## Thèse (Dissertation)

# "Topology simplification algorithm for the segmentation of medical scans / Algorithme de simplification topologique pour la segmentation d'images médicales volumétriques"

Jaume, Sylvain

**Abstract**

Magnetic Resonance Imaging, Computed Tomography, and other image modalities are routinely used to visualize a particular structure in the patient's body. The classification of the image region corresponding to this structure is called segmentation. For applications in Neuroscience, it is important for the segmentation of a brain scan to represent the boundary of the brain as a folded surface with no holes. However the segmentation of the brain generally exhibits many erroneous holes. Consequently we have developed an algorithm for automatically correcting holes in segmented medical scans while preserving the accuracy of the segmentation. Upon concepts of Discrete Topology, we remove the holes based on the smallest modification to the image. First we detect each hole with a front propagation and a Reeb graph. Then we search for a number of loops around the hole on the isosurface of the image. Finally we correct the hole in the image using the loop that minimizes the modification to the image. At each step we limit the size of the data in memory. With these contributions our algorithm [...]

## Référence bibliographique

Jaume, Sylvain. *Topology simplification algorithm for the segmentation of medical scans / Algorithme de simplification topologique pour la segmentation d'images médicales volumétriques*. Prom. : Macq, Benoît (2004)

**Université catholique de Louvain**
Faculté des Sciences Appliquées
LABORATOIRE DE TÉLÉCOMMUNICATIONS
ET TÉLÉDÉTECTION

Louvain-la-Neuve, Belgium

# Topology Simplification Algorithm for the Segmentation of Medical Scans

**Sylvain Jaume**

*In fulfillment of the requirements for the degree of*
*Docteur en Sciences Appliquées*

**Examination Committee:**

Benoît MACQ (UCL/TELE) - *Advisor*
Alan H. BARR (California Institute of Technology, USA)
Simon K. WARFIELD (Harvard Medical School, USA)
Jean-Philippe THIRAN (EPFL, Switzerland)
Luc VANDENDORPE (UCL/TELE) - *Chairman*
Christophe DE VLEESCHOUWER (UCL/TELE)

February 2004

*To Camille, Odile and Alice*

# Acknowledgements

I wish to express my gratitude to my supervisor, Professor Benoît Macq, who gave me the opportunity to carry out this study with significant freedom in my research directions. I particularly thank him for his trust and openmindness that allowed me to perform research in three different laboratories without losing academic nor personal track.

I would also like to thank Professor Simon K. Warfield from the Brigham and Women's Hospital at Harvard Medical School (USA) for his dedication to share with me his experience in medical applications and his kindness for introducing me to Harvard Medical School. I fully appreciated the advice and suggestions he gave me while I was searching for interesting applications for my research, and again while preparing research papers.

I am equally most grateful to Professor Alan H. Barr from the Computer Graphics group at the California Institute of Technology (USA) for his support and understanding throughout my work. Thanks to his insight, our discussions were invaluable in helping me make the right decisions for my research.

I am indebted to Professor Peter Schröder for giving me the opportunity of spending a year in his Computer Graphics laboratory, at the California Institute of Technology (USA). My interactions with him helped me realize the exciting and demanding aspects of being a professor. I especially recognize his enthusiasm and skills for driving projects and motivating his team.

Thanks, also, to the examination committee for their many interesting comments, which resulted in the considerable improvement of the manuscript. I particularly thank Professor Jean-Philippe Thiran from the ITS Laboratory at the EPFL (Switzerland) and Doctor Christophe De Vleeschouwer from the TELE Laboratory for their availability and challenging attitude towards the scientific groundings of the thesis.

I would also like to thank my fellow researchers and colleagues in the three laboratories that I visited for the pleasant atmosphere at work and at social times. I would like to make sure to thank Vincent Nicolas for the prompt and sympathetic help he gave me any time it was needed, and Matthieu Ferrant for introducing me to research.

Lastly this thesis would not have been possible without my friends Hugues and Craig dragging me away from my desk, nor my family caring and encouraging me at all times. To all of them, ”Thank you!”

The research reported in this thesis has been funded by the Belgian Fonds pour la Recherche pour l’Industrie et l’Agriculture, the Belgian Fonds National pour la Recherche Scientifique, and the MERCATOR grant from the Walloon Region of Belgium.

# Contents

# Abstract

*Magnetic Resonance Imaging, Computed Tomography, and other image modalities are routinely used to visualize a particular structure in the patient's body. The classification of the image region corresponding to this structure is called segmentation. For applications in Neuroscience, it is important for the segmentation of a brain scan to represent the boundary of the brain as a folded surface with no holes. However the segmentation of the brain generally exhibits many erroneous holes. Consequently we have developed an algorithm for automatically correcting holes in segmented medical scans while preserving the accuracy of the segmentation.*

*Upon concepts of Discrete Topology, we remove the holes based on the smallest modification to the image. First we detect each hole with a front propagation and a Reeb graph. Then we search for a number of loops around the hole on the isosurface of the image. Finally we correct the hole in the image using the loop that minimizes the modification to the image. At each step we limit the size of the data in memory. With these contributions our algorithm removes every hole in the image with high accuracy and low complexity even for images too large to fit into the main memory. To help doctors and scientists to obtain segmentations without holes, we have made our software publicly available at http://www.OpenTopology.org.*

**Keywords:** Topology, hole correction, Reeb graph, non-separating loops, connectivity, segmentation, isosurface, volumetric images, medical scans, medical image processing.

# Chapter 1

# Introduction

This introductory chapter will review the common clinical practices in Medical Imaging, and will point at their limitations. Different imaging modalities allow the visualization of the patient's body. First we will briefly present these modalities, and how the medical scans represent the human body. Second we will explain how these data are stored. Third we will describe the techniques used for the visualization of these volumetric images. Finally we will detail the limitations of the medical images.

## 1.1 Imaging modalities

Various imaging technologies, called modalities, are used for the visualization of the human body. The most common modalities are Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Tomography (PET), and Ultrasound (US). They provide different images based on the interaction of the scanning process with the human tissues. Therefore some modalities are more appropriate for some medical examinations than for others. Other factors in the choice of an imaging modality are the side effects for the patient, the procedure of the treatment, and the cost of the image acquisition.

Computed Tomography (CT) acquires images of the human anatomy with a high resolution. It uses X rays that propagates through the body of the patient. The recorded intensities cannot distinguish the soft tissues in the brain. Besides, a long exposition to X rays could cause cancer, and thus the duration of CT acquisition must be limited.

Magnetic Resonance Imaging (MRI) offer images of the human anatomy based on the application of a strong magnetic field. The MRI images have

a lower resolution than CT images, but they can distinguish some human tissues that CT cannot distinguish. The acquisition of MRI scans does not have major impact on the human cells, and thus the patient can undergo several exams without risk. It is the preferred modality for imaging the brain tissues.

*Figure 1.1* shows images created from a Magnetic Resonance Imaging scan of the head and a Computed Tomography scan of the abdomen. The left image is a plane through a volumetric MRI scan of a head. The convoluted shape of the brain tissues is visible inside the skull. The right image shows a plane through a CT scan of the abdomen. The white structures on the left and the right represent the pelvic bone. The black round region on the top of the image represents the bladder inflated with air.

Positron Emission Tomography (PET) is used for the visualization of the metabolism. A dose of radioactive glucose is injected into the body of the patient. The absorption of the glucose in the most active regions of the human body is observed with a PET camera. Since tumors are very active structures, they are particularly visible in the PET scan. Since they use of a radioactive doses, PET scans have a negative impact on the human cells. Besides, the resolution of the PET scan is lower than MRI and CT images.

Ultrasound images (US) take advantage of the Doppler effect of an ultrasound signal to create an image. US images have a low resolution, but can be acquired at a low cost and in real time. The acquisition volume describes a cone, but can be re-sampled along a regular 3D grid.

Every modality interacts with the human tissues to record intensities in the volume of the human body. The recorded intensities followed a regular sampling, and thus the scan represents a 3D grid of intensities. In this grid representation, the nodes are called voxels, for volume elements.

## 1.2 Representation of the medical scan

The scanner records the intensity values in the body of the patient according to a regular sampling. This sampling creates a grid of values. Every grid node is called a voxel.

To create an image from a medical scan, the intensities are generally mapped to gray values. Mapping the entire range of intensities to a gray scale from black to white does not always produce the most contrasted image. To highlight a given anatomical structure, the mapping can be limited to the range of intensities corresponding to the structure. The range of intensities in the scan can be split in different segments, and a different

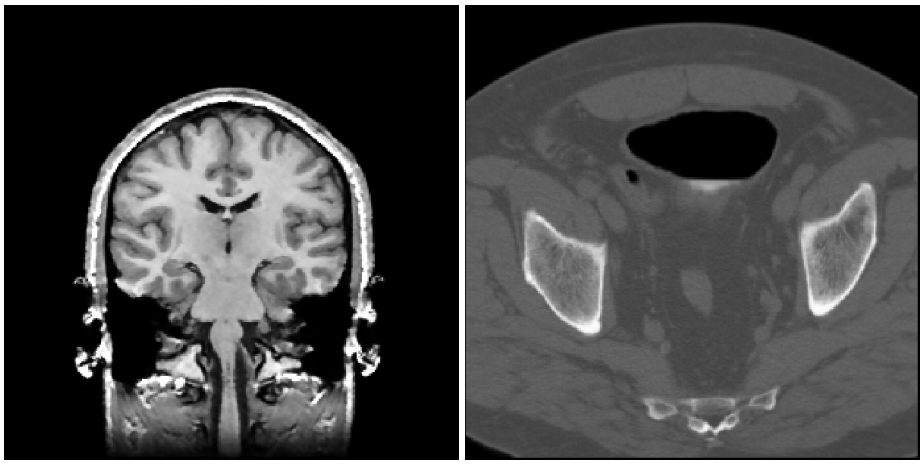

Figure 1.1: *Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) record different intensities for the different human tissues. The left image shows a cut through a MRI scan of the head. It indicates the complex structure described by the brain folds. The right image shows a cut through a CT scan of the abdomen. The two white regions are cuts through the pelvis bone, and the round black region on the top represents the bladder.*

mapping function applied for each segment.

*Figure 1.2* shows a sequence of planar images with a constant spacing. Six planar images extracted from a Magnetic Resonance Imaging scan of the head are displayed. The spacing between the images is exaggerated for clarity.

## 1.3 Visualization of the scan

Different techniques exist to visualize the structures visible in the scan. The limited quality of the image and the complex anatomy of the human body are two major challenges for the visualization. The most common visualization techniques include reviewing planar cuts through the volume, volume rendering, and isosurfacing.

### 1.3.1 Reviewing planar cuts

First, reviewing cuts made by parallel planes through the scanned volume is a common way of visualization of a medical scan. The scan can be

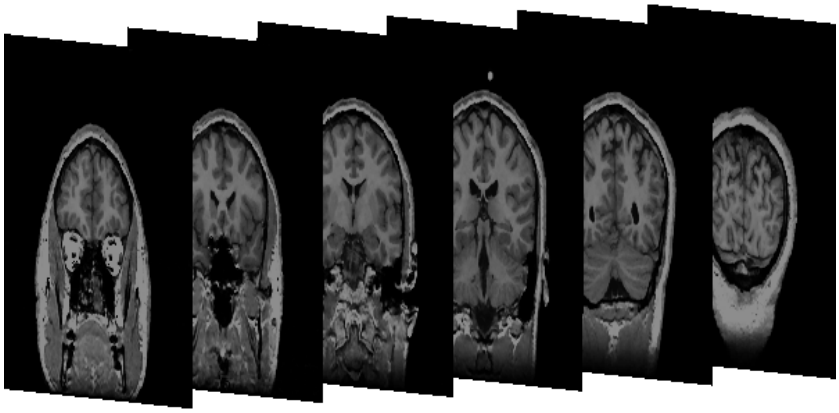

Figure 1.2: *A volumetric scan can be considered as a pile of planar images. The Figure shows six parallel cuts among the 124 planar images in the MRI scan of the head.*

considered as a sequence of equally spaced planar images. The direction orthogonal to the images is the direction of image acquisition. The doctor reviews one planar image after the other to virtually navigate through the body of the patient.

*Figure 1.3* shows six planar images selected from a Magnetic Resonance Imaging scan of the head of a patient. They reveal different regions of the brain. The top left image shows the contour of the skull and a portion of the brain. The top right image indicates that the white matter is surrounded by the folded gray matter in the brain. Within the white matter the two black spots are the ventricles. Below the brain is the cerebellum. On the second row the ventricles create a butterfly shape region in the center of the brain. The spine goes down from the brain in the left image. The ears are partially visible in the right image. Finally the bottom left image shows in black the air in the nose and in the mouth, while the bottom right image indicates the eyes as two spherical white regions.

As expressed from these images, the different tissues of the brain can easily be distinguished. However even with a strong anatomical background it is challenging to imagine the shape of the brain from planar cuts. Therefore reviewing planar cuts does not offer a satisfactory representation for some applications.

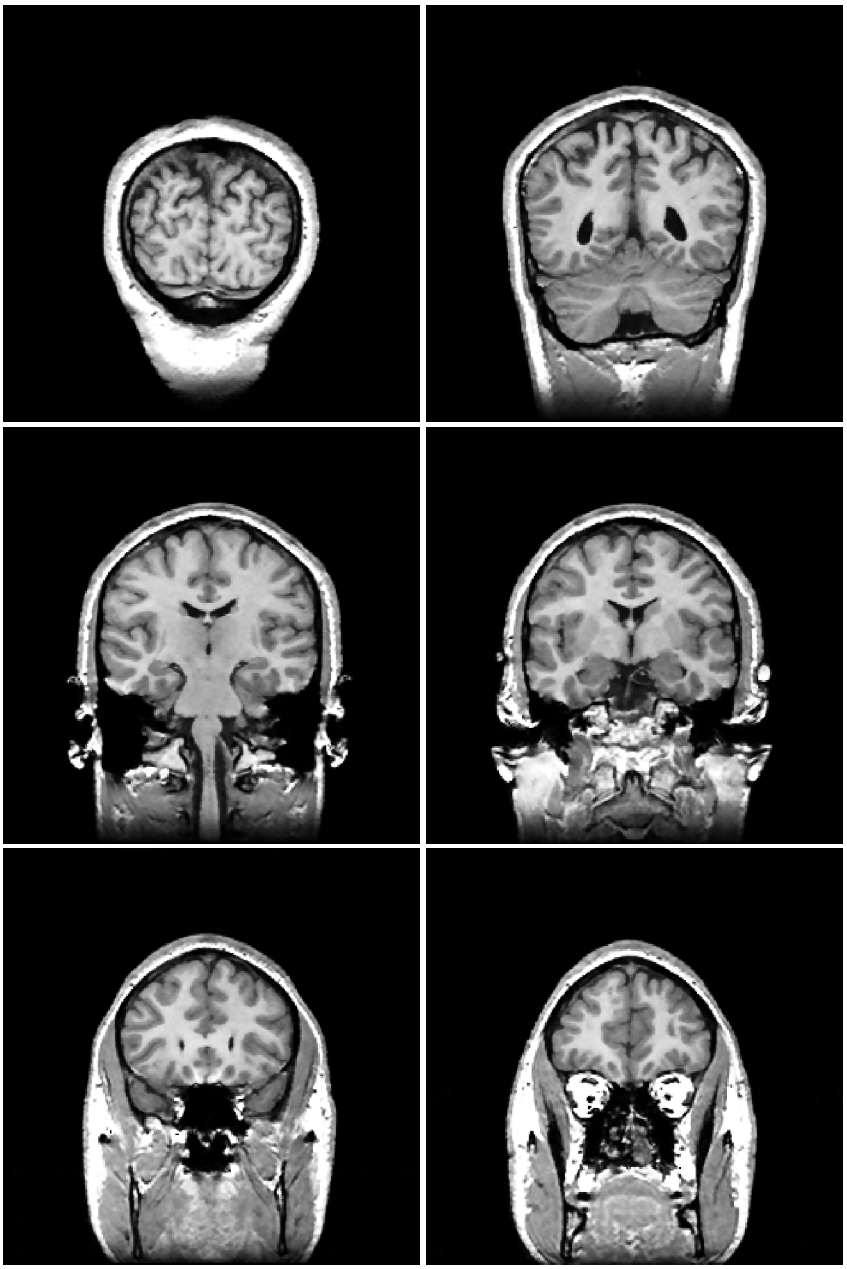

Figure 1.3: *Reviewing the planes of a medical scan allows to easily distinguish the human tissues. However it is challenging even for a trained radiologist to imagine the folded shape they describe in space. This Figure shows the plane number 20, 40, 60, 80, 90 and 100 from a MRI scan with 124 planes spaced by 1.5 mm. Every plane has 256 x 256 voxels spaced by 1 mm.*

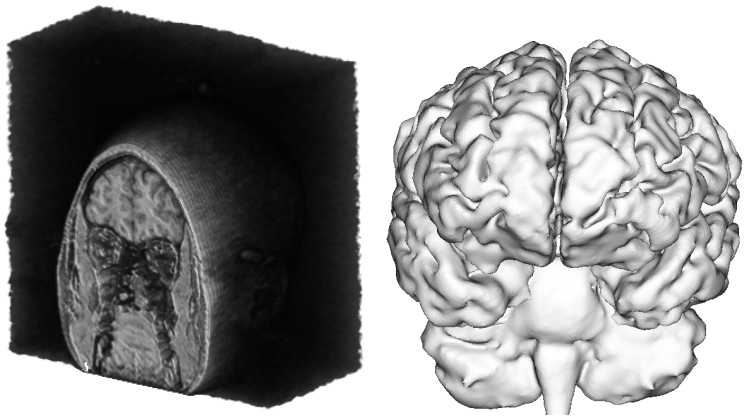

Figure 1.4: *Volume rendering (left) and surface rendering (right) are two techniques to visualize the 3D structure in a volumetric image. The left image shows a volume rendering of a MRI head scan. The volume is semi-transparent to see through the structure. The right image shows a surface rendering of a brain. This surface is a surface mesh made of 688,248 triangles, and is extracted from a segmented brain MRI scan with the Marching Cubes algorithm.*

### 1.3.2 Volume rendering

Second, volume rendering allows for the visualization of the entire volumetric scan. Conceptually, in volume rendering, a series of rays are propagated through the volume. The attenuation of the ray along its propagation direction creates a gray value, which is rendered on the screen to create an image. The sharpness of the image depends on the definition of two mapping functions. The first function maps the intensities of the scan to gray values, while the second maps the intensities to an opacity value. To mask out some regions, a zero opacity can be given to the corresponding intensities. It can be difficult to find the best parameters to visualize a given structure in the scan.

