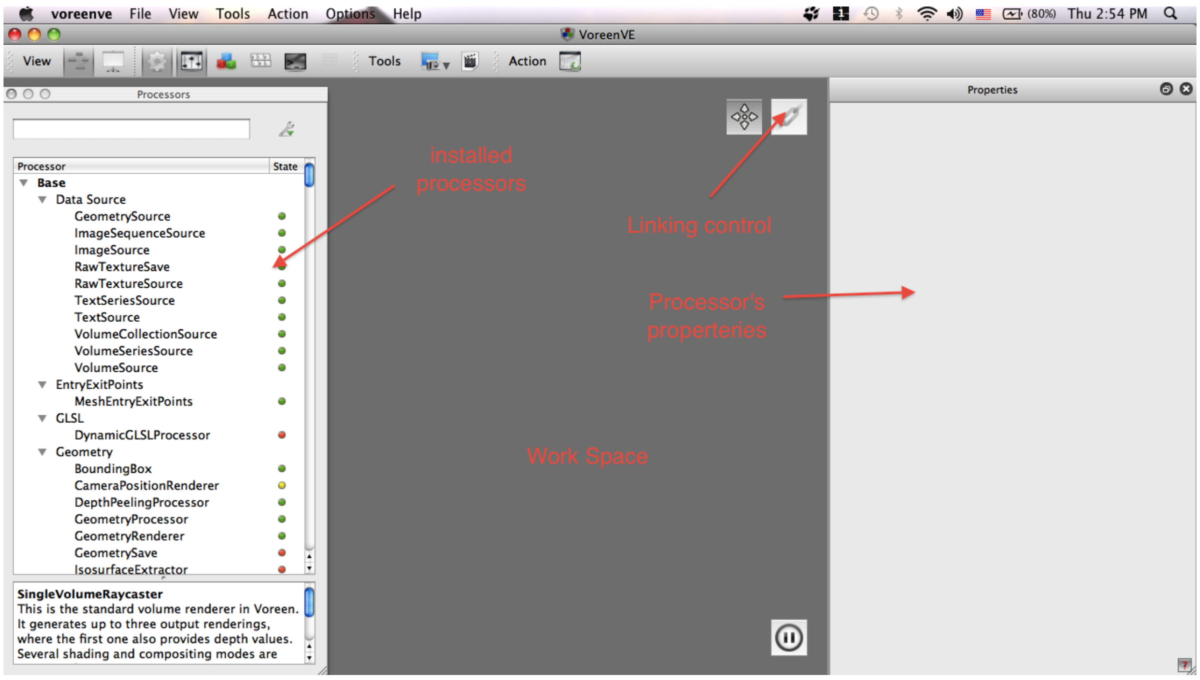
The aim of this thesis project is to develop some Voreen processors to allow neurologists to visualize the brain and design the case studies. Voreen is a quite mature software that is supported with enormous graphics and image processing processors. During this master thesis project, some of these processors are used to build the workspaces.

6.1 Full View Visualization

Voreen's user interface, allows the neurologist to connect the processors component in a highly abstract and easy way. The user does not get involved in any kind of mathematical or programming development. There are four important components in the main window as shown in figure 6.1.

The workspace component is used to add new processors and build the visualization pipeline. The processors can be found in the Processor window. The processors are dragged to the workspace. The dragged processors should be connected to other processors to contribute in the final output. Every processor has property interface to adjust its parameters through it. The property window is the only way to adjust the processor's parameters.