The left image in *Figure 1.4* is a volume rendering obtained by assigning a semi-transparent value for every voxel in an MRI head scan. We can thus see through the head image. The same head scan as is *Figure 1.3* was used to create this image.

### 1.3.3 Surface rendering

Third, isosurfacing allows for the visualization of the surface around different anatomical structures. The surface defines a region of equal intensity through the scan. This surface is called an isosurface. Like isoelevation lines describe closed contours on a geographic map, an isosurface describes the surface for a given intensity value.

The right image in *Figure 1.4* renders a surface mesh of the brain cortex. The process to obtain this mesh is the following. First the brain region in the head scan is segmented to create a binary image. Then the Marching Cubes algorithm is used to extract the surface mesh from a segmented image. This surface, composed of adjacent triangles, is finally displayed, and the shadows on the surface highlight the convoluted shape of the brain cortex.

The most popular technique for isosurface extraction is the Marching Cubes proposed by Lorensen and Cline [Lorensen and Cline(1987)]. The Marching Cubes operate on a logical cube made of eight voxels, and then 'marches' to the next. It creates a surface patch within the cube when the boundary surface of the structure intersects the cube edges. When all eight voxels are black, the cube lies entirely outside the structure, and no surface patch is created. Similarly, when all voxels are white, the cube lies inside the structure, and again no surface patch is created. For every other configuration, the Marching Cubes create vertices on the intersected edges, and triangulates the surface between the vertices. Since every cube voxel can be either white or black, $2^8 = 256$ configurations exist. However, due to symmetries and complementary cases, only 15 configurations are required in a table. Every triangle in a given configuration is defined with three indices that point to three edges of the cube. The doctor can rotate and zoom on the triangle mesh to visualize the structure of interest in real-time.

## 1.4 Limitations of medical scans

Every imaging modality offers an image with a limited quality. The most significant limitations are the resolution of the scanner, the limited time available for scanning the patient, the movement of the patient, the artifacts due to metal objects, and the radiation doses.

### 1.4.1 Resolution of the scanner

First, the scanner has a limited resolution. The imaged volume represents a grid with a limited spacing between the nodes. Anatomical features smaller than this spacing cannot be observed in the resulting image. When two points in the body of the patient are too close from each other, the recorded intensity is an average of the neighboring values. This problem is called the Partial Volume Effect.

### 1.4.2 Acquisition time

Second, for the comfort of the patient and for scanning a large number of patients, the duration of the image acquisition is limited. Many patients suffer from claustrophobia when they lie in the scanner. To reduce the discomfort of the patient, the scanning process is limited in time. Besides, scanners are an expensive equipment that can only be profitable if intensively operated. The use of a Magnetic Resonance Imaging scanner has a tight schedule, to scan one patient after another without interruption. The duration of an image acquisition limits the number of samples that can be acquired, and thus the quality of the image.

### 1.4.3 Movement of the patient

Third, the movement of the patient when the intensities are recorded has a blurring effect onto the resulting image. The most common movements occur along the arms and in the neck region. While movements of the arms and the neck can be reduced by binding the patient or by applying a mask, movements in the torso cannot be eliminated due to breathing. Every movement in a region of the body of the patient affects the sharpness of this region in the image.

### 1.4.4 Artifacts due to metal objects

Fourth, metal objects, such as brain clips, prostheses, or implants, cause artifacts in CT scans. These artifacts are visible as bright lines in the image originating from the location of the metal object. Due to the concentration of the line artifacts around the metal object, the neighborhood of the metal object is generally not visible in the image.

### 1.4.5 Radiation doses

Finally, the maximum radiation doses that can be tolerated limits the quality of CT scans. The resolution of a CT scan could be improved by increasing the acquisition time. However, a long exposition to X rays can cause cancer. Therefore, the acquisition time of a CT scan, and thus the image resolution, is limited to reduce the risk of cancer.

## 1.5 Conclusion

The present chapter has covered the common clinical practices in Medical Imaging. Medical scans allow the visualization of the patient's body by a regular sampling of this volume. The resulting grid of intensities can be mapped to gray values to create a volumetric image. The techniques to visualize the volumetric image are: reviewing the sequence of planar images, volume rendering, and surface rendering.

However these representations are limited for medical applications, such as Neuroscience. Therefore new methods are required as discussed in Chapter 2.

# Chapter 2

# Neuroscience Applications

The previous chapter showed that medical scans offer easy visualization through the human body. However, the representation of the brain in a head scan is often not sufficient for Neuroscience applications. This chapter will first briefly present the anatomy of the brain. Then the application of brain mapping, using functional and anatomical head scans, will be discussed. Third the segmentation of a brain scan will be defined. Finally, we will point at the problem of holes in brain segmented images.

## 2.1 Brain structures

Neuroscience tries to get a better understanding of how the brain works, develops, and adapts. The improved knowledge about the brain can help learning, brain surgery, and recovery after a brain accident. The main focus of Neuroscience is the brain functions, and their spatial organization within the brain. The basic functions of the brain are motion, vision, and speech.

The brain is a very variable organ both with respect to its anatomy and its functions. Anatomy and functions vary from one subject to another, and for a given subject they evolve during lifetime. Thanks to this variability, a patient can recover some functions affected by a brain accident. However, the basic brain functions, speech, vision, and motion, are believed to have a less variable location.

The brain cells are made of two main types of tissues: gray matter and white matter. Basically, the gray matter is a few millimeter thick layer at the periphery of the brain. It describes a very convoluted shape, and only a third of its area is visible when looking at the brain from the exterior.

The other two thirds are hidden within the brain folds. The white matter makes most of the inner volume of the brain. Its fibers connect the gray matter to the spine. It can be considered as the electrical wires between the gray matter and the rest of the human body.

Most of the brain functions are believed to be located in the gray matter, and to occupy a given area of the gray matter layer, called the cortex or cortical surface. When they study a function of the brain, the neuroscientists are particularly interested in locating the area of the gray matter that corresponds to this function. The cortical layer can be considered as a single sheet with many folds hiding most of its area. Unfolding a representation of the cortical surface onto a sphere or a plane would reveal the entire area of the cortex.

## 2.2 Anatomical and functional brain imaging

Anatomical and functional Magnetic Resonance Imaging (MRI) scans are particularly valuable to analyze the brain functions for a given patient, or to draw a map of the brain functions for a large population of subjects.

On the one hand, anatomical MRI (aMRI) provides a representation of the tissues of the brain. The white matter appears brighter than the gray matter. Different structures inside the brain are recognizable.

On the other hand, functional MRI (fMRI) takes a fast image of the brain when the subject experiences a given sensation, performs a mental task, or moves her or his fingers. The fMRI measures the increased blood flow during the activity of the subject based on the BOLD effect. The brain mapping is then achieved by associating this activity with the activated region in the brain.

The precise changes in the brain metabolism cannot be observed in a fMRI, but the effects of local increases in blood flow and oxygenation compared to the normal MRI mechanisms can be mapped as a change in the image intensity. The observed MRI images depend on the level of deoxygenation. The deoxygenated haemoglobin is a ”blood oxygenation level dependent” or ”BOLD” effect that can be observed by MRI at high magnetic fields.

*Figure 2.1* illustrates the mapping of a functional activation onto the anatomy of the brain. The left image shows as a color spot the activation of a brain function imaged with fMRI. This spot is superimposed on the corresponding cut through the aMRI scan to indicate the region of the brain responsible for the activated brain function. The right image shows

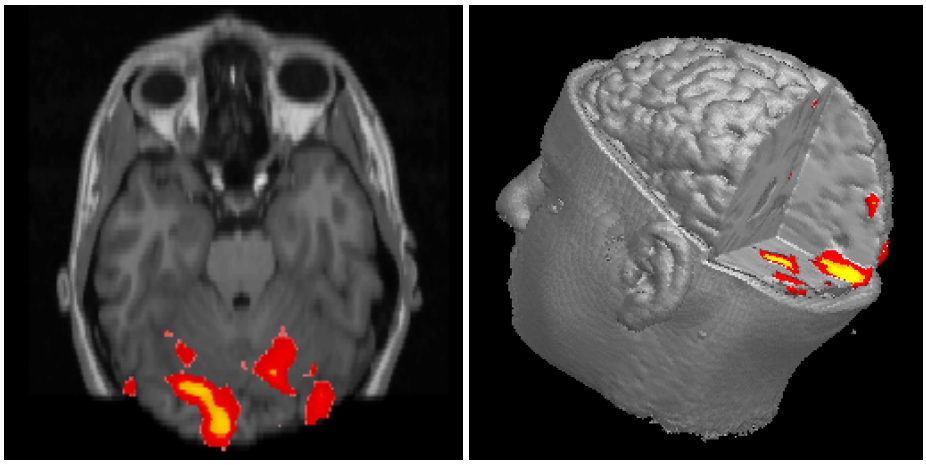

Figure 2.1: *Functional MRI (fMRI) represents the activated region of the brain when the subject performs a given task. The left image illustrates the activated region in color resulting from the fMRI, overlaid on the corresponding cut through the anatomical MRI. The right image shows a 3D rendering of the fMRI color spots. Courtesy Steve Smith.*

a rendering of the fMRI activation spot onto a rendering of the head of the subject. This locates the 3D position of the brain function in the brain volume. However, the superimposition of the fMRI activation onto the brain image does not accurately locate which brain fold is activated.

## 2.3 Segmentation

The segmentation of an anatomical structure within a scan is the classification of the region corresponding to this structure in the image. Segmentation is particularly important for quantitative analysis and visualization. In a binary segmentation, the object voxels, called foreground voxels, are typically represented in white, and the background voxels are represented in black.

In a segmented brain scan, the number of object voxels measures the volume of the brain. Besides, a surface representation of the brain cortex can be extracted from the brain segmentation, to improve the visualization. Neuroscience is particularly interested in the thin cortical layer of the brain. Since this thin layer is approximated as a surface, it is important for Neuroscience.

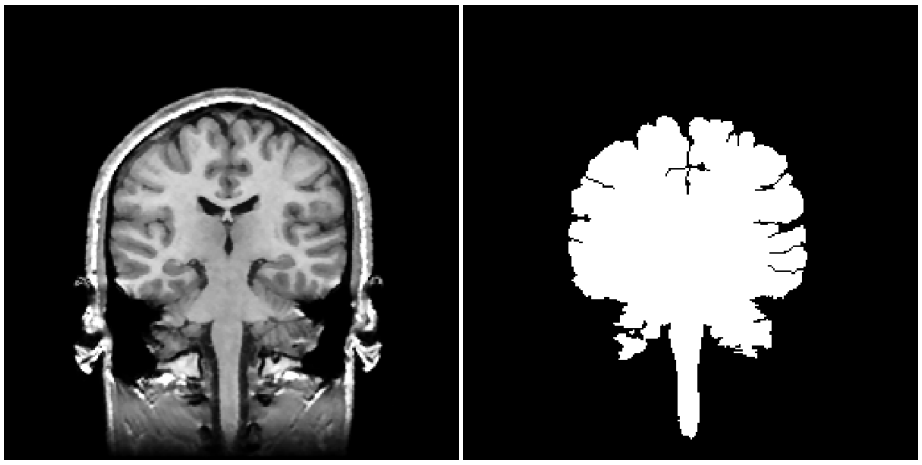

Figure 2.2: *The segmentation of a brain scan classifies the image into the brain region and the background. The left image shows a cut through a MRI brain scan while the right image shows the corresponding cut through the segmented image where the brain appears in white.*

The cortical surface can be represented by a surface mesh extracted from the segmented image. This surface, that goes between the object and the background voxels, is called an isosurface. To extract an isosurface from an image, one can use a deformable surface or an isosurface algorithm.

For a deformable surface, the mesh model can be made of triangles, splines, or non-uniform regular B-splines (NURBS). The nodes of the mesh evolve to fit the boundary between the foreground and the background voxels. Alternatively, an isosurface algorithm, such as the Marching Cubes, can be used to extract the cortical surface from the segmented image.

*Figure* 2.2 shows a plane through a head MRI scan (left image) and a plane through the brain segmentation of this scan (right image). The brain region is isolated from the background as a set of white voxels. This representation is useful for quantitative measurements on the brain volume, and for the visualization of the brain surface via an isosurface algorithm.

*Figure 2.3* illustrates the mapping of the brain cortical surface onto a sphere [Haker *et al.*(2000a)]. The isosurface of a brain segmentation is first color-coded based on curvature to highlight the brain folds, as shown in the left image. The vertices of the isosurface mesh are then mapped to

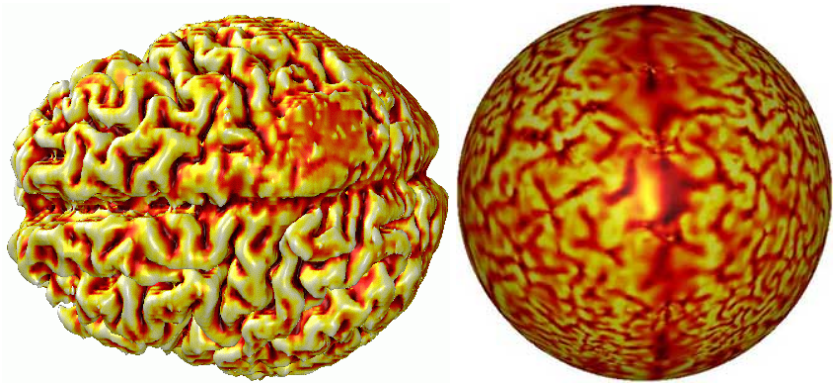

Figure 2.3: *The mapping of the brain surface onto a sphere reveals the entire area of the brain surface. The left image is an isosurface from a brain segmentation. The color-coding based on the surface curvature highlights the brain folds. The surface mapped to a sphere, shown in the right image, has the advantage that the inner region of the brain folds (red in the curvature color-coding), partially visible on the left image, is now entirely visible. Courtesy Steven Haker*

spherical coordinates, to create a spherical mapping of the surface. The color-coding of the input vertices is then transferred to the vertices on the sphere, as shown in the right image.

## 2.4 Holes in the segmentation

The exterior surface of many anatomical structures, such as the brain, have the shape of a folded surface without holes. However the segmentation of the brain in a medical scan generally do not present this property.

### 2.4.1 Causes of holes

Although most segmentation methods provide an accurate segmentation, several image limitations prevent the segmentation of a surface without holes due to wrong connections in the image. First when two brain folds are very close one to the other, they can appear connected in the scan due to the Partial Volume Effect. Second noisy voxels in the image can create a wrong connection between two folds. Third the misclassifications of a few voxels can occur in the segmentation and can also connect different brain folds in the segmented image.

The wrong connections in the image often create tunnels into the brain, or bridges between the top of two brain folds. These tunnels and bridges are two complementary representations of holes in the image: tunnels in the foreground are bridges in the background, and vice-versa.

### 2.4.2 Impact of holes

The holes in the segmented brain image are anatomically wrong. However, since they are generally small, they only have a small impact on the 3D surface representation. Nevertheless, they prevent the unfolding of the cortical surface. Indeed, two brain folds cannot be expanded when a bridge connect the top of the folds. Since the holes are small and have an arbitrary orientation, it can be difficult and time-consuming to locate them, and locally correct the image.

Due to the holes in the image, the use of spherical and flat maps is only marginal. For a brain segmentation withtout holes, the brain surface could be extracted and mapped onto a sphere or a plane when no opening or one opening is created into the surface. Since the entire area of the brain could be observed onto the spherical and the flat mapas, they could provide a much improved visualization and more accurate representation of the brain surface. However, since they generally require a manual intervention to remove the holes in the image, they only have a marginal use in Neuroscience. With automated detection and correction of holes in the image, spherical and flat maps could have a major impact in many medical applications.

*Figure 2.4* highlights a hole in a brain image, and on the brain surface. The misclassification of a few voxels in the segmented image results in a hole. The hole is visible on the cut through the segmented image (top left image), and in the close-up image (top right image). The bottom row shows the isosurface extracted from the brain image. It indicates that the hole creates a wrong connection between the hemispheres (close-up in the right image).

*Figure 2.5* illustrates that the holes in the brain image prevents the visualization of the entire brain cortical surface. To visualize the brain surface area hidden inside the brain folds, we inflate the isosurface extracted from the brain segmentation. Figure show this isosurface before and after inflation. The many holes hardly visible in the top left image create wrong connections that limit the inflation of the surface (top right). However when the holes in the segmented image are removed, the isosurface (bottom left) can be completely inflated (bottom right).

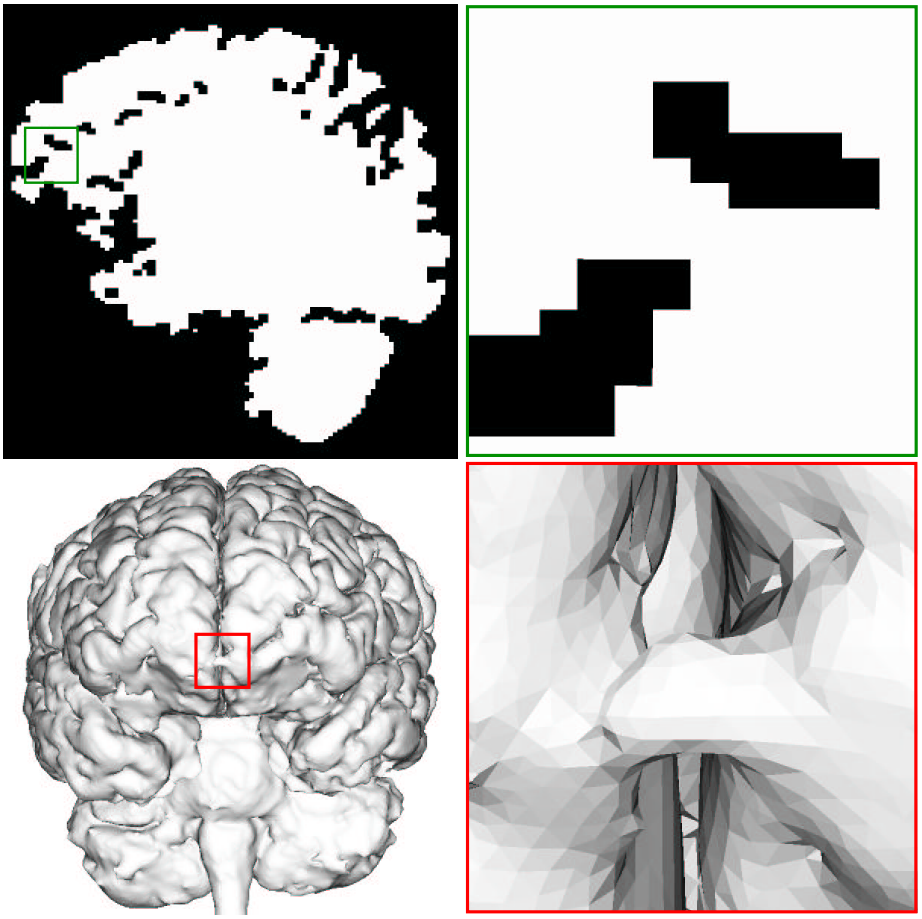

Figure 2.4: *The misclassification of a few voxels in a brain segmented image can create a hole close to the boundary of the brain. A cut through a brain segmented image is shown in the top left image, and the top right image zooms on the inside of the green square. The brain surface extracted from the segmented image is shown in the bottom left image. One can see the effect of the hole in the red square and in the close up in the right image: it creates a wrong connection between the left and right hemispheres.*

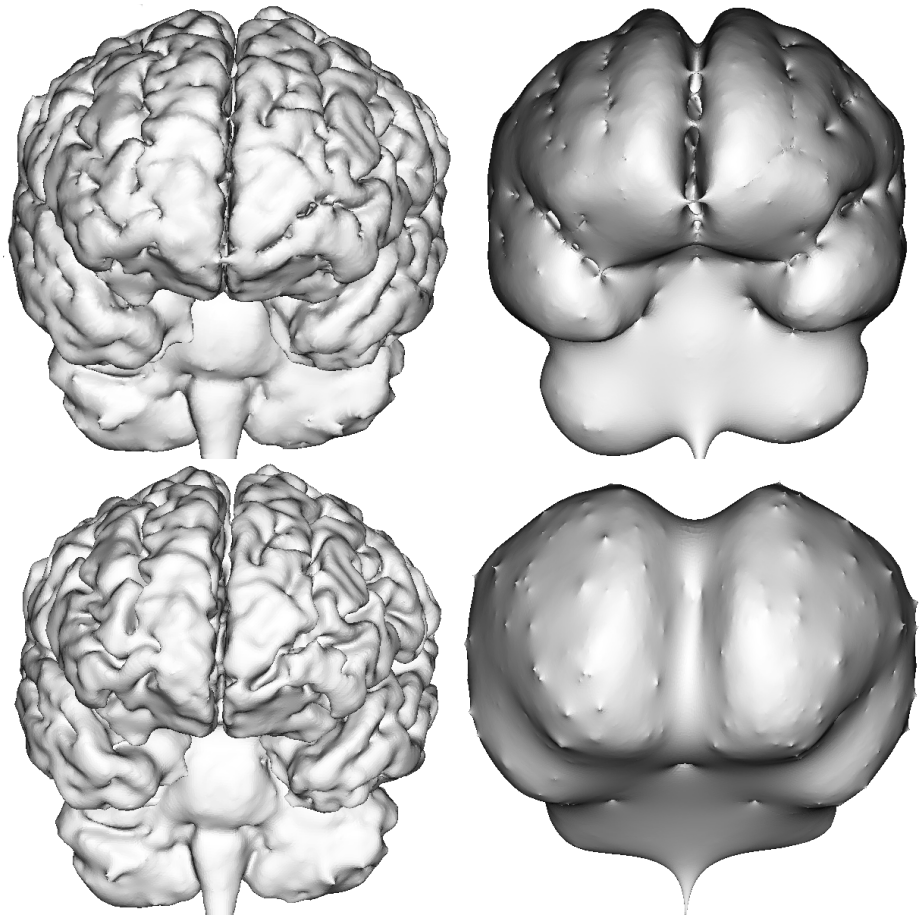

Figure 2.5: *The holes in the brain image prevent the visualization of the entire brain cortical surface by inflation of the isosurface. The isosurface extracted from the brain segmentation (top left image) exhibits a large number of hardly visible holes, that connect the top of some brain folds. Therefore the inflation of the isosurface (top right image) cannot unfold these brain folds, and the surface inside is not visible. However, once the holes are removed from the image, the resulting isosurface (bottom left image) can be inflated to reveal the entire brain surface (bottom right image).*

## 2.5 Conclusion

This chapter has defined the segmentation of a brain scan as the classification of the image into brain and background. Although the exterior surface of the brain describes a single folded sheet with no holes, most segmentation methods yield a segmentation that is very accurate but that exhibits numerous holes. The holes in the segmentation affect many Neuroscience applications such as mapping the human functions onto the brain surface.

The holes, due to the Partial Volume Effect, noise or voxel misclassifications, are created by wrong connections between the sides of some brain folds, or between the cavities inside the folds. These small holes into the background or into the brain are difficult to locate in the convoluted shape of the brain.

The goal of this thesis is to develop an algorithm to automatically correct the holes in segmented brain scans. The next chapter will define the terminology of Topology needed to understand our proposed algorithm.

# Chapter 3

# Theory of Topology

In the previous chapter, we noted that the brain cortical surface is a single folded sheet, but that its representation in a segmented brain scan generally exhibits a large number of holes. The present chapter sets the terminology we need to develop our hole removal algorithm. We will first introduce the theory of Topology, or the study of holes. We will then explain how the concepts of Continuous Topology can be adapted to the discrete setting of 3D images.

## 3.1 Continuous Topology

In this section, we provide a general definition of Topology, and present the concept of non-separating loops that we will largely used for the development of our algorithm.

### 3.1.1 Definition

Topology is the branch of Mathematics that study the properties of objects which are preserved through deformations, such as twisting, and stretching. However tearing is not allowed [Massey(1967)], [Munkres(2000)]. Therefore, new holes cannot be created in an object. Practically, Topology can be defined as the study of holes in geometric objects.

For Topology, two objects with the same number of holes are equivalent, or *homeomorphic*. A ball can be deformed into a Teddy bear, but it cannot be deformed into a donut since a hole into the ball would have to

be created. Similarly, a donut is equivalent to a cup since we can deform the hole of the donut into the hole in the handle of the cup.

Topology is only interested in the number of holes, not in their spatial location. Hence, it must be complemented with geometric tools to find the position of the holes.

### 3.1.2 Non-separating loops

A concept of Topology that we will extensively use in this dissertation is the non-separating loops. We will explain this concept for two geometric object: a sphere and a torus.

First we consider a sphere and draw a closed contour on its surface. We thus create two patches for this surface: one inside the contour, one outside the contour. The closed contour, or loop, separates the sphere surface into two patches.

Second we consider a torus, obtained by the revolution of a small circle around a large circle. We then cut the surface of the torus along the largest diameter. This loop does not separate the surface of the torus into two patches. When we cut a second time this surface along the smallest diameter, we obtained a band, and the surface is still made of a single patch.

The above example has shown that for every hole in an object, its surface can be cut with two loops that do not separate the surface. These loops are then called non-separating loops.

The non-separating loops of a hole indicate a method of topology simplification, i.e. to remove the holes. As shown in *Figure 3.1*, we can either fill or empty the area inside a non-separating loop to close the hole or open the hole respectively. In the case of the torus, the loop with the orientation of the largest diameter must be filled, while the other loop must be emptied.

Since both loops simplify the topology of the object, a decision must be made between using one loop or the other. The user can decide based on the desired appearance of simplified object. Alternatively, we could use a criterion that minimize the volumetric modification to the object. For such a criterion, it makes sense to fill a fat torus, and to empty the side of thin torus.

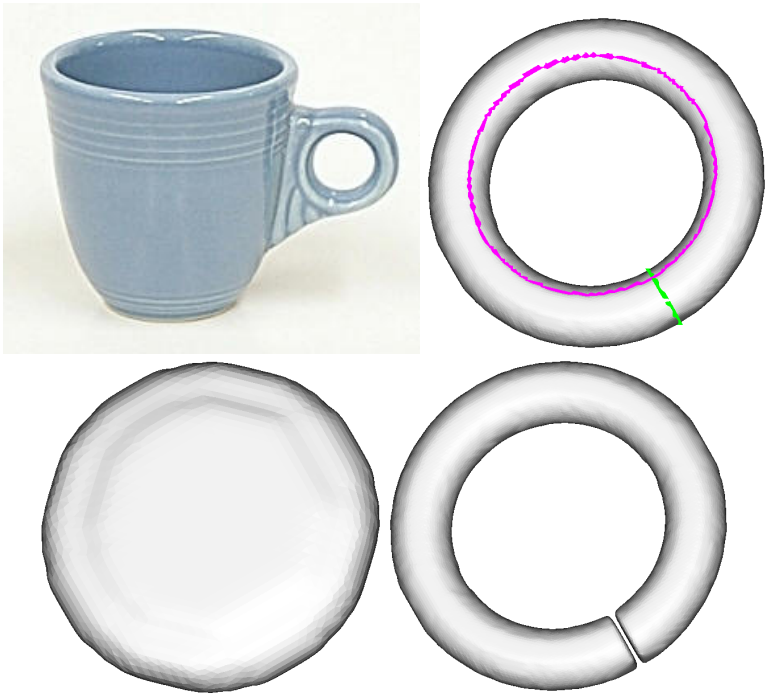

Figure 3.1: *A cup and a torus are topologically equivalent, or homeomorphic (top row). We can define two non-separating loops around the hole in the torus. These loops cross one another, and are shown in pink and green in the top right image. To simplify the topology of the torus, we can either fill the volume inside the pink loop (bottom left), or empty the volume inside the green loop (bottom right).*

## 3.2 Discrete representation of a volume and a surface

To adapt the continuous notions of Topology to the discrete setting of volumetric images, we need a representation of the object as a set of connected elements. In this section, we describe the dual representations of a set of volume elements, and a set of surface elements.

### 3.2.1 Volume representation

In a discrete volume representation, the object is represented as a set of foreground voxels in the 3D grid of the image. We consider the volume as a grid of cubic voxels, and follow the conventional definition of adjacency to consider three types of connectivities: 6-, 8-, and 26-connectivities. For instance, two cubic voxels are 6-adjacent if they share a common face. Similarly, two voxels are 18-adjacent if they share a face or an edge. Finally, two voxels are 26-adjacent if they share a face, an edge, or a vertex. The number 6, 18, and 26 correspond to the number of adjacent voxels for every connectivity rule.

To avoid topological ambiguities, different connectivities must be defined for the foreground and the background. Only the following pairs of the foreground and background connectivities are compatible: {6,26}, {6,18}, {18,6} and {26,6}.

### 3.2.2 Surface representation

Similarly, we can use a surface representation of the object to define the topology of the object. The surface representation describes the boundary surface of the object, i.e. the isosurface between the set of foreground voxels and the set of background voxels. The isosurface can be represented with splines, NURBS, or polygon faces. These representations can all be converted to a mesh of triangle faces. This surface mesh is made of faces, edges, and vertices. Two adjacent triangles share one edge. The triangles that share one vertex can be mapped to a disk or a half disk in case of a vertex on the boundary of the surface. The three edges of every triangle have a given orientation to define a coherent direction of the normal on the triangle mesh.

To guarantee the duality between the volume representation and the surface representation, the same connectivity rule must be used. An isosurface algorithm, such as the Marching Cubes, can extract the isosurface from the volume based on the given connectivity rule. The Marching

Cubes consider the eight adjacent voxels at the corners of a logical cube in the image. If some voxels belong to the foreground and some others to the background, the isosurface intersect this logical cube. Therefore the algorithm triangulates the section of the isosurface that crosses this cube. Then it 'marches' to the next logical cube. Finally the set of triangles tile the entire isosurface in the volume.

### 3.2.3 Holes in the volume and in the surface

A hole in the volume creates a bridge made of foreground voxels, or a tunnel made of background voxels. The same phenomenon applies to the surface representation, where only the boundary surface of the bridge or the tunnel is represented.

## 3.3 Discrete Topology

In this section, we describe the computation of the number of holes for a discrete representation. Then we present how we can take advantage of the discrete setting to approximately locate every hole in the image.

### 3.3.1 Euler characteristic

The Euler characteristic computes the number of holes in the image based on the number of elements in the representation of the object. Equation 3.1 shows the expression of the Euler characteristic $\chi$ for a volume representation. We consider every voxel as a small cube (not to confused with the logical cubes of the Marching Cubes). Therefore every voxel has a number of faces, edges, and vertices that could be shared with adjacent voxels. In the expression, $C$ is the number of foreground voxels, $F$, $E$, and $V$ are respectively the number of faces, edges, and vertices for these cubic voxels.

$$\chi = -C + F - E + V \tag{3.1}$$

Equation 3.2 shows the expression of the Euler characteristic for a surface representation. The symbols $F$, $E$, and $V$ are the number of faces, edges, and vertices in the surface mesh, and $H$ is the number of boundaries of the surface.

$$\chi = F - E + V + H \tag{3.2}$$

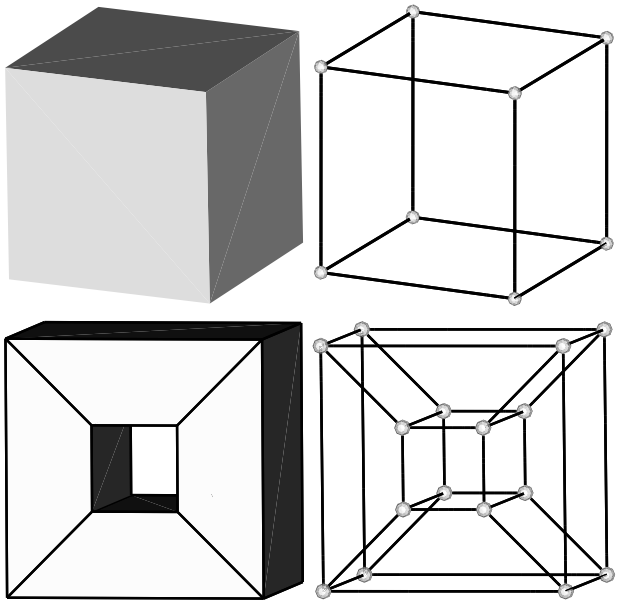

Figure 3.2: *The genus of a discrete surface, i.e. its number of holes, can be computed using its Euler characteristic. The top row shows a cube and its wireframe representation. Since a cube has F=6 faces, V=8 vertices, and E=12 edges, its Euler characteristic* $\chi$ *= 6 - 12 + 8 = 2, and its genus* $g = \frac{2-2}{2} = 0$*. The bottom image shows a cube with a hole and its wireframe representation. This surface has F=16 faces, V=16 vertices, and E=32 edges. The Euler characteristic* $\chi$ *= 16 - 32 + 16 = 0, and the genus* $g = \frac{2-0}{2} = 1$*, indicating that this surface has one hole.*

We can derive from the Euler characteristic the number of holes in the image, i.e. its genus $g$. Equation 3.3 shows this expression, where $K$ is the number of connected components in the image.

$$g = \frac{2K - \chi}{2} \tag{3.3}$$

*Figure 3.2* illustrates the computation of the number of holes based on the Euler characteristic for two surfaces. Since we consider surface representations, we apply the Equation 3.2. The first surface shown in the top row defines the surface of a cube. A cube has F=6 faces, V=8 vertices, and E=12 edges. The cube is the single connected component, so K=1. Therefore, its Euler characteristic is $\chi$ = 6 - 12 + 8 = 2, and its genus is $g = \frac{2-\chi}{2} = 0$. This confirms that the cube does not have any hole.

The surface shown in the bottom row of *Figure 3.2* is the surface of a cube with a hole. We remove the region of a horizontal square bar from the cube, and create polygons for the boundary surface. The left image shows eight of the F=16 faces of this surface, while the right image show the E=32 edges and the V=16 vertices respectively as bold lines and balls.

$$\chi = F - E + V = 16 - 32 + 16 = 0 \tag{3.4}$$

$$g = \frac{2-\chi}{2} = \frac{2-0}{2} = 1 \tag{3.5}$$

We compute the number of holes for the second surface in *Figure 3.2* based on its Euler characteristic. As shown in the expressions 3.5, the Euler characteristic of this surface is zero, and its genus is one. This indicates that the surface has one hole.

The Euler characteristic does not provide any information about where the holes are located. Actually, this information is not relevant for Topology, but for Geometry. The location of a hole is not an invariant under deformations. Nevertheless, it is important to accurately locate every hole to correct them with only a small modification to the geometry of the shape.

The phrase 'the topology of a structure' generally refers to the number of holes in the structure. On the other hand, 'the geometry of a structure' means the spatial location of its voxels relative to each other. A deformation changes the geometry of the structure, but not its topology. Piercing a hole into the structure both modifies its geometry and its topology. When the hole is small, the impact on the geometry can be hardly visible. However, the topological change is dramatic for some applications such as neuroscience. Indeed if the brain boundary has a hole, it cannot be described as a single sheet.

### 3.3.2 Reeb graphs

We describe the Reeb graphs as defined by Wood [Wood(2003)]. A Reeb graph is a graphical representation of the connectivity of a surface between critical points [Reeb(1946)]. They describe the skeleton of the surface. More commonly they are used to represent the relations between the level sets of a surface.