The built workspace uses all of the components explained in the previous chapters and also uses some of Voreen's processors to show the reconstructed volumes as shown in figure 6.2. First, two volume sources were added to load the MRI and CT volume. The MRI data was segmented using the Matlab script before loading. The contrast of the MRI data was stretched by using the *VolumeContrast* processor and the CT was transformed by using the *VolumeTransformation* processor. After this, the volumes were sent to the *multivolume* processor to construct the proxy geometry. The following processors are geometric processors until the ray casting stage. A stereotactic



Processor	State
Base	
Data Source	
GeometrySource	🟢
ImageSequenceSource	🟢
ImageSource	🟢
RawTextureSave	🟢
RawTextureSource	🟢
TextSeriesSource	🟢
TextSource	🟢
VolumeCollectionSource	🟢
VolumeSeriesSource	🟢
VolumeSource	🟢
EntryExitPoints	
MeshEntryExitPoints	🟢
GLSL	
DynamicGLSLProcessor	🔴
Geometry	
BoundingBox	🟢
CameraPositionRenderer	🟡
DepthPeelingProcessor	🟢
GeometryProcessor	🟢
GeometryRenderer	🟢
GeometrySave	🔴
IsosurfaceExtractor	🔴

SingleVolumeRaycaster
This is the standard volume renderer in Voreen. It generates up to three output renderings, where the first one also provides depth values. Several shading and compositing modes are

Figure 6.1: Voreen workspace

frame is a heavy metal containment. In order to remove these leads from the scene, the proxy geometry was clipped by using six *MeshClipping* processors. Another *MeshClipping* processor was added after this for arbitrary clipping. After clipping, the *MeshEntryExit* processor was used to compute the entry and exit points of the rays.

The two volume sources, as well as the two outputs from *MeshEntryExit* processors were connected to the *dbsraycast* processor. Finally, the reconstructed volume generated from *dbsraycast* processor was sent to the *QuadView* to be visualized.

The *QuadView* renders on the screen, four different views according to the inputs. The neurologists and physicists are interested in studying the electric field in the brain. The MR volume source was connected to the *CubeMeshGeometry* to generate a proxy geometry for the brain volume. The generated mesh was further processed with the *sliceExtract* processor to view certain region with arbitrary thickness. The *sliceExtract* was connected to the *MeshEntryExit* to generate the entry and exit points buffers. The entry and exit buffers and the brain volume were used by the *SingleVolumeRayCast* to be rendered.

The user can add as far as four leads for his studies to the generated volume. First, the physician adjusts the lead properties through the *Lead-*