Given a scalar function defined on the surface, the Reeb graph tracks the connected components of the pre-image of the function. For instance, if the distance function follows the increments of the $z$ function, the nodes in the Reeb graph are the contours created by the intersection of the surface

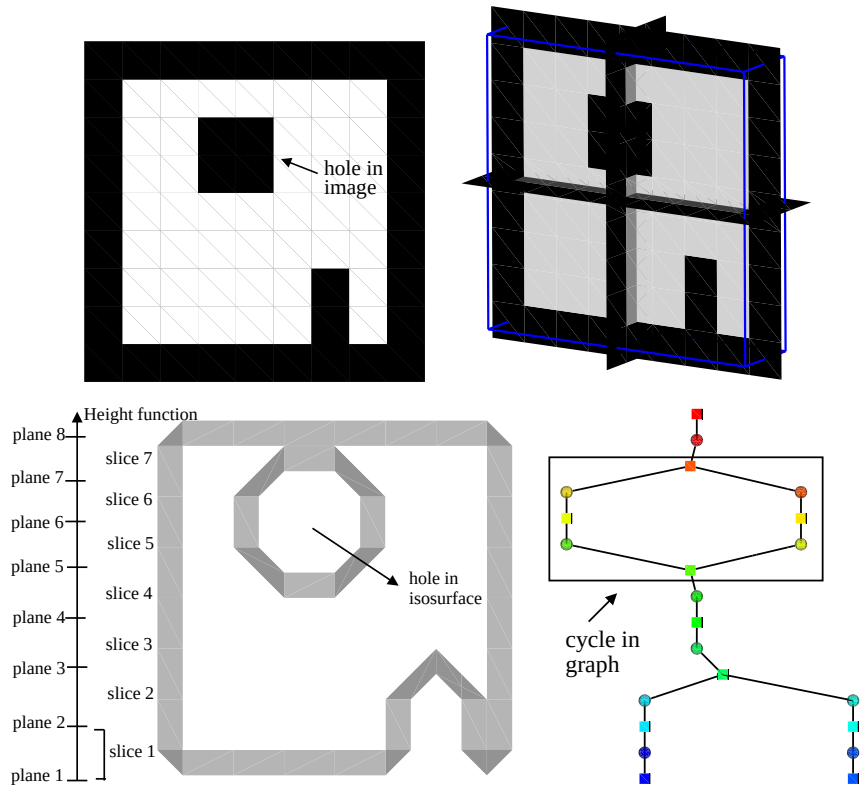


Figure 3.3: *The Reeb graph detects a hole in the image as a cycle. The top left and right images respectively shows one cut and three orthogonal cuts through a 3D image with a hole. The isosurface and the Reeb graph for this 3D image are displayed in the bottom row. The graph is made of ribbon nodes (cubes) and contour nodes (balls). The hole in the center of the isosurface corresponds to the cycle in the graph. Therefore we can detect holes in the image by searching for cycles in the graph.*

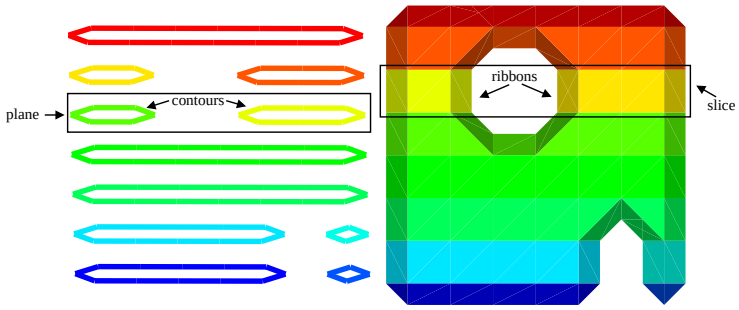


Figure 3.4: *Every closed line within a plane is a contour while every surface strip within a slice is a ribbon. The left image shows every contour in the image with a different color. Similarly the right image shows every ribbon with a different color. The ribbons and contours have the same color-coding as the Reeb graph in Figure 3.3.*

with the horizontal planes at each increment of the $z$ function. The Reeb graph tracks how these contours split or merge along the $z$ direction.

As the distance function propagates over a hole, the contour split at a given plane, and merge at a higher plane. The volume between these two planes indicates the region where the hole can be found in the image.

### Augmented Reeb graphs

In a standard Reeb graph, the relation between contours from successive planes is lost, and must be recovered with an assumption. To avoid ambiguities, Wood et al. [Wood *et al.*(2003)] introduce the Augmented Reeb Graph.

While a standard Reeb graph only considers the intersection of the surface with horizontal planes, the Augmented Reeb Graph takes into account the volume between consecutive planes. This volume is called a slice of the image, and is bounded by a lower and an upper horizontal planes. The connected components inside a slice are called ribbons.

The ribbons connect the contours on the lower plane with the contours on the upper plane. This removes the relation ambiguities between these contours. Similarly, the contours connect one ribbon from the lower slice with one ribbon from the upper slice. The Augmented Reeb Graph is then a succession of two layers: a layer of contour nodes, followed by a layer of ribbon nodes, and so on.

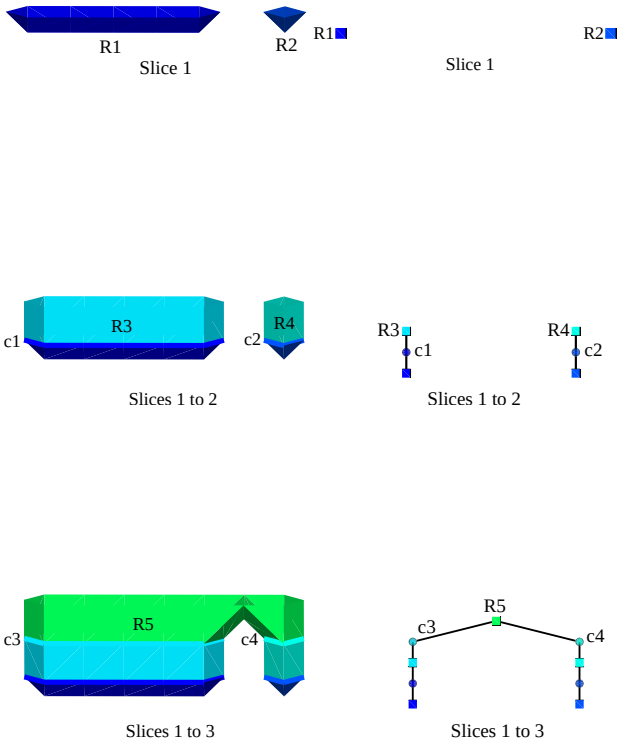


Figure 3.5: *(Reeb graph construction see also next Figure) The Reeb graph is built from bottom to top to detect and isolate the holes in the image. Every row of images shows the surface and the corresponding Reeb graph one slice at a time. The top left image shows the first two ribbons R1 and R2 found in slice 1, while the ribbon nodes are represented with cubes in the graph on the right. The second row of images indicates the ribbons R3 and R4 and the contours c1 and c2 found in slice 2. The graph represents every contour with a ball. The last row shows that ribbons R5 connects contours c3 and c4 in slice 3.*

*Figure 3.3* illustrates the Augmented Reeb graph for a given isosurface, extracted from a volumetric image. The top left image shows a rendering of this isosurface, where a hole into the surface is visible. The top right image corresponds to the graph of this surface. The bottom left image shows the contours that intersect every horizontal plane. The bottom right image shows isosurface color-coded based on its ribbons. In the graph, the contour nodes are represented with a ball, while the ribbon nodes are represented with a small cube. One can see that the hole in the image corresponds to a cycle at the top of the graph. Therefore the Augmented Reeb Graph is useful to detect the holes in the image, and find a contour that gives an approximate localization of the hole.

As shown in *Figure 3.3*, a hole in the image corresponds to a cycle in the graph. Therefore we can use the graph to detect a hole in the image and locate the plane where this hole is detected. The cycle in the graph is formed by the contours that split and then merge in the graph. Every contour in the cycle is a non-separating loop around the hole. Nevertheless another non-separating loop that crosses these contours could be smaller.

*Figure 3.4* indicates that contours are closed lines within a plane while ribbons are surface strips within a slice. The color-coding for the ribbons and the contours indicates the correspondence with the graph nodes in *Figure 3.3*.

*Figures 3.5, 3.6 and 3.7* illustrate the detection of a hole during the construction of the Reeb graph. Every row shows the surface (left)) and the graph (right) during the construction of the Reeb graph from the bottom slice to the top slice. Every ribbon and every contour are represented in the graph with a cube and a ball respectively.

In Figure 3.5, the surface minima are discovered in slice 1 and correspond to ribbons R1 and R2. In slice 3, ribbons R5 connects contours c3 and c4, but does not create a cycle in the graph.

Figure 3.6 the beginning of a hole is hit in slice 5. Contours 7 and 8 splits from ribbon 7 and create two branches in the graph. Nevertheless the hole is not completely explored in slice 6.

In Figure 3.7, ribbon R10 merge contours c9 and c10 and creates a cycle in the graph. Therefore a hole is detected. The nodes in the cycle (R7, c7, c8, R8, c9, c10 and R10) correspond to a region in the image where the hole is isolated. The hole can then be accurately localized and corrected in this region.

*Figure 3.8* illustrates a case where the Reeb graph fails to detect the hole. The top row shows the four planes through the 3D image. The bot-

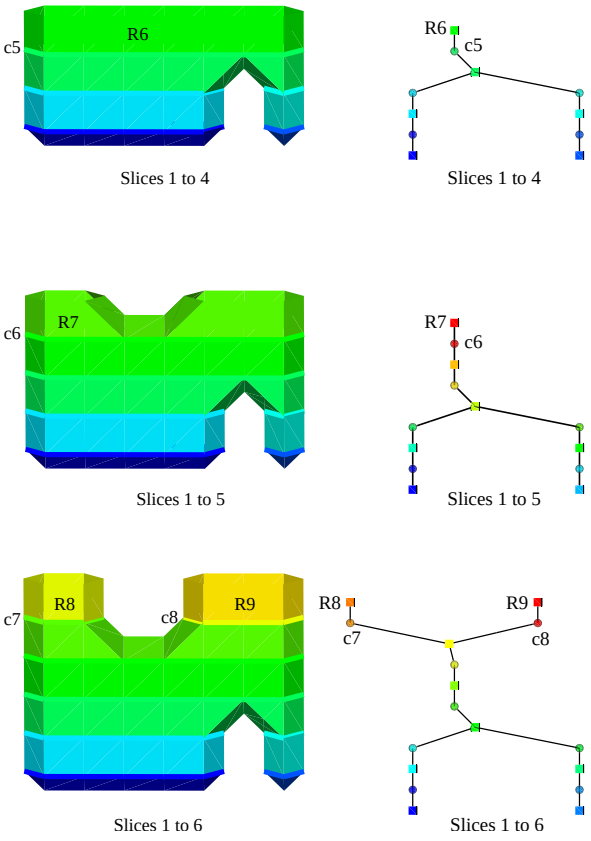


Figure 3.6: *(continued from previous Figure, see also next Figure) To detect the holes in the image, the Reeb graph tracks where the contours split and merge. The top left image shows ribbon R6 and contour c5 found in slice 4. As shown in the graph on the right, R6 is connected to c5. In the next row, ribbon R7 and contour c6 appear to be connected in slice 5. Then in the third row, ribbon R8 is connected to contour 7 while ribbon R9 is connected to contour c8.*

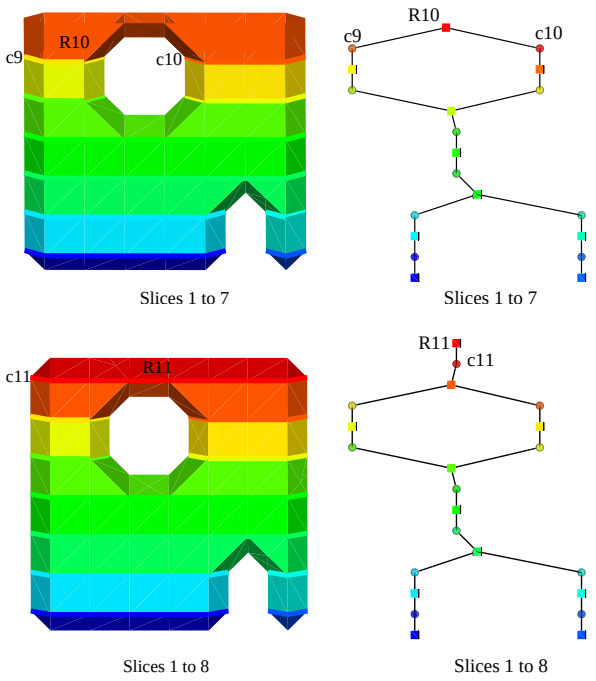


Figure 3.7: *(continued from previous Figure) A hole in the image is detected as a cycle in the Reeb graph. The first row of images shows ribbons R8 and R9 and contours c6 and c8 on the surface (left) and in the graph (right). In the next slice, ribbon R10 connects contours c9 and c10 and creates a cycle in the graph with the nodes R7, c7, c8, R8 and R9. Therefore the nodes in the cycle isolate the hole in a region of the image. Later we will accurately localize and correct the hole within this region. Finally the last ribbon and contour in the image (R11 and c11) are discovered and the graph construction ends.*

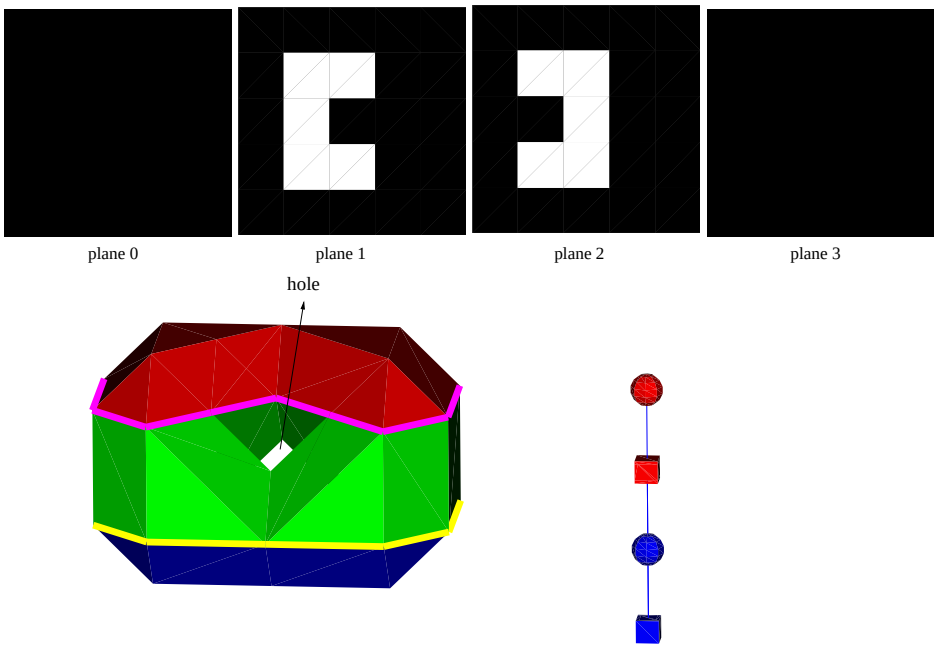


Figure 3.8: *The Reeb graph does not detect holes that lie entirely within one slice of the image. This Figure shows the simplest example of such a hole. The first row of images shows the four planes of the image. The bottom left image shows the isosurface extracted from this image, with its ribbons and contours in color. The bottom right image shows the corresponding Reeb graph. The colors indicate the correspondence with the ribbons and contours on the left image.*

tom left image shows the isosurface for this binary image. An arrow points to the hole in the green ribbon. However, the graph in the bottom right image does not indicate a cycle. Due to its construction from horizontal planes, the Reeb graph cannot detect holes entirely contained in a single slice, or a single ribbon.

## 3.4 Definitions

In this section, we summarize the definitions proposed previously. These definitions could be used as a reference for the reading of this dissertation.

- **3D image:** 3D grid of intensities. The grid nodes are called voxels.

- **hole:** a hole in a binary image is a small background region (respectively foreground) voxels that go through the foreground (respectively background) region. A tunnel, a bridge, or a handle are representations of a hole in an image. The same definition applies to a hole in a surface, where the foreground is the region inside the surface, and the background is the region outside the surface.
- **loop:** closed line on a the surface, used to locate a hole.
- **surface mesh:** set of adjacent faces (generally triangles), edges, and vertices.
- **isosurface:** surface extracted from a 3D image. Like iso-elevations on a 2D geographic map describe lines of constant elevation, an isosurface describe a surface of constant intensity value (isovalue).
- **Marching Cubes:** isosurface algorithm, that triangulates the volume between eight neighboring voxels (logical cube) at a time.
- **plane:** used to define an horizontal cut through a 3D image.
- **slice:** volume of the image between two consecutive planes.
- **contour:** closed line obtained by the intersection of an isosurface with an horizontal plane.
- **ribbon:** connected portion of an isosurface, limited by two consecutive planes.
- **Reeb graph:** graphical representation of the topology of a 3D image, that connects the image components contained in horizontal planes.
- **Augmented Reeb graph:** Reeb graph added with ribbon nodes.
- **cycle:** a cycle in a Reeb graph corresponds to a hole in the image.
- **Euler characteristic $\chi$:** expression from Discrete Topology. $\chi = -C + F - E + V + H$, where $C$, $F$, $E$, $V$ and $H$ are the number of cubes, faces, edges, vertices and boundaries respectively.
- **genus $g$:** number of holes, computed from the Euler characteristic $\chi$, as $g = \frac{2K-\chi}{2}$, where $K$ is the number of connected components in the volume.

## 3.5 Conclusion

In this chapter, we have defined some important notions of Topology, which studies the holes. As a reference, we have listed the key definitions of Discrete Topology for volumetric images. Two Discrete Topology tools are especially important to understand topological algorithms. First, the Euler characteristic allows to compute the number of holes, but does not locate their position. Second, the discrete Reeb graph indicates an approximate location of the holes across horizontal planes. However it cannot detect holes contained between two consecutive planes. Therefore, other techniques to accurately locate every hole in the image are required.

In the next chapter, we will use the terminology we have defined and review the techniques proposed to simplify the topology of the image, i.e. to remove the holes in the image.

# Chapter 4

# Related Work

In the previous chapter, we defined some concepts of Topology. Several methods build on these concepts to simplify the topology of an image, i.e. to remove the holes in the image.

These methods presented in this chapter can be classified in two categories: the methods that fit a model into the image with a topological constraint, the methods that modify the image to correct its topology.

When assessing these methods, one major criterion is how much of the geometric accuracy of the brain segmentation they preserve. Therefore the difference between the input image and the topologically corrected image must remain small.

## 4.1 Topologically constrained models

The first proposed approach to extract a representation of the brain with spherical topology is to fit a model into the image with the constraint that the topology of the model remains the topology of a sphere. Both surface and volume models have been used in this approach.

### 4.1.1 Surface model

Many authors, such as Davatzikos [Davatzikos and Prince(1995)] and Bischoff [Bischoff and Kobbelt(2003)] propose to fit a surface model with spherical topology onto the brain surface. This surface, called a Deformable Surface, evolve generally based on a gradient descent toward the brain boundary in the image. Smoothing operators are introduced to prevent surface intersections and splittings.

**limited accuracy** To push the surface into the brain folds, some authors use a multiresolution strategy. The surface is first attracted towards the largest features of the image, and then towards the smallest features. The trade-off between the image-based term and the smoothing term could be adapted to help the surface fit smaller and smaller features.

**self-intersections** If the surface intersects itself during the evolution of the surface, it would not describe a correct volume anymore. Bischoff and Kobbelt [Bischoff and Kobbelt(2003)] propose to evolve a deformable contour along the grid lines to detect and prevent topological changes. When the contour touches itself at a contour node, a topological change or self-intersection could occur and the contour is thus stopped around this node. The generalization of this method to 3D surface might be more involved.

**limitations** However due to the effect of the smoothing operators, the deformable surface often does not completely penetrate into the brain folds of the surface. Therefore the representation of the brain with a deformable surface is generally not accurate.

### 4.1.2 Volume model

Early methods proposed by Mangin [Mangin *et al.*(1995)] and Aktouf [Aktouf *et al.*(1996)] apply a region growing process constrained to preserve a spherical topology. The region is initialized with a seed voxel and progressively expanded to include neighboring voxels within the brain volume.

In case the insertion of a voxel into the region creates a hole, it is ignored and the region expands to other voxels. The region stops when no more neighboring voxels can be inserted without creating a hole.

Since the shape of the resulting region depends on the order of the voxels, the region could grow into small holes of the brain and not penetratre into other large parts of the brain. Therefore Krieskorte and Goebel [Kriegeskorte and Goeble(2001)] introduce a distance-to-surface metric to prioritize the voxels in the image. The distance from every image voxel to the image isosurface defines the cost of inserting this voxel into the growing region.

However the folded shape of the brain surface results in many voxels with a similar cost. Therefore even with the prioritization of voxels large differences could occur between the input and the output images.

### 4.1.3 Limitations of topologically constrained models

Topologically constrained models have three main problems. First they require a sufficiently good initialization to capture the brain volume. Second they often do not penetrate in the narrow brain folds. Third they can miss a large region of the brain during their evolution.

## 4.2 Correction of the image

Instead of fitting a topologically correct model into the image, an alternative approach it to correct the topology of the image. Similarly to the model based methods, authors have used surface and volume representations for the correction of the image. Below we describe both methods.

### 4.2.1 Correction of a surface

**surface inflation method** Fischl et al [Fischl *et al.*(2001)] propose to correct a surface representation of the brain image. They first apply an isosurface algorithm to represent the brain surface with a triangle mesh. Then they 'inflate' the brain mesh based on equations of Mechanics. Topological defects are detected as flipped triangles on this mesh. Then the flipped triangles are removed from the input mesh and the surface is re-triangulated in this region.

Two main problems occur with this technique. First the number of removed triangles could be larger than strictly necessary creating a large difference with the input mesh. Second the surface in the re-triangulated region could intersect other parts of the surface and thus the resulting surface would not define a volume anymore.

**front propagation method** Axen et al. [Axen and Edelsbrunner(1998)] [Axen(1999)] use a wavefront traversal over a surface mesh to define a distance function over the surface. A discrete distance is propagated to all vertices in a triangulated mesh starting from an arbitrary vertex. Some vertices, called grounded vertices, must be detected first and processed separately. However the processing is very sophisticated and thus its practical use is limited.

Guskov and Wood [Guskov and Wood(2001)] propose to simplify the topology of the surface mesh based on a wavefront traversal of the surface triangles. Starting from a number of seed triangles, they grow each

region concurrently until it is bounded by neighboring regions or until it encloses a hole. The hole is detected when two boundary components of the wavefront merge.

Then the triangles around the hole are subdivided and the surface is cut open along the subdivision. Both sides of the cut are finally re-triangulated to seal the opening on the surface.

As a disadvantage only holes fully contained within a region area can be detected. Besides this method only consider a cut along the wavefront while a transverse cut could result in a much improved correction.

**limitations** For both of these methods, the surface is retriangulated after the removal of a hole. The retriangulation can cause the output surface to intersect itself. Therefore this surface would not represent a 3D volume anymore.

### 4.2.2 Correction of the volume

Various methods have recently been proposed for the correction of the volume. Unlike surface based methods, the isosurface extracted from the resulting volume is guaranteed to be free of self-intersections. Therefore they always define a 3D volume.

Some volume correction techniques are based on the decomposition of the image into components. Other methods use the concept of Reeb graph presented in the previous chapter. Although decomposition methods also build a graph, they differ significantly from Reeb graph methods.

To compare these methods, we must observe the modifications they tend to perform in the image. They must preserve the geometric accuracy of the brain input image.

### 4.2.3 Decomposition into components

**mathematical morphology method** Han et al [Han *et al.*(2002)] propose to apply successive morphological openings to decompose the object in the image into a set of body and residual components. The residual components are inserted back into the body component if they do not create a hole. A sequence of openings at increasing scales are needed to remove every hole in the image.

This method has two main disadvantages. First the shape of the residual components, and thus the shape of the corrections, depend on the shape of the structuring element used for the morphological openings.

Second the applications of successive openings require successive reading of the entire dataset.

**multiple region growing method** Ségonne et al [Ségonne *et al.*(2003)] [Ségonne *et al.*(2004)] decompose the image based on a multiple region growing process. The region growing process depend on statistical and geometrical information.

Unlike other methods that can produce images with different objects as long as they do not exhibit a hole, this method produces an output image with a single foreground object and a single background object. Although this assumption can be valuable for some images, it can cause a wrong result for other images.

As an example where this method would fail, we consider a brain segmentation where the ventricles are represented as a background region inside the foreground region of the brain. Even when both the brain and the ventricles have a spherical topology in the image, this method could modify the image. Two potential modifications are the removal of the ventricles and the connection of the ventricles to the larger background region outside the brain.

**limitations** The decomposition into components based on mathematical morphology requires several passes over the image, which can be computationally intensive. On the other hand, the decomposition of the image of Ségonne et al assumes that the correct object in the image has only one component which might not be verified.

### 4.2.4 Methods based on a Reeb graph

Two research groups have recently proposed to correct the topology of images based on the construction of a Reeb graph. The first group uses the Reeb graph both for the detection and the simplification of the topology, while the second group proposes an alternative method for the simplification of the topology.

**detection and simplification based on a Reeb graph** As described in Chapter 3 the Reeb graph represents image components in horizontal planes and detects a hole in the image as a cycle in the graph. Shattuck and Leahy [Shattuck and Leahy(2001)] propose to break the graph cycle, and thus the hole by removing one component in the cycle. To remove

an object (respectively background) component they modify its voxels to background (respectively object).

To search for the smallest component that would remove the hole, they they build a Reeb graph along the three cardinal directions. Nevertheless since the correction only occur in a cardinal plane, the correction in the planes can be unnecessarily large.

Besides this method cannot correct every hole in the image. Indeed Reeb graphs cannot detect holes contained in two consecutive holes as shown in Chapter 3. Some holes are undetected in the three Reeb graphs built along the three cardinal axes.

**simplification based on a shortest loop** Wood et al [Wood *et al.*(2003)] [Wood(2003)] improves the previous method. First they address the pathological cases, called intra-ribbon holes, with a modified Reeb graph built on a face-by-face traversal.

Like Shattuck and Leahy, Wood et al detect a hole with a component in the Reeb graph. The main difference is the simplification of the topology. Wood et al modify the image inside a loop located either around or across the hole. These two loops, called Reeb loop and cross loop, are computed sequentially based on a shortest path algorithm.

Compared to the method of Shattuck and Leahy, the method of Wood et al do not require an exploration of the image along each cardinal axis.

**limitations** The correction of holes along cardinal directions can cause large deformations in the image. On the other hand the correction of holes along shortest loops generally provide a smaller deformation. Nevertheless the minimization of the length of the loop is not always the best criterion. The shortest loop can lead to a larger deformation than strictly necessary. For instance a longer loop might have less voxels inside to modify than a shorter loop.

## 4.3 Conclusion

This chapter has presented the methods for the simplification of image topology according to two categories, models with topological constraint, and the correction of the image. We have assessed the quality of every method based on how much of the geometric accuracy of the input image they preserve.

On the one hand, the models with topological constraint try to progressively capture the brain volume and to preserve a spherical topology during the progression. These methods suffer from two main drawbacks. First the model needs a good initialization, which might require the intervention of a human operator. Second the model sometimes stops before capturing a large region of the brain because to capture this additional region it would create a hole with the region inside the model.

On the other hand, the methods that correct the image detect the hole either on a surface or in the volume, and apply a correction for every hole.

First the correction of a hole on the surface can intersect other regions of the surface, and thus the corrected surface would not represent a 3D volume.

Second the methods that correct a hole in the volume search for the smallest correction to the image based on either a Reeb graph and a shortest loop or a decomposition into components. However the correction along a shortest loop around the hole do not always provide the smallest correction. Besides the correction along image components requires either computational intensive processings of the image or assume that the output image has only one component.

In the following three chapters, we will develop the concepts for our volume correction method and our algorithmic choices to preserve the accuracy of the input image. These chapters present the main steps of our algorithm: the hole detection, the hole localization, and the hole removal.

# Chapter 5

# Topology Detection

The review of topology simplification algorithms in the previous chapter showed that they either require a computational intensive processing of the volume or they potentially modify a large region of the image.

We assume that the removal of holes from the brain segmentation should modify the smallest number of voxels to preserve the global shape of the brain. Based on this assumption, we search the shortest loop around the hole, and modify the region inside. The small size of the loop guarantees that we only modify a small number of voxels. For every hole, we proceed in three steps: the detection of the hole, the localization of the shortest loop, and the modification of the volume inside the loop.

After an overview of our algorithm, this chapter will present the first step of our algorithm, the detection of the image topology. Our principle of hole detection is based on a local front propagation, and a modified Reeb graph. The advantage is to detect the hole with less computation, and less memory than existing methods. We can thus remove the holes in the segmentation for large medical images.

## 5.1 Overview of the algorithm

To introduce the context of this chapter, we first give an overview of our algorithm. Then we focus on the subject of this chapter, the detection of holes.

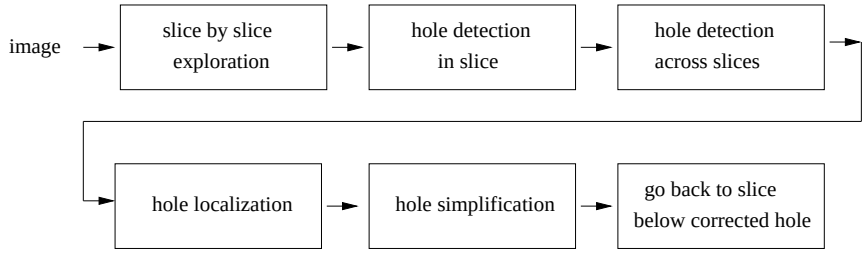


Figure 5.1: *Our algorithm explores the image one slice after the other. If a hole is detected within the current slice or across the previous slices, then the hole is localized and removed. Since the hole simplification modifies a region of the image, the algorithm starts over just below this region.*

### 5.1.1 Entire algorithm

Our algorithm progressively explores the image from the bottom slice to the top slice and performs three operations: hole detection, hole localization and hole simplification. The diagram of our algorithm is shown in *Figure 5.1*. First it detects whether a hole exists within the current slice, and then whether a hole exists across the current slice and the previous slices. Second if a hole is detected, the algorithm localizes the hole within the image. Finally it simplifies the localized hole, i.e. it removes the hole from the image. Since the hole simplification modifies a region of the image, the image exploration starts over from the slice just below the modified region.

### 5.1.2 Hole detection

*Figure 5.2* shows the main steps of our algorithm for the detection and localization of holes. Every hole is guaranteed to be detected in a single exploration of the image, by the combination of two methods. First, a wavefront explores one slice of the image, and detects the holes limited in this region. Then, the construction of a modified Reeb graph detects the holes within the region of the image already visited.

If a hole is detected in the wavefront or in the graph, the process to localize the hole in the image starts. The hole is localized with a short loop around its boundary surface. The correction of the hole will depend on the extent on this loop.

Only two successive planes of the image are needed in memory. Topo-

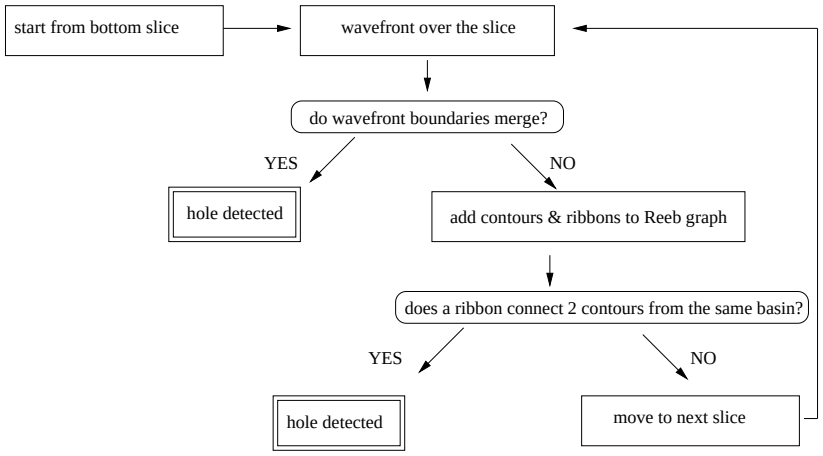


Figure 5.2: *Our algorithm detects every hole in the image in a single exploration. Holes are either detected during the exploration of the image, or in the graph. When a hole is detected, the algorithm localizes the hole within a small extent of the image.*

logical information about already explored planes is constructed by the wavefront, and recorded in a modified Reeb graph. This information is made of ribbons, contours, and basins. A basin is a concept we introduce to code more efficiently the topology of the image.

## 5.2 Exploration of the image

We explore the image in such a way that we can detect holes between two planes of the image in a single exploration of the image. The method is based on a wavefront that propagates over the isosurface in the image. When the wavefront splits and later merges again, we know we have enclosed a hole.

### 5.2.1 Wavefront over the image

We pick one triangle of the isosurface, and start the wavefront traversal of the image from this seed triangle. Since the wavefront propagates from one triangle to another across common edges, we create a list of active edges. To start with, the list contains the three edges of the seed triangle.

We pick the last edge from the active list, and conquer the triangle that shares this edge with an already conquered triangle. We thus remove the edge from the active list, and insert the other two edges of the newly conquered triangle. To make the wavefront propagate over the isosurface, we repeat the process of picking one edge from the active list, and conquering one more triangle.

To reduce the memory required during the wavefront propagation, we remove from the active list every edge that lies on either the lower plane or the upper plane. The wavefront propagates within these two planes across edges. Since lower and upper edges only connect to triangles beyond these planes, it is not worth keeping them in memory. We still keep track of the boundaries of the wavefront with an ordered list of edges between the lower and upper planes.

### 5.2.2 Detection of holes in the wavefront

Sometimes the newly conquered triangle shares more than one edge with the wavefront. We do not insert the edges of this triangle that are already in the active list. What is more important for detection of holes, these edges can cause the wavefront to either split or merge.

We track the closed lines on the boundary of the wavefront to detect a split or a merge. These events only occur when the triangle conquered across an active edge has a second active edge. In case both active edges belong to the same boundary line of the wavefront, the line folds over at the position of the second edge. The boundary line is thus split between two new boundary lines, one on each sides of the second edge.

When two boundary lines of the wavefront merge, a hole is detected. In case the second edge of the conquered triangle belong to a different boundary line than its first edge, the two boundary lines merge. The wavefront has propagated around a hole, since the boundary line has first split at one end of the hole, and the two new boundary lines have then merged again at the other end of the hole. The second edge defines the position where we have detected a hole.

The following pseudo-code summarizes the sequence of operations to localize the hole in the ribbon. As a notation, we call Ts the first triangle of the wavefront, and Ta and Tb the two triangles where the hole is detected. We create two lists of triangles La and Lb starting from the triangles Ta and Tb. We then iteratively insert one triangle into each list. We consider the last triangle in the list TN and insert the parent triangle TN+1 of this triangle. Since TN+1 is adjacent to TN, both sequences of triangles in La

and Lb define a surface strip. Finally La and Lb reach Ts and we move to step 5. In step 5, we find Li, the triangles common to La and Lb. Then we remove Li from La and from Lb, and merge the reduced lists into Lj. Lj defines a closed strip of triangles around the hole. Finally we extract the two closed lines that bound Lj and keep the shortest. The shortest boundary is the Reeb loop that localizes the hole.

```
step 0: Ts := seed triangle of the wavefront
step 1: find the triangles Ta and Tb that detect the hole
step 2: initialize the lists La := {Ta} and Lb := {Tb}
step 3: for L := {T0,...,TN} insert TN+1 (parent of TN)
step 4: if the TN+1 = Ts , go to step 4,
otherwise go back to step 2.
step 5: Li := intersection of La and Lb
step 6: Lj := La\Li + Lb\Li
step 7: B0 and B1 := boundaries of Lj
step 8: Reeb loop := argmin {B0,B1}
```

### 5.2.3 Illustration of the wavefront

*Figure 5.3* shows the detection of a hole during the propagation of the wavefront. We start the wavefront from a single triangle, and conquer the triangles connected across edges of the active list. The portion of the isosurface that is already conquered is shown in blue.