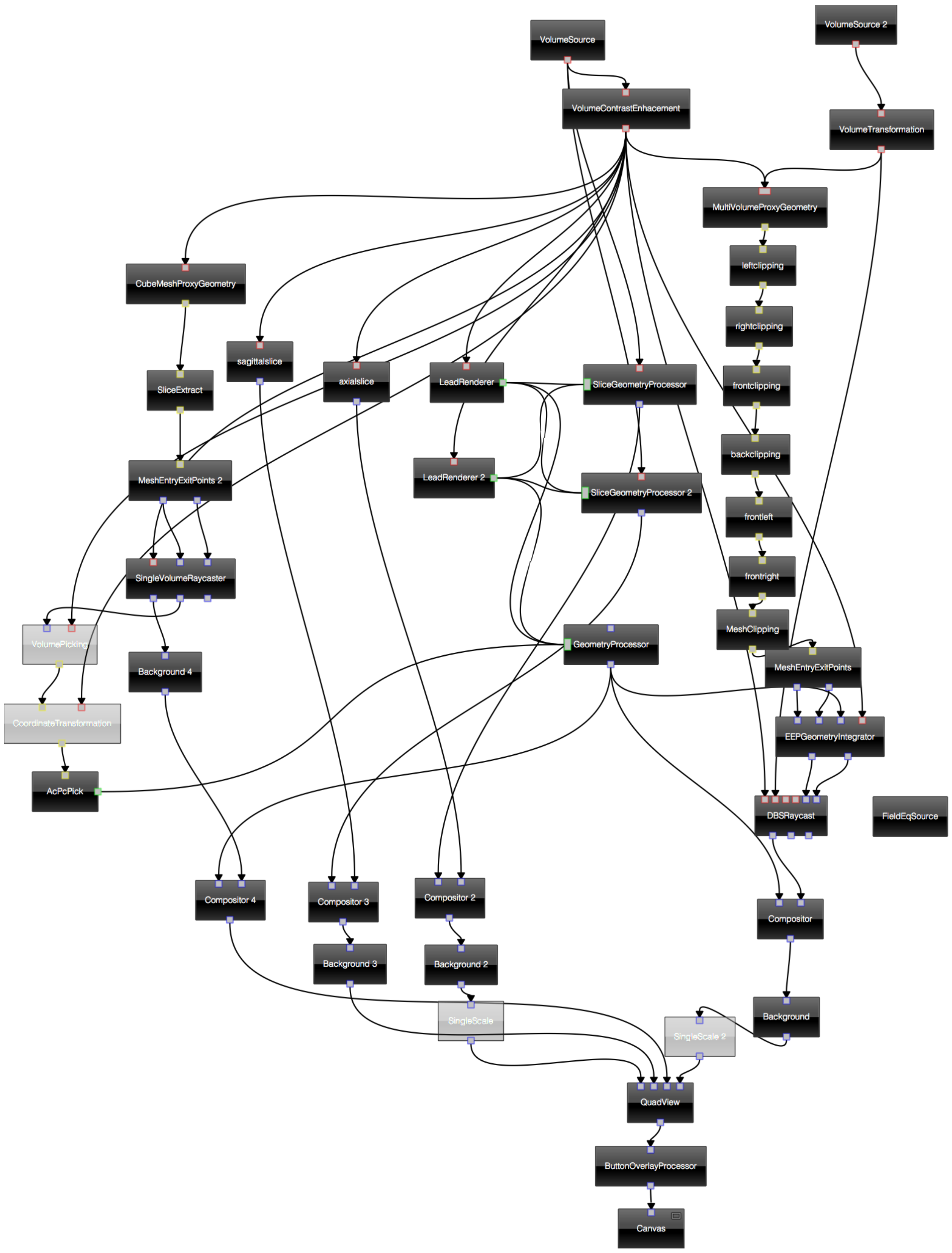


Figure 6.2: Network graph of the final view

Renderer's processor properties. The *LeadRenderer* was connected to *GeometryProcessor* to be rendered to a framebuffer. The generated image was added to the final constructed volume through the *compositor* processor. The final image was generated from the compositor and was sent to the viewer instead of the image generated from the ray caster processor. In order to consider the geometry contributions during reconstruction of the volume, the *EEPGeometryIntegrator* was used between the *MeshEntryExit* processor and the volumetric ray caster.

The last two views in the final quad viewer were two slice views. The output from the *SliceViewer*, was connected to the output view through a compositor. Instead of connecting the *LeadRenderer* to the *geometyprocessor*, it was connected to the *SliceGeometryProcessor*. Two *SliceGeometryProcessors* were added to the workspace for every slice extractor. The slice alignment and slice position properties in the *SliceGeometryProcessor* were controlled by the corresponding properties in the *SliceViewer* via linking property in the workspace as shown in figure 6.3. The output from the *SliceGeometry* processor contributed in the final slice view through the *Compositor*.

The output leads can be inhibited in the output through the use of *ButtonOverlay* processor in the output Canvas. Every *ButtonOverlay* can disable/enable the lead rendering by connecting the button property in the *ButtonOverlay* to the enable property in *LeadRenderer*.

The output from the *QuadView* processor was connected to the *Canvas* that shows the output volumes on the screen.

6.2 Meshing WorkSpace

There are not many mesh generators available for the research purpose in the medical imaging field. The implanted electrodes in the previous workspace must be meshed and boundary condition are applied to solve the Laplace equation 5.4 using FEM. It will be superior if the meshing can be done within the same *Voreen* framework.