The wavefront propagation for an object with a hole between two planes is featured on *Figure 5.3*. The object is shown in the top left image. The wavefront boundary contains the active edges. When we start, the boundary of the wavefront is a single closed line. It is colored in green in the upper right image. When the newly conquered triangle has more than one edge from the active list, it disconnects two parts of the boundary. The two parts of the boundary are colored in green and red respectively on the second line of the figure.

The wavefront conquers more and more triangles until it encloses the hole. At this point, the newly conquered triangle connects the two parts of the wavefront boundary. The two parts merges, and the wavefront boundary is a single closed line again. The bottom left image in *Figure 5.3* shows this event. The new boundary of the wavefront is colored in green. A hole is detected for the edge that connects both parts of the boundary.

Finally, the boundary of the wavefront splits again when the isosurface is completely conquered. Each resulting boundary line represents

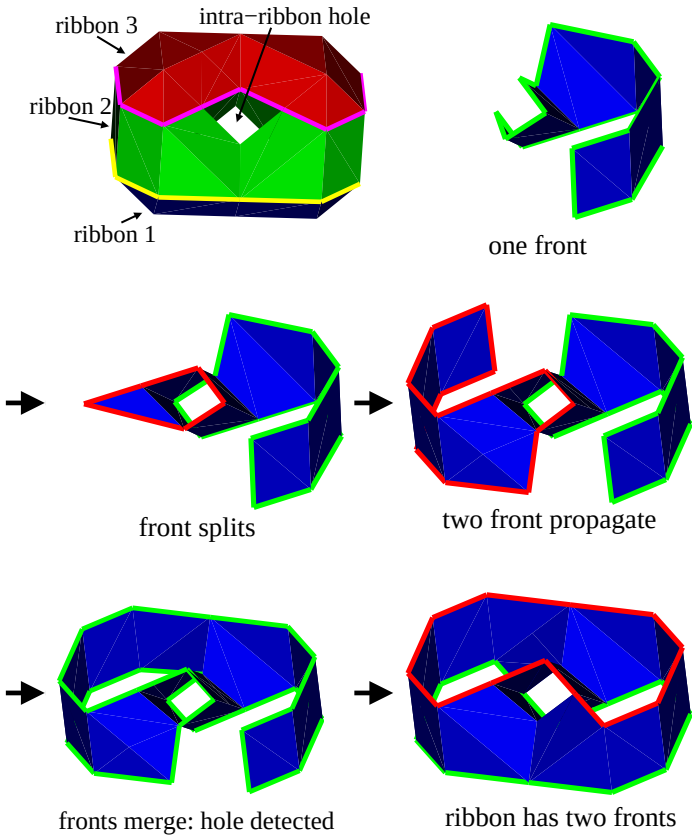


Figure 5.3: *We detect a hole inside a ribbon when a wavefront encloses the hole. The upper left image shows the isosurface with one hole in the green ribbon. The other images from top to bottom and left to right illustrate the propagation of the wavefront in this ribbon. The surface inside the wavefront is colored in blue, and the boundaries of the wavefront in green and red. The third image shows that a new boundary (red) is created when the wavefront first reaches the hole. On the fifth image, we see that this boundary merge with the first boundary when the hole is completely enclosed into the wavefront. The merging of boundaries allows the detection of the intra-ribbon hole. The wavefront ends when it has propagated over the entire surface of the ribbon as shown in the last image.*

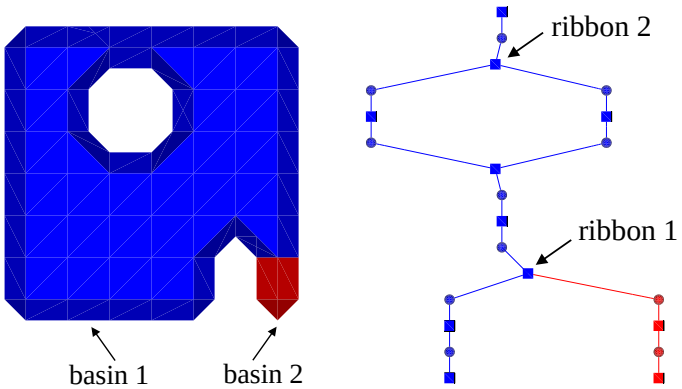


Figure 5.4: *With our concept of basin, the detection of holes is easier than searching for cycles in the graph. The left image shows the two color-coded basins while the right image represents the corresponding graph with the same color-coding. Ribbon 1 connects two lower contours from two different basins (blue and red), and thus does not correspond to a hole. On the other hand, since the two lower contours of ribbon 2 belong to the same basin (blue), a hole is detected.*

the boundary of the wavefront on one plane of the image. The bottom right image in *Figure 5.3* shows the boundary lines on the upper and lower planes in red and green respectively. Since the wavefront was constrained to propagate within one slice of the image, it only describes a strip of the isosurface. This strip, shown in blue in the bottom right image, corresponds to a ribbon in the graph.

## 5.3 Modified Reeb graph

We develop a modified Reeb graph to detect the holes that extend over more than two image planes. We introduce the concept of basin to efficiently detect holes in the image.

A basin is a region of a connected component, that starts from a minimum of the surface, and stops when it merges with another basin or when it reaches the top of the connected component. A similar notion exists for the watershed method to segment images.

To illustrate the concept of basin, we use an analogy with water filling the interior of the object. Every connected component is an empty tank. The basin represents the region filled with water if a tap is connected to

every minima of the tank. A tap is closed when its water mixes with the water from another tap within the same tank, or when the tank is full.

With the introduction of basins, we detect the holes in the image without searching for their corresponding cycles in the graph. When a ribbon node connects several contour nodes in the graph, a cycle could occur. If every contour belongs to a different basin, the graph has no cycles. On the other hand, if two or more contours belong to the same basin, they create a cycle in the graph. Therefore we detect a hole when a ribbon node connects two contour nodes from the same basin, and do not need to search for cycles in the graph.

*Figure 5.4* illustrates the detection of hole based on our modified Reeb graph. The left image shows the minima of the surface colored in blue, and red. The graph of this surface is shown on the right image, where ribbon and contour nodes are color-coded based on their basin. The ribbon indicated with the number 1 connects two contours from two different basins (blue and red), and thus does not correspond to a hole in the image. The top ribbon, with the number 2, connects two contours from the same basin (blue). Therefore, it represents a hole in the image. It is sufficient to analyze the lower contours for ribbons in the current slice to detect holes in the image. This method is thus easier than searching for cycles.

## 5.4 Memory requirements

Since our algorithm operates in a limited number of slices in the image, its complexity is limited to $O(n^2)$ for an image with $n^3$ voxels.

### 5.4.1 Wavefront propagation

Only the image data in a single slice, i.e. in two successive planes, is accessed during the wavefront propagation. We build a structure with a similar size to record whether a triangle has been visited or not. Since the ribbons within a slice do not share any triangle, we initialize this structure only once, and use it for the propagation of every ribbon within the two planes. For the ribbon traversal, we limit the size of the data in memory to the data in one slice.

The largest data structure for the wavefront propagation corresponds to the edges we visit during the wavefront propagation. This data structure has a size proportional to the size of an image plane. For an image with dimensions $n$ x $n$ x $n$ = $n^3$ voxels, the size of an image plane is $n$ x $n$

= $n^2$. Every cube of the image can contain up to five edges. Hence the size of this data structure is $5n^2$, and increases much more slowly than the size of the image on disk, i.e. $n^3$.

### 5.4.2 Graph construction

The largest data structure in memory during the graph creation does not grow with the number of planes in the image. We create three data structures for the cubes located between two consecutive planes. These data structures corresponds to the contours ids, the ribbon ids, and the basin ids. Every cube can contain up to five points and $n^2$ cubes are located between two planes. Therefore, like the wavefront data structures, each data structure has $5n^2$ integers.

The overall memory requirement for the ribbon traversal and the graph construction is thus on the order of $O(n^2)$. Since this figure does not increase as fast as the size of the image ($n^3$), large images can be processed even when the computer memory is smaller than the size of the image on disk. Our algorithm is thus appropriate for correcting holes in large images.

## 5.5 Conclusion

In this chapter, we have proposed a method to detect every hole in the image in a single exploration of the image and with only one slice of the image in memory at any time. First a wavefront propagation detects the holes within a slice. Second the holes across the explored slices are detected with a Reeb graph. We introduce the concept of basin in the Reeb graph construction to simplify the detection of these holes.

The advantage of our approach is its low complexity since we detect every hole in a single exploration of the image. Besides the memory requirements for the wavefront and the Reeb graph are only proportional to $n^2$ and $n$ respectively for an image with $n^3$ voxels. Therefore our algorithm can detect the holes in images larger than the main memory.

In the next chapter, we will describe how we localize a region of the image where we will later correct the hole.

# Chapter 6

# Topology Localization

The previous chapter explained our method to detect a hole in the image. This chapter will detail how we localize a region of the image where the hole could be removed.

Our goal is to localize a correction region where the smallest number of voxels will be modified. We could search for this region in the volume. However, to reduce the complexity, we build upon the method of Wood et al [Wood *et al.*(2003)] and search on the isosurface for a loop that goes around the hole. Since this loop encloses the correction region, the minimization of the loop length is a reasonable approximation for the minimization of the number of modified voxels. To improve this approximation, we propose to create a set of candidate loops, and select the loop that minimizes the number of modified voxels.

In this chapter, we will describe our shortest path algorithm to localize every loop in our set of candidate loops. We will highlight our contributions to accurately and efficiently find the path on the isosurface.

## 6.1 Principle of hole localization

We need to find a small section of the hole where we could seal both sides of the hole, so the hole would disappear. Based on the observation in Chapter 3, two transverse non-separating loops respectively around and across the hole define such a section along the isosurface.

For a given hole, we could localize as many pairs of non-separating loops as we want. Based on the criterion of Wood et al [Wood *et al.*(2003)], we search for the pair of shortest loops, and constrain the loops to follow the isosurface edges. Removing this constraint would not guarantee

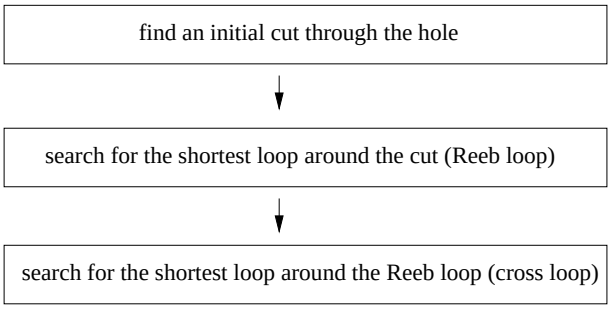


Figure 6.1: *To localize a hole, we search for two transverse loop, a Reeb loop and a cross loop. Starting from a cut through the hole, we first search for a loop transverse to this cut, to create the Reeb loop. Then we search for a loop transverse to the Reeb loop. This creates the cross loop.*

a strictly better solution for the hole removal. Indeed the voxels that we will modify inside the loop have a discrete position.

As proposed by Wood et al, we localize each non-separating loop based on a shortest path algorithm. Starting from an initial cut through the hole, the algorithm propagates from one side of the cut to the other side. The issue is thus to find such an initial cut.

*Figure 6.1* illustrates the principle for the hole localization with two transverse shortest loops. We need a cut through the hole to initialize the search for the two loops. The length of this cut does not matter since the shortest path algorithms that we will use will guarantee that we find a short loop around the hole. Then we apply a shortest path algorithm around this cut to find the Reeb loop. Finally we apply a second time the shortest path algorithm, but this time around the Reeb loop. This results in a loop transverse to the Reeb loop, and called the cross loop.

## 6.2 Localization of a hole detected in the graph

For a hole detected as a cycle in the graph, every contour in the graph cycle defines a cut through the hole. Such a cut was used by Shattuck and Leahy [Shattuck and Leahy(2001)] as a non-separating loop. However the shortest path algorithm generally results in a shorter loop around the hole.

Wood et al [Wood *et al.*(2003)] give the name Reeb loop to the loop around the hole detected in the graph. Consequently they give the name

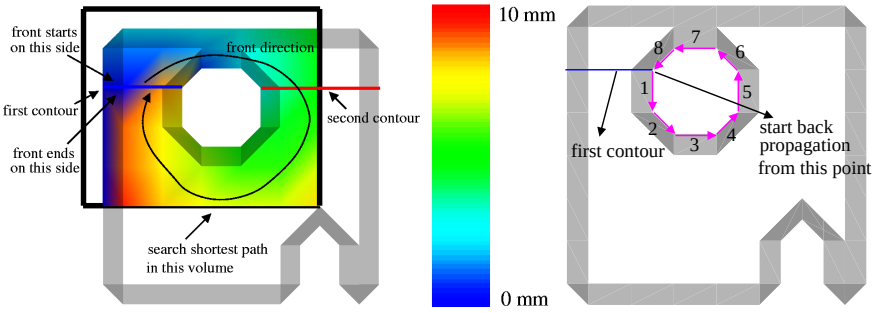


Figure 6.2: *To localize a hole, we enclose it with a loop using a shortest path algorithm. In the first image, the fronts of the shortest path are color-coded with the distance to the start contour from 0 mm (blue) to 10 mm (red). The propagation starts from the first contour and is constrained to go through the second contour. The second image shows the back-propagation from one side of the contour to the other side. The arrows in magenta indicate the sequence of edges that create the loop.*

cross loop to the loop transverse to the Reeb loop.

To find a Reeb loop, Wood et al [Wood *et al.*(2003)] consider the last two contours in the graph cycle. We call these contours the first and the second contours. Both of these contours define an initial cut to initialize the shortest path algorithm. Since a contour can belong to several graph cycles, the path from one side of this contour to the other side is not guaranteed to correspond to the graph cycle between the first and the second contours. Therefore we constrain the path to start from the first contour, propagate through the second contour, and end on the other side of the first contour.

*Figure 6.2* illustrates the localization of the Reeb loop for a hole detected in the graph. This hole is detected when two contours merge. We call the shortest contour the first contour, and the longest contour the second contour. The left and right images in the Figure illustrates the two steps of the method: first the propagation of a front from the upper side to the lower side of the first contour, and then the back-propagation from the lower side to the upper side of the first contour. First a shortest path algorithm is applied from the upper side to the lower side and is constrained to propagate through the second contour. Then as indicated in the right image the back-propagation starts from a point on the contour and returns along edges from the second side to the first side. This path around the hole is

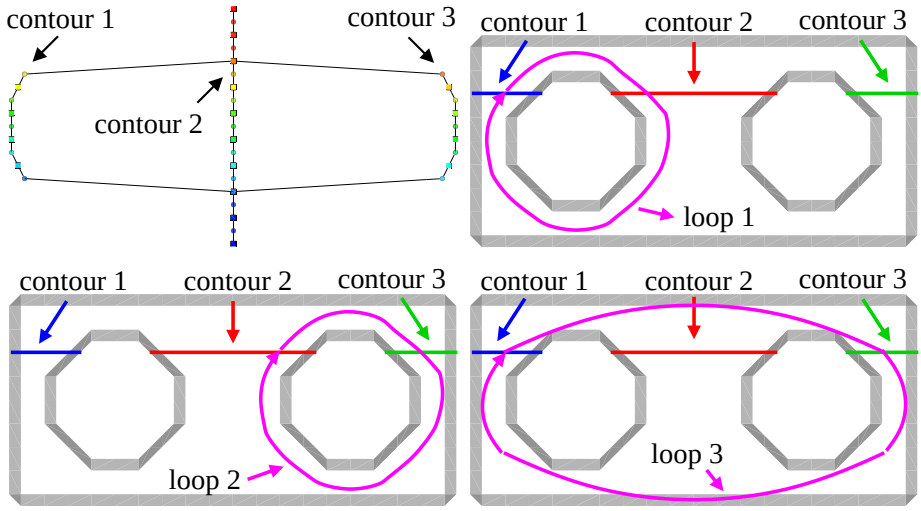


Figure 6.3: *To create a loop around a hole, we constrain a shortest path to cross the last two contours of the hole. The Reeb graph in the top left image detects a cycle between three pairs of contours: (1,2), (2,3) and (1,3). To localize the loop corresponding to each cycle, we use a shortest path that starts from one contour and crosses the other contour. The arrows in magenta in the last three images indicate the loop for each of the three cycles respectively.*

colored in magenta in the right image.

*Figure 6.3* illustrates the creation of a loop from three contours that create a cycle in the Reeb graph. In the first image the Reeb graph detects three cycles, between the contours (1,2), (2,3), and (1,3). To localize the hole that corresponds to each of these cycles, we create a loop through each pair of contours. Our method uses a shortest path algorithm constrained to cross both contours. First the path starts on one side of the first contour. Before it reaches the other side of the first contour, the path must cross the second contour. In the second, third, and fourth images an arrow in magenta illustrates the resulting path for each of pair of contours.

Practically, to constrain the propagation of the path through a particular contour, we prevent the propagation through the other contours.

## 6.3 Localization of a hole inside a ribbon

To localize a hole inside a ribbon, we build upon the method of Guskov and Wood [Guskov and Wood(2001)] and build a triangle strip around the hole. Then we propose to extract one boundary of this triangle strip: this defines one loop around the hole, that we call a Reeb loop. Finally we search for a cross loop transverse to the Reeb loop.

To enclose the hole in a triangle strip, we define two sequences of triangles between the edge where the wavefront splits (splitting edge) and the edge where the wavefront merges (merging edge). One sequence of triangles goes along one branch of the hole, the second sequence goes along the second branch. The concatenation of the sequences of triangles creates a triangle strip around the hole.

We proceed in three steps. First we create a sequence of triangles on both branches of the hole. Second we remove the triangles common in both sequences. Third we concatenate both reduced sequences to create a triangle strip around the hole.

**Creation of a sequence of triangles** Every triangle in the wavefront has a parent triangle. Therefore we can create a sequence of triangles from any triangle to the first triangle in the wavefront by repeatedly moving from one triangle to its parent.

We create two sequences of triangles starting from the two triangles that share the merging edge. Since one triangle belongs to one branch and the other belongs to the other branch, one sequence of triangles follows one branch while the other follows the second branch. Therefore the concatenation of both sequences of triangles encloses the hole.

**Removal of duplicated triangles** However both sequences have a common tail of triangles. Indeed the triangles between the first triangle in the wavefront and the splitting edge exist in both sequences. Therefore we remove these triangles from both sequences and concatenate the reduced sequences.

**Concatenation of the sequences of triangles** The concatenation of the reduced sequences of triangles create a short triangle strip around the hole. This triangle strip covers both branches of the hole and thus encloses the hole.

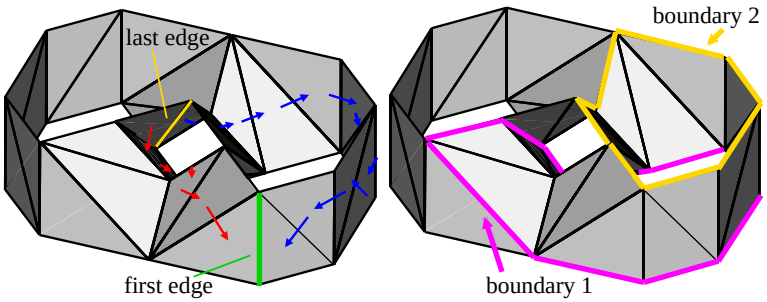


Figure 6.4: *To localize the hole detected during the image exploration, we search for a short loop that encloses the hole. The green edge in the left image is the result of the hole detection. The blue and red arrows show the sequences of triangles created on both sides of the detected hole. The concatenation of the sequences of triangles creates a triangle strip that encloses the hole. The right image indicates in yellow and magenta the two boundaries of the triangle strip. The shortest boundary (yellow) is selected as the Reeb loop since it encloses less voxels than the loop in magenta.*

**Boundaries of the triangle strip** In the method of Guskov and Wood the triangle strip is subdivided and a path inside this triangle strip is created to enclose the hole within a loop. Instead we propose to extract the boundaries of the triangle strip. Therefore we obtain two loops around the hole. Finally we select the shortest boundary and call it a Reeb loop.

**Illustration** *Figure 6.4* illustrates the localization of a loop around an intra-ribbon hole. The hole was detected when two branches of the wavefront merged at an edge. This edge is colored in green in the left image. The blue and red arrows indicate the creation of the sequences of triangles on both sides of the edge. Every arrow point from one triangle to its parent triangle. The concatenation of these triangles creates a triangle strip around the hole. Then the boundaries of the triangle strip are extracted and colored in yellow and magenta in the right image. Both boundaries define a Reeb loop around the hole. However we prefer the yellow boundary to the magenta boundary since it encloses less voxels.

**Guaranteed detection of every hole** The wavefront guarantees that every hole in the ribbon is detected. Indeed it considers every triangle in the ribbon one at a time, and test if a new hole has been created. Therefore ev-

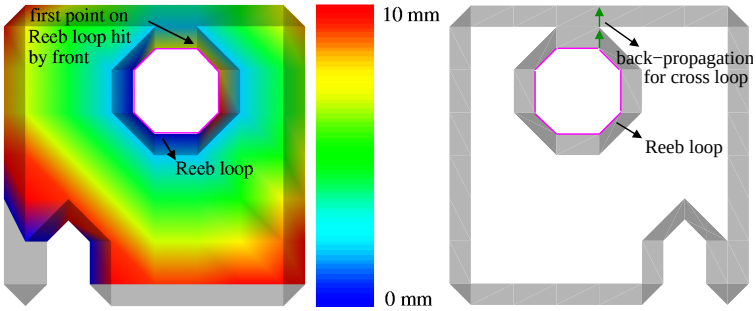


Figure 6.5: *To localize the cross loop transverse to the Reeb loop, we apply a shortest path algorithm from one side of the Reeb loop to the other side. The left image shows the Reeb loop in magenta. The distances propagated by our shortest path algorithm are displayed with a color-coding from blue (0mm) to red (10mm). When the front hits a point on the second side of the Reeb loop, we propagate this point back to the first side of the Reeb loop along edges of the isosurface following decreasing distances. The resulting loop is shown in green in the right image.*

ery hole is detected when the wavefront has visited every triangle in the ribbon surface.

## 6.4 Localization of a cross loop

Once we have localized a Reeb loop, we apply the shortest path algorithm from one side of the Reeb loop to the other side to localize the loop transverse to the Reeb loop. This second loop is called a cross loop.

*Figure 6.5* illustrates the localization of the cross loop based on our shortest path algorithm. Starting from a Reeb loop, we propagate the distance from one side of the Reeb loop to the other side. When we propagate from one vertex to another, we compute the length of the edge between these vertices. Once we hit the second side of the Reeb loop, we propagate from the hit vertex back to the first side of the Reeb loop. To define this path, we move along edges following the decreasing distances of the vertices.

## 6.5 Shortest path algorithm

Compared to the shortest path algorithm of [Wood *et al.*(2003)], we make the following contributions. First, we improve the accuracy with the computation of the edge lengths. Second, we reduce the complexity from $O(n^2\ log(n))$ to $O(n^2)$ for an image with $n^3$ voxels.

### 6.5.1 Improved accuracy

Wood et al [Wood *et al.*(2003)] assign an equal length to every edge in the isosurface during the shortest path computation. However some edges can be much longer than others. For instance, the diagonal edges are generally longer than other edges. Therefore the resulting path can be very different than the actual shortest path over the surface.

We observe that a shortest path algorithm based on unit length edges could lead to a shift of the shortest path along diagonal edges. Indeed if the diagonal edges are visited first, the path will preferrably propagate along these edges. This results in a path longer than necessary.

Therefore we take into account the Euclidean length of the edges, and propagate the distances over the edges of the isosurface.

### 6.5.2 Reduced complexity

As a second contribution, we take advantage of the structure of the isosurface mesh, and propagate the distances from one front of vertices to the next. We thus avoid to sort the vertices in a queue. Therefore the complexity of the algorithm is reduced from $O(m\ log(m))$ to simply $O(m)$ where $m$ is the number of vertices in the isosurface. Since $m$ is proportional to $n^2$ for an image with $n^3$ voxels.

Sorting the vertices in a queue would guarantee to find the shortest path along the edges. However experiments showed that our front propagation without sorting reasonably approximate the actual shortest path.

## 6.6 Closing the loop

Similarly to the shortest path algorithm, we propose to take into account the length of edges to close the loop. The shortest path algorithm defines a path from one side of the Reeb loop to the the other side. Generally, the ends of this path do not match the same point on the Reeb loop, thus the path is open at its ends.

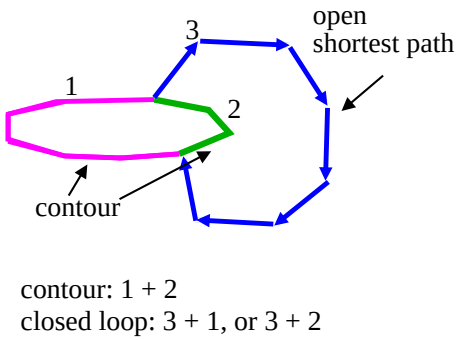


Figure 6.6: *We close the shortest path along the contour to create a closed loop around the hole. The blue arrows and the number 3 indicate the shortest path. The contour is divided in two sections between the ends of the shortest path: the magenta and green closed line with numbers 1 and 2. To close the shortest path, we can merge the shortest path with either the magenta part or the green part. We prefer the green part is it is shorter than the magenta part.*

Following the method of Wood et al [Wood *et al.*(2003)], we insert a section of the Reeb loop to connect both ends of the shortest path. Therefore the shortest path is now closed, and the cross loop is created.

Since we could move along two directions along the Reeb loop, we need to choose one section of the Reeb loop or the other to close the shortest path. We propose to compute the length of the Reeb loop edges, and to choose the section with the shortest length.

*Figure 6.6* illustrates how the shortest path is closed to create a loop. The shortest path in blue can either be merged with the magenta section of the contour or the green section of the contour. Since merging the shortest path with the green section results into a shorter loop than merging the shortest path with the magenta section, we prefer this solution.

## 6.7 Reducing the complexity

To reduce the complexity, we propose to only search for a loop in a limited region around the hole.

The wireframe box in *Figure 6.7* illustrates the limited region where the shortest path operates. We start from the bounding box of the start contour, and extend this volume by an offset length along every cardinal axis. The offset length depends on the length of the shortest candidate loop lo-

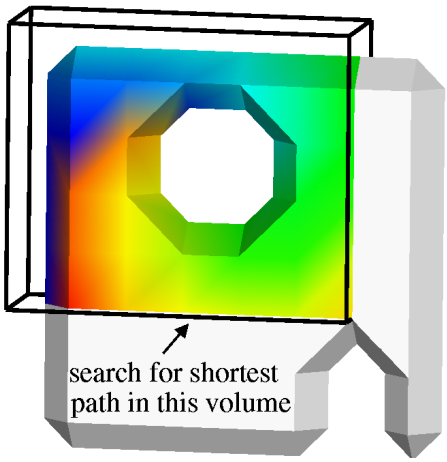


Figure 6.7: *Our shortest path algorithm only explores a limited volume to locate a loop around the hole. Therefore the size of the data in memory remains low, even for very large images. The volume where the shortest path algorithm operates is represented with a wireframe box in the image.*

calized so far since we only search for new candidates that are shorter than this loop.

Since the start contour is a candidate loop, we do not need to search for a loop longer than this loop. The round trip from the start contour to a point outside the bounding box would be longer than twice the offset distance. We only extend the bounding box by half the length of the start contour. Hence any path that exceeds the extended bounding box would be longer than the start contour.

Before starting the search for a path, we check that the second contour is at least partially contained within the search volume. In case all edges of the second contour are outside the search volume, we do not need to start the search.

## 6.8 Conclusion

In this chapter, we have made a number of improvements to the method of Wood et al [Wood *et al.*(2003)] to localize the hole with non-separating loops. First we localize the hole with four loops instead of two to improve the selection of the best loop. Second we find a better approximation of the shortest loop with the computation of the edge lengths. Third we reduce the complexity of the shortest path algorithm from $O(m\ log(m))$ to $O(m)$ when we search over a surface with $m$ nodes. Fourth we make the complexity of the method independent from the size of the image by restricting the computations in a limited volume around the hole. These contributions provide a faster and more accurate loop computation.

The next chapter will describe how we select one loop among the contour loops, the Reeb loop, and the cross loop, and how we correct the hole based on the selected loop.

# Chapter 7

# Topology Simplification

In the previous chapter, we described the localization of the hole with four loops: two contour loops, one Reeb loop, and one cross loop. The present chapter explains how we select one of these loops, and how we use this loop to remove the hole. Since our goal is to modify the smallest number of voxels, we propose to select the loop based on this criterion.

We will first explain the principle of hole correction based on a loop. Then we will present how we compute the number of voxels that will be modified for every loop. Finally we describe the selection of the best loop, and the actual correction in the image.

## 7.1 Structure of the algorithm

*Figure 7.1* outlines the steps of our algorithm for the simplification of topology. The input is a number of loops around the detected hole. Every loop defines a different region, where the modification of the voxels would result in the removal of the hole. We select the loop that would modify the minimum number of voxels. Then we modify these voxels, in an operation called rasterization. At the output, only a local region of the image has been locally modified to remove the hole.

## 7.2 Principle of hole removal

To remove a hole, we modify the voxels inside a loop that encloses the hole. The principle of hole removal based on a loop builds upon the idea of Guskov and Wood [Guskov and Wood(2001)]. To remove holes on surface

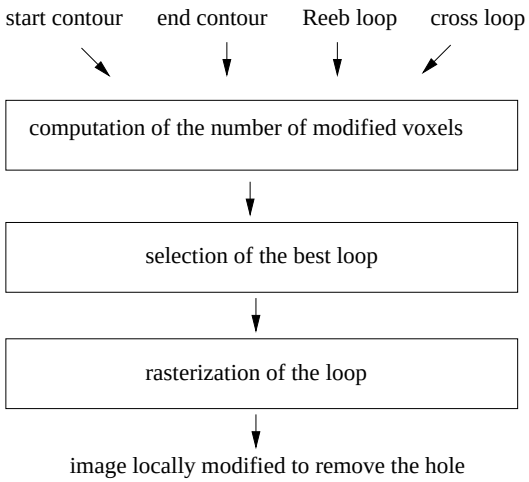


Figure 7.1: *To remove a hole in the image, we select the loop that encloses the smallest number of voxels, and then rasterize these voxels. We could choose between the contour loops, the Reeb loop, and the cross loop around the hole. For every loop, we compute the number of voxels that would be modified if the loop was chosen. We then select the loop with the smallest number of modified voxels. Finally we rasterize the region inside this loop from every loop point to the center of the loop.*

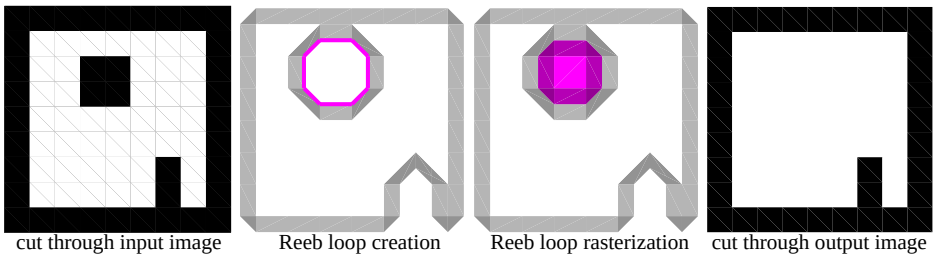


Figure 7.2: *Filling a loop around a hole removes the hole from the image. The first image shows the hole in the input image. The second image indicates the Reeb loop in magenta on the isosurface. In the third image the volume rasterized inside the loop is colored in magenta. Finally the rasterization of the loop removes the hole from the image.*

meshes, they first define a loop on the surface around the hole. Then they cut the surface mesh, and seal both sides of the cut. The hole has thus been removed from the surface. Similarly, to remove a hole in an image, Wood et al [Wood *et al.*(2003)] define a loop on the isosurface of the image, and then modify the image region inside the loop.

Wood et al [Wood *et al.*(2003)] propose to modify the region inside the loop in the image. If the region is made of object voxels, it must be replaced with background voxels. Conversely, if the region is made of background voxels, it must be replaced with object voxels. In the first case, the loop is 'emptied', while in the second case, the loop is 'filled'.

*Figure 7.2* illustrates that a hole in the image can be removed with the creation of a loop around the hole. First the cut through the input image indicates a hole inside the image. Second a Reeb loop shown in magenta is created around the hole on the isosurface. Third the volume inside the Reeb loop is rasterized in the image. Finally the rasterization in the image modifies the image so that the hole is removed.

*Figure 7.3* illustrates the removal of the hole by emptying the region inside a loop. The hole in the 3D image is visible in the cut through the input image. The cross loop is then localized on the isosurface of this image as shown in the second image. The volume inside this loop is rendered and colored in yellow in the third image. Finally the hole inside the image is removed by the rasterization of the cross loop.

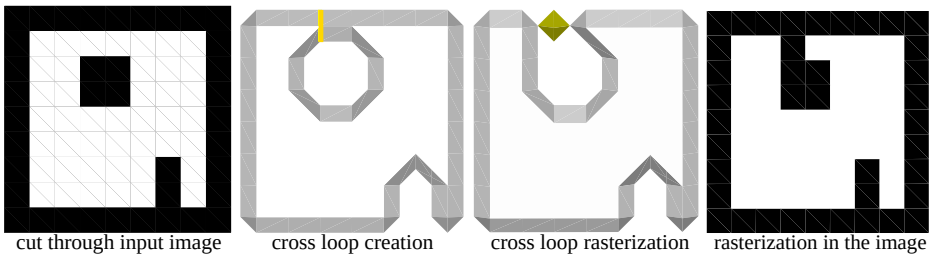


Figure 7.3: *We can remove a hole in the image by emptying a loop in the image. The first image shows the hole inside the image. The second image indicates the cross loop in yellow across one side of the hole. In the third image the region inside the cross loop is colored in yellow. Finally the rasterization of the cross loop removes the hole from the image.*

## 7.3 Filling or emptying the loop

We need to find if a loop goes around background or object voxels, so we can determine to either fill or empty the volume inside the loop. Since the contours lie in a plane, we can use the orientation of the contour edges to find the type of voxels inside the contour. Then we deduce the type of voxels for the Reeb loop and the cross loop. Therefore we define a set of rules based on the signed areas of the contours.

### 7.3.1 Signed areas of contours

The signed area of a contour indicates its orientation. To compute the signed area of a contour, we proceed as follows. First we pick an arbitrary point in the plane of the contour and call it the origin. Then we visit every point in the contour following their orientation. We create the vector from the origin to this point, and the vector from the origin to the next point in the contour. We then compute the cross product of these two vectors, and move to the next point in the contour.

Since every vector lie in the XY plane of the contour, the cross products either points to the positive Z or to the negative Z. The sum of the cross products has thus only one non-zero component along the Z axis. This value is the signed area. Depending on the orientation of the contour, the signed area is positive or negative.

We build the contour as a sequence of edges from the Marching Cubes algorithm. Based on our construction of the contour, the signed area is

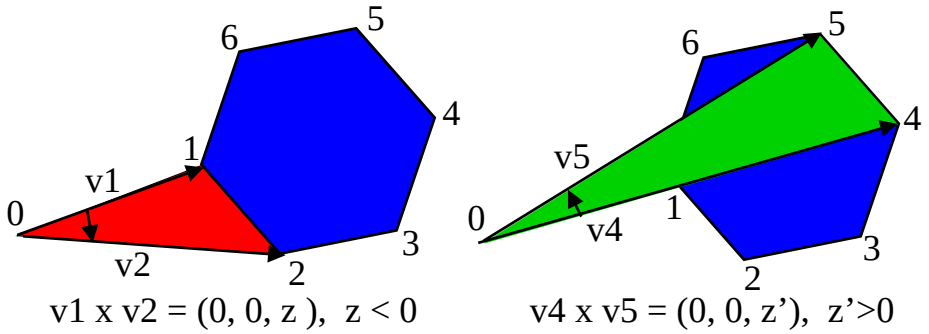


Figure 7.4: *To decide whether a contour needs to be filled or emptied, we deduce its orientation based on the sign of its signed area. To compute the signed area of the contour shown in blue, we create the vectors from an arbitrary point 0 to two successive points 1, 2, 3, 4 and 5 in the contour. The cross product of vectors (0,1) and (0,2) corresponds to the area of the pink triangle in the left figure and has a negative sign. On the other hand the cross product of vectors (0,4) and (0,5) defines the area of the green triangle and has a positive sign. When we sum the triangle contributions, we obtain the signed area of the contour.*

positive when the contour encloses object voxels, and is negative when it encloses background voxels.