The *VolumeElectrode* processor was developed to carry out the mesh generation and save the result to a text file. The mesh engine from *CGAL* [1] was added to this processor. The workspace for mesh generation can be explained as two pipelines; the visualization pipeline and mesh computation pipeline. The visualization pipeline is a normal pipeline, starting from loading the volume source that should be an unsigned char type volume. The binary volume mask generated from the segmentation of the MRI was used

[1]www.cgal.org

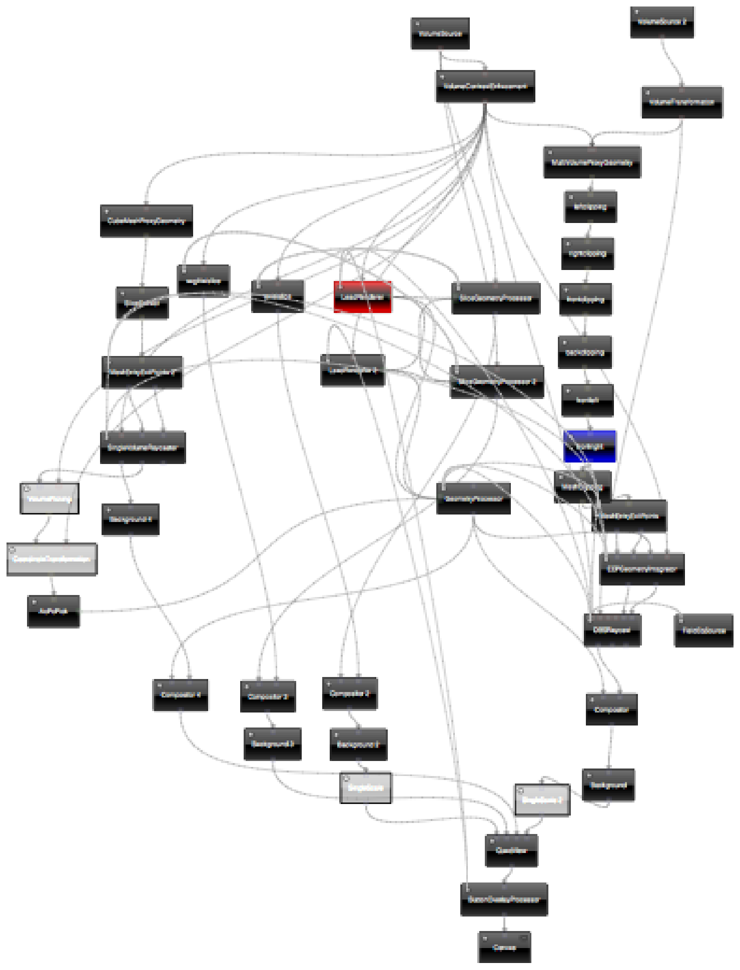


Figure 6.3: Linking property network

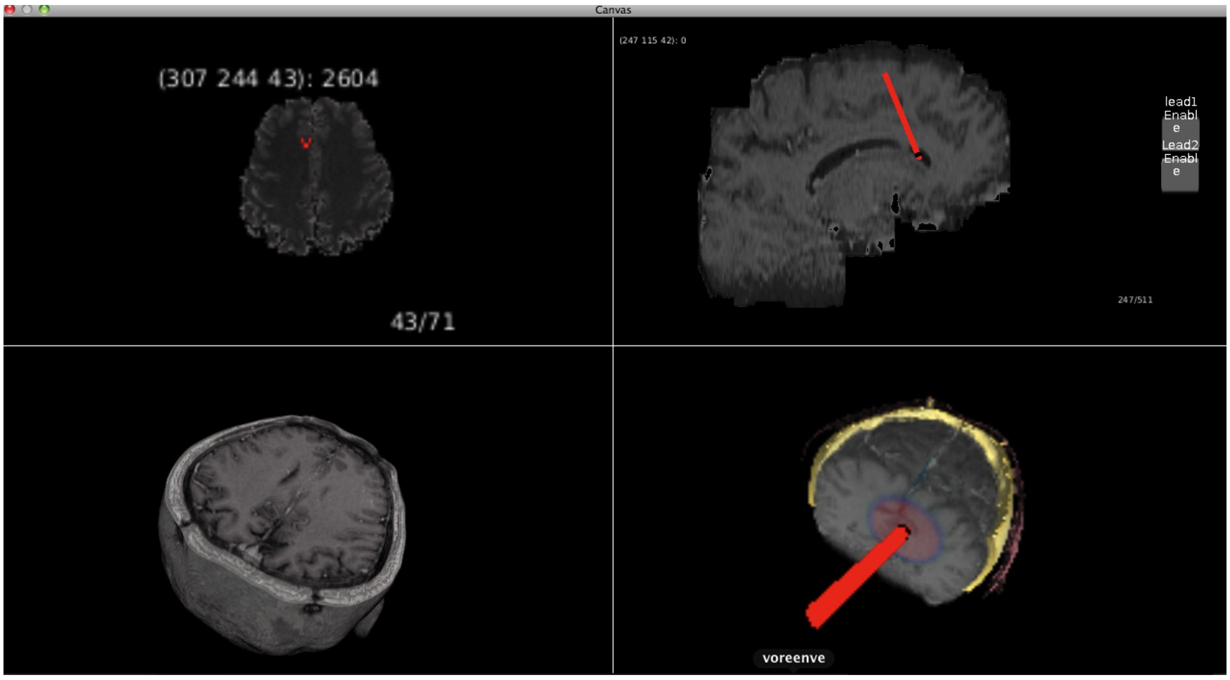


Figure 6.4: Final view. The top left view is axial view. The top right view is sagittal view. The bottom left view is MRI. The bottom right is reconstructed volume with a lead

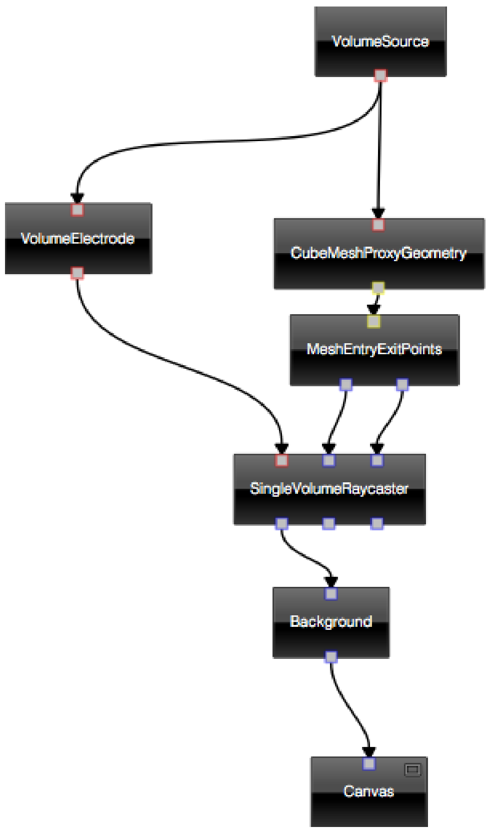


Figure 6.5: Mesh generation workspace

as an input instead of the MRI gray value data. The gray value MRI exhibits holes in the volume, i.e. voxels have value zero inside the volume. This makes the meshing processing a tedious task. Also, the physicians are interested in visualizing the potential distribution in every voxel of the brain even if it has a zero value. Then, the process continued by labelling the voxels that were in the leads areas. A proxy geometry was computed for this volume and was sent to a *SingleVolumeRaycast* processor with a volume generated from the *VolumeElectrode* processor as an input volume.

The mesh computation part was carried out by *CGAL* mesh generation engine encapsulated in the *VolumeElectrode* processor. These leads were enabled by the processor's property. The processor can add upto four leads in the volume. It can be easily extended to add more than four leads. The lead voxels were labelled with a value and the MRI voxels were also labelled but with a different value. The generation of mesh for the segmented MRI volume are shown in figures 6.6 and 6.7.

The mesh criteria values control the delaunay triangulation process. They

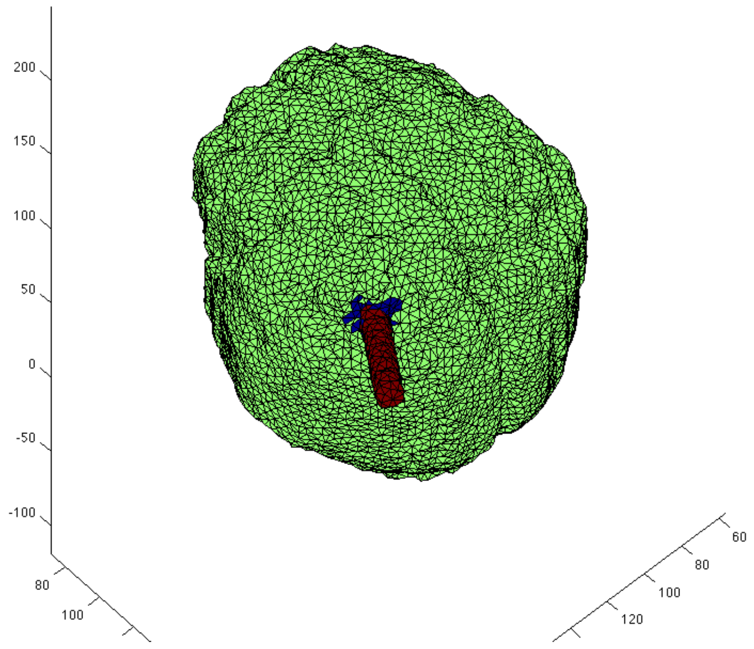


Figure 6.6: Mesh generation of the brain and one lead

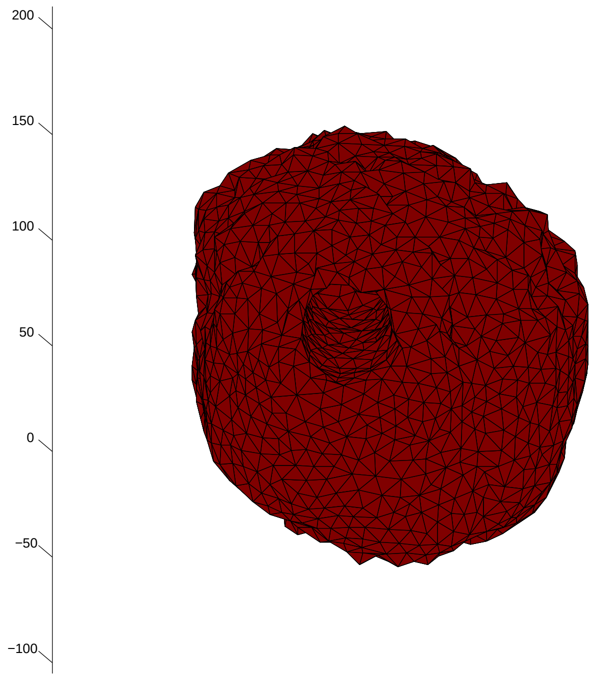


Figure 6.7: Mesh generation of the brain with lead's voxels set to zero

were chosen to be: 30 degrees for the facet angles, 3 for the surface area of the delaunay face, 2 for the maximum distance between the faces, 3 for the maximum ratio between the circumradius of a tetrahedron and its shortest edge and 4 for the maximum circumradius of the mesh.

The output tetrahedra are saved in *.mesh* file. This file is a text file with sections for the vertices, triangles, tetrahedra and labels. This file can be easily viewed by using: GeomView [2], GMSH, Matlab or Octave scripts [3]. This file can be processed by some scripts to be converted to Comsol. The free *FENICS project*[4] accepts the mesh file, process it to solve the PDE and converts it to a XML based file or a *VTK*[5] file.

[2]http://www.geomview.org/

[3]http://iso2mesh.sourceforge.net/cgi-bin/index.cgi

[4]http://fenicsproject.org/

[5]http://www.vtk.org/

Chapter 7

Conclusion

Thanks to Voreen's infrastructure and design, it was easy to develop and integrate our solution to Voreen's framework. The goal pursued in this thesis was to develop a frame work to facilitate the visualization of the brain structure and visualize the electric potential distribution in the brain. For this goal, some processors and shaders were developed to tune Voreen for visualizing the data from CT and MRI. A text file describing the mesh file was generated to facilitate the integration of the case studies to the PDE.

During this work, we tried to segment some important parts in the brain. The region growing technique failed to segment out the third ventricle in 1.5T MRI. The tiny structure of the third ventricle exhibited its edges in a blurry way. We tried to enhance the edge by developing an anisotropic diffusion filter. The edges were enhanced but the internal structure of the ventricle was not totally filled with CSF. There was white matter from surroundings in it. The segmentation can be modified by using semantic region growing [8] or active contouring algorithms.

We tried to develop an automatic Mid Sagittal Plane extraction algorithm in our work. The idea that we tried to develop depended on estimating the middle point in every axial slice in the fissure between the two hemisphere. This idea failed to show an accurate result due the sharp irregularity in the fissure.

7.1 Future Work

The project can be extended to include some processors for MSP extraction and brain organs segmentations.

The depth perceptual in the reconstructed volume in figure 5.4 needs to be modified using volume illumination techniques.

Voreen can be strengthened by adding visualization modules of geometric data for the Finite Element Methods.

Bibliography

[1] *DBS electrode specification model 3387, 3389.*

[2] *GPU Gems 3.* Nvidia.

[3] *Medical Imaging Physics, Fourth edition*, chapter 15. Willeys-Liss, 2002.

[4] *Medical Imaging Physics, Fourth edition*, chapter 8. Willeys-Liss, 2002.

[5] *Biomedical Technology and devices handbook*, chapter 8. CRC-Press, 2004.

[6] *Deep Brain Stimulation in Neurological and Psychiatric Disorders*, chapter 3. Hummana Press, 2008.

[7] *Image Processing, Analysis and Machine Vision*, chapter 5. Cengage Learning, 2008.

[8] *Image Processing, Analysis and Machine Vision*, chapter 10. Cengage Learning, 2008.

[9] *Principles and Practice of Stereotactic Radiosurgery.* Springer-Verlag, 2008.

[10] `http://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm149529.htm`, 2011.

[11] `http://www.voreen.org/`, 2011.

[12] Sorin Breit . Jrg B. Schulz . Alim-Louis Benabid. Deep brain stimulation. In *Springer-Verlag.*

[13] Gross C et al. Benabid AL, Pollak P. Acute and long-term effects of subthalamic nucleus stimulation in parkinsons disease. *StereotacticFunction Neurosurgery*, 1994.

[14] Louveau A Henry S-de Rougemont J. Benabid AL, Pollak P. Combined (thalamotomy and stimulation) stereotactic surgery of the vim thalamic nucleus for bilateral parkinson disease. *Applied Neurophysiology*, 1987.

[15] B. Delaunay. Sur la sphre vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793800, 1934.

[16] E.F.D'Azvedo. On optimal interpolation triangle incidence. *SLAM Journal on Scientific and Statistical Computing*, 1989.

[17] Klaus Engel. *Real-time volume graphics [Elektronisk resurs]*, chapter 4. A K Peters, Ltd., Wellesley, Mass., 2006.

[18] Cameron C. McIntyre et al. Electric field and simulating influence generated by deep brain stimulation of subthalamic nucleus. In *Clinical Neurophysiology.*

[19] Gerig et al. Nonlinear anisotropic filtering of MRI data. In *IEEE Transactions on Medical Imaging.*

[20] Speigel GildenBerg PL and Wycis. The early years. In *Stereotactic funct neurosugery.*

[21] Charles L. Lawson. Software for $c1$ surface interpolation. In John R. Rice, editor, *Mathematical Software 3*, pages 161 –194. 1977.

[22] Pietro Perona and Jitendra Malik. Scale space and edge detection using anisotropic diffusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[23] Stphane Tayeb Jane Tournois Pierre Alliez, Laurent Rineau and Mariette Yvinec. *3D Mesh Generation.* CGAL User and Reference Manual, cgal editorial board, 3.9 edition edition, 2011.

[24] Wycis HT Spiegel EA and LEE AJ. Stereotaxic apparatus for operations on the human brain. In *Science 106:349-50, 1947.*

[25] Henri Maitre Thierry Geraud, Isabelle Bloch. 3D Segmentation of Brain Structures from MR Images using Morphological Approaches. *Medical Image Analysis, Oxford University Press.*