*Figure 7.4* illustrates the computation of the signed area for a contour with five points 1, 2, 3, 4, 5 shown in blue. In the left image, the z component of the cross product of the vectors (0,1) and (0,2) has an absolute value that defines the area of the pink triangle and a negative sign that defines the relative orientation of the vectors. Similarly, in the right image, the z component of cross product of the vectors (0,4) and (0,5) corresponds to the green triangle and has a positive sign. We compute the cross product for every pair of successive points in the contour and sum all these signed values to obtain the signed area. The signed area is positive if the contour encloses object values, and negative if the contour encloses background values.

### 7.3.2 Reeb loop and cross loop

We now deduce the type of voxels inside the Reeb loop and the cross loop based on the orientation of the two contours. Since each contour can have two orientations, we have a number of four configurations.

If both contours used for the detection of the hole go around object voxels, the Reeb loop goes around background voxels. Conversely, if both contours go around background voxels, the Reeb loop goes around object voxels.

For the last two configurations, we need to find which contour is contained within the other one. To do this we use the absolute value of the signed areas. The contour with the smallest area is thus contained within the contour with the largest area.

If the largest contour goes around object voxels, and the smallest contour goes around background voxels, the Reeb loop goes around object voxels. Conversely, if the largest contour goes around background voxels, and the smallest contour goes around object voxels, the Reeb loop goes around background voxels.

Since the cross loop is transverse to the Reeb loop, the type of voxels inside the cross loop is the inverse of the voxels inside the Reeb loop. The following pseudo-code summarizes these rules.

```
F : object value
B : background value

Known:
A_start := signed area of start contour
A_end := signed area of end contour
Convention: |A_end| > or = |A_start|

Unknown:
V_Reeb := voxel value inside Reeb loop (F or B)
V_cross := voxel value inside cross loop (F or B)

if A_start > 0 and A_end > 0, V_Reeb := B
if A_start < 0 and A_end < 0, V_Reeb := F
if A_start < 0 and A_end > 0, V_Reeb := F
if A_start > 0 and A_end < 0, V_Reeb := B

V_Cross = not( V_Reeb )
```

## 7.4 Rasterization inside the loop

The conversion of a mesh into a set of voxels is called rasterization. We use this term to define the following operations: the conversion of the

region inside the loop into a set of voxels and the modification of these voxels in the image.

As an analogy, we consider the image as a recipient where the object in the image is surrounded with water. To remove the hole in the object, we need to find a thin watertight layer to seal the hole. This layer defined by a set of voxels in the image must disconnect the voxels on both sides of the loop. Since our goal is the simplification of the topology with the minimum modification to the image, we need to search for the smallest number of voxels in this watertight volume.

### 7.4.1 Rasterization along lines

We propose to operate from every loop point to the center of the loop, and locate the voxels along these radii. For a given loop point, we move toward the center with steps smaller than the distance between adjacent voxels. At every position, we locate the eight neighboring voxels. The set of located voxels create the watertight volume.

To guarantee that for a flat loop we only insert voxels in the plane, we do not insert voxels outside the bounding box of the loop. Therefore, for a flat loop, we only consider the four adjacent voxels for the positions along the radii.

Since our contour creation needs an isosurface without boundaries, we do not rasterize any voxel on the boundary of the image volume. We padded the image with values below the isovalue before the application of our algorithm. Those voxels on the boundary of the image must keep a value below the isovalue. This guarantees that any contour is a closed polyline, and can be constructed by traversing the contour from one vertex back to this vertex.

### 7.4.2 Comparison with other methods

Our method has a lower complexity than the rasterization method of Wood et al. [Wood *et al.*(2003)]. They define the volume inside the loop as follows. First they build a triangle fan that connects the loop points to the center of the loop. Then they convert this surface mesh representation into a set of voxel.

Instead, we do not create a mesh, but simply rasterize the lines from the loop points to the loop center. Since the spacing between loop points is smaller than the sampling of voxels, it is guaranteed that we do not miss a voxel between the lines.

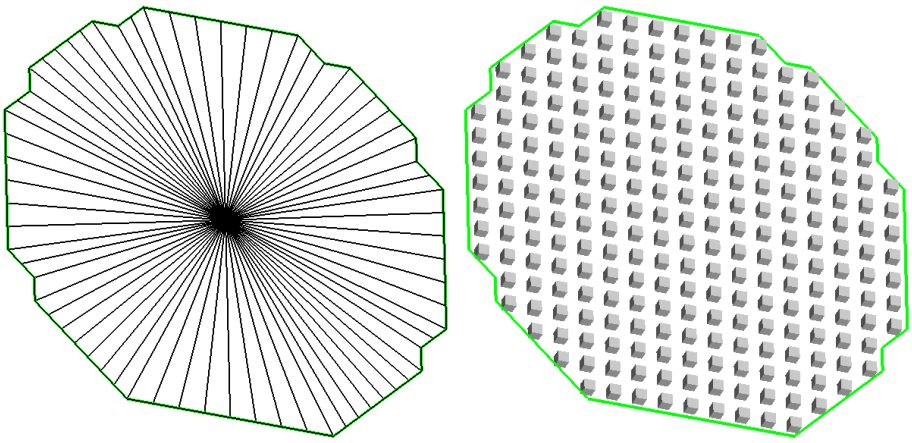

Figure 7.5: *To remove a hole, we propose to change the values of a small number of voxels inside a loop. We move from every loop point to the center of the loop with small increment steps. These radii are shown in the left image. We then locate the voxels around every radius, as represented with small cubes in the right image.*

*Figure 7.5* illustrates our rasterization method inside the loop. The left image shows the radii from every loop point to the center of the loop. The located voxels along the radii are represented in the right image as small cubes.

Unlike mesh correction techniques, the modification of image values is guaranteed to create a two-manifold surface. A two-manifold surface is a surface where the neighborhood of every point can be mapped onto a disk. More practically, every edge on a triangulated two-manifold surface belongs to one or two triangles. A two-manifold surface does not have T-junctions, where one edge belongs to three triangles. In our algorithm, we build a new isosurface after rasterization of the surface. Hence we are guaranteed to recover a two-manifold surface since the isosurface construction rules guarantee this property.

## 7.5 Loop selection

The localization of the hole has provided four different loops, two contour loops, one Reeb loop, and one cross loop. Each one of these loops

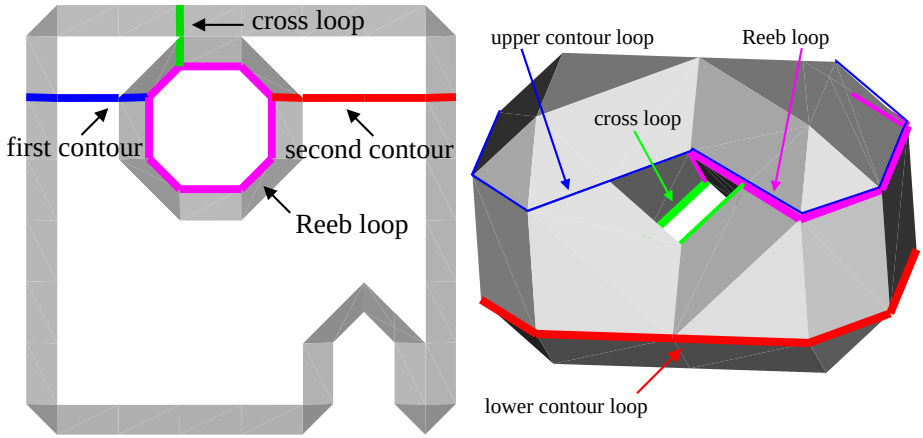


Figure 7.6: *Every candidate loop defines a different region where we could correct the hole. The left image shows the four loops around a hole detected in the graph, while the right image shows five similar loops defined for a hole detected inside a ribbon. To remove each hole, we choose one of these loops.*

could be used to remove the hole from the image. We need a criterion to select the best loop.

Our goal is to minimize the number of modified voxels when removing the holes. Therefore we propose to evaluate how many voxels would be modified for every loop, and select the loop that modifies the smallest number of voxels.

To measure the number of modified voxels, we perform the rasterization of the loop in a copy of the image. Since only a local region could potentially be modified, we only copy the region inside the bounding box of the loop.

We perform the rasterization of the loop inside the copy of the image, and count the number of modified voxels.

*Figure 7.6* shows the set of four loops that we localize around a hole. On the left image, one contour loops or one cross loop could be emptied to remove the hole. Alternatively, the Reeb loop can be filled, and the hole would be removed. For the intra-ribbon hole shown in the right image, we could empty the Reeb loop, the lower contour loop, or the upper contour loop, or we could fill the cross loop.

## 7.6 Correction of gray scale images

To minimize the modifications to a gray scale image, we rasterize with the closest values to the isovalue. The isovalue is the intensity value chosen to separate the object values from the background values. Our isosurface algorithm separates every pair of consecutive voxels, where one has a value above the isovalue, and the other has a value below the isovalue or equal to the isovalue.

Therefore using the isovalue for the rasterization has the same effect as using a lower value. Similarly, any value strictly above the isovalue has a similar effect than any other value strictly above the isovalue. We aim at modifying the image values by the minimum amount. Hence we choose to rasterize with either the isovalue or the isovalue plus an epsilon, where epsilon is the smallest value that produces a new value in a computer representation.

When the image values use an integer representation, we consider the isovalue without its decimal part, and the integer immediately above this value. For instance, if the image is represented with values from 0 to 255 and the isovalue is 127, the rasterization values will be 127 and 128. Most medical images store voxel values with unsigned shorts. The two values are formalized as follows.

$$Background = Isovalue \tag{7.1}$$
$$Object = Isovalue + \epsilon \tag{7.2}$$

### 7.6.1 Updating the graph

When a hole has been corrected, the image has been modified, and the graph does not represent the image anymore. However, it is not worth starting over the construction of the graph from the bottom of the image. We can preserve the graph nodes below the modified voxels.

**Locating the region to preserve** We first locate the lowest plane where a voxel was modified. This corresponds to the lowest voxel in the correction loop. We call this height $h$. The ribbons between the plane at height $h$ and the plane at height $h-1$ are also modified. We thus go down to the plane $h-1$, and rebuild the ribbons.

**Inserting nodes in the updated graph** We need to connect the new ribbon nodes with the contours nodes we have preserved in the graph. We thus build the new ribbons from contour seeds. We use the contours in plane $h-1$. This creates the connections between the contours in the plane $h-1$, and the ribbons between planes $h-1$ and $h$. Then we build the new ribbons that are not connected to any contour in plane $h-1$.

### 7.6.2 Avoiding the halting problem

The correction of a hole could cause the creation of another hole. If one hole is located in the object, the other would be located in the background. Then the algorithm could go back and forth between the correction of each hole, and never terminates. This is called a halting problem.

To avoid a halting problem, we impose only one type of correction when the algorithm seems to have fallen into a loop. We do not strictly detect that the algorithm alternates between filling one hole and breaking one handle. However, we assume a halting problem when we explore one same plane of the image a large number of times.

For every plane in the image, we count how many times we explore its voxels. A plane is re-explored every time the correction of a hole modifies at least one of its voxels. In case the algorithm alternates between the correction of the same pair of holes, the corresponding planes would be repeatedly explored and the number of explorations would increase.

To avoid that the exploration of the same planes goes forever, we stop the algorithm when the number of explorations exceeds a given threshold, and we fall in a 'safe mode'. Indeed, we only allow the correction of a hole by filling its interior, thus the algorithm cannot alternate with emptying a handle. Since the correction of a hole always increases the number of object voxels, the algorithm is guaranteed to end, in the worst case, when every voxel of the image is an object voxel. Similarly we could avoid halting problems by only allowing to empty loops.

## 7.7 Conclusion

In the present chapter aiming at simplifying the topology, we have proposed to remove a hole based on a loop. In a process called rasterization, if the loop encloses object (respectively background) voxels, these voxels are replaced with background (respectively object) voxels. We have presented a set of rules to define whether the loop encloses object or background voxels.

We have made the following contributions in the rasterization of the loop. First we propose a rasterization method with a low complexity: we move from every loop point to the center of the loop, and modify the voxels neighboring these lines. Second we propose to improve the accuracy by selecting among the contour loops, the Reeb loop, and the cross loop the loop that modifies the smallest number of voxels. Third to minimize the modification to a gray scale image, we propose to replace the voxel values inside the loop with one of the two closest values around the isovalue.

The following chapter will present the results of the algorithm that we developed based on the principles and the methods described in Chapters 5, 6 and 7.

# Chapter 8

# Results

Chapters 5, 6, and 7 of the present work have explained the principles and the methods driving the development of our topology simplification algorithm. The present chapter will now prove the concept by applying the algorithm to create a brain representation with correct topology from a segmented scan. Firstly, we will show the resulting image, and compare its appearance with the input image. Secondly, we will visualize the corrections to the image. Thirdly, we will report some statistics about the corrections. Finally, we will discuss the results.

## 8.1 Goal of the experiment

We want to remove the holes in a segmented brain image to obtain a representation of the brain cortical surface as a single folded sheet, since the actual brain surface follows such a folded sheet. We assume that the segmentation describes the global shape of the brain, and we want to preserve as much as possible of this shape while removing every hole. As a measure of how much of the brain shape is preserved we propose to count the number of modified voxels after the removal of holes.

Practically we formulate our goal as the removal of holes with the modification of the smallest number of voxels in the image. The measure to assess the success of the method is thus the number of modified voxels in the image. This number should be the smallest possible.

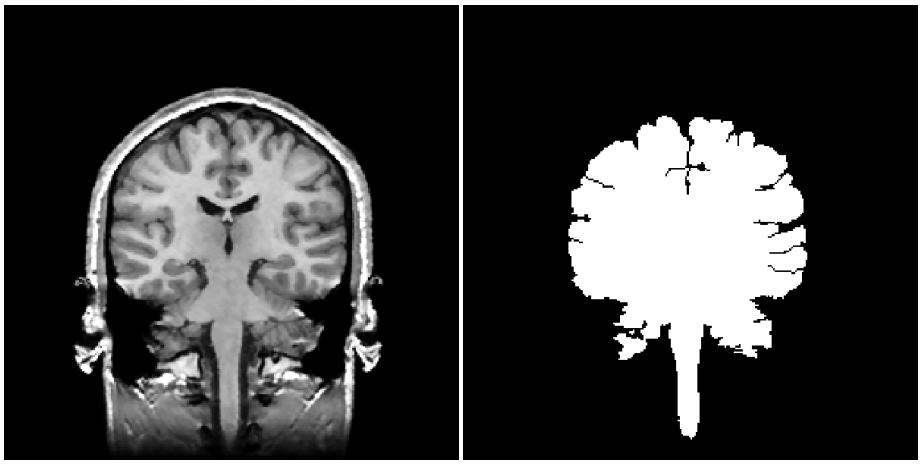

Figure 8.1: *At the input of our algorithm, we use a manually segmented brain scan of a healthy subject. The left image shows one cut through the Magnetic Resonance Imaging scan. On the right image the corresponding cut through the segmented image is displayed. In this image, the gray matter, the white matter, and all other structures inside the brain appear in white.*

## 8.2 Input and output of the algorithm

The input image was a brain segmentation of a head Magnetic Resonance Imaging scan. The scan was acquired for a healthy subject, and had a resolution of 256 x 256 x 124 voxels with a spacing of 1 mm x 1 mm x 1.5 mm. It was segmented by a group of experts as part of the Digital Brain Atlas project [Kikinis *et al.*(1996)].

A number of structures inside the brain were classified with different values. Therefore, as a pre-processing step, we binarized the image. In the binary image, the object voxels represented the brain gray matter, the white matter, and the other structures inside the brain. *Figure 8.1* shows one cut through the MRI scan (left), and the corresponding cut through the segmented image (right).

The brain volume slightly exceeded the volume of the scan on the first and last planes. Besides, the brain stem was visible on the boundary of the image since it extended to the spine. To close the boundaries of the brain isosurface, we padded the brain segmentation with background voxels. The computation of the Euler characteristic (see Chapter 3) for the resulting isosurface indicated that this representation exhibited 91 holes.

| number of voxels | input image | output image | processing time |
|---|---|---|---|
| 256 x 256 x 124 = 8.1 $10^6$ voxels | 91 holes | 0 hole | 2 min 15 sec |

Table 8.1: *Our algorithm only takes 2 min 15 sec to remove every hole in a real size brain image. Therefore it can be applied for common medical applications.*

The output of the algorithm was a binary image of the brain, where every hole was removed. The difference between the input and the output images was a small number of voxels changed to either foreground or background. Our goal was to obtain the smallest number of different voxels possible.

*Table 8.1* reports the size of the input data, and the time taken by our algorithm. The size of the input data, 256 x 256 x 124 voxels, corresponds to the typical size of a brain scan. The algorithm can thus process real size images. Besides the time taken to process this image is small enough to be acceptable in most clinical applications. Therefore the algorithm can be used routinely to help the doctor in the analysis of medical scans.

## 8.3 Visualization of the output image

*Figure 8.2* shows the brain surface before and after the application of our algorithm. The left image renders the brain isosurface extracted from the input image, while the right image renders the brain isosurface extracted from the output image. Although a number of voxels have been modified to remove the holes between the input and the output images, one can see that both surfaces have the same appearance. This indicates that our algorithm has globally preserved the brain shape.

One visible difference between the two surfaces is the correction of the connection between the hemispheres. The red square on the left image highlights a hole in the input image. On the left surface, this hole creates a bridge between the hemispheres. This hole is removed on the right surface and the shape of the brain folds in the neighborhood of the hole is not significantly modified.

*Figure 8.3* illustrates one advantage of the removal of holes in the brain segmentation: the output isosurface can be inflated to reveal the region inside the brain folds. The left image shows that the inflation of the input isosurface is limited due to the holes that keep both sides of some brain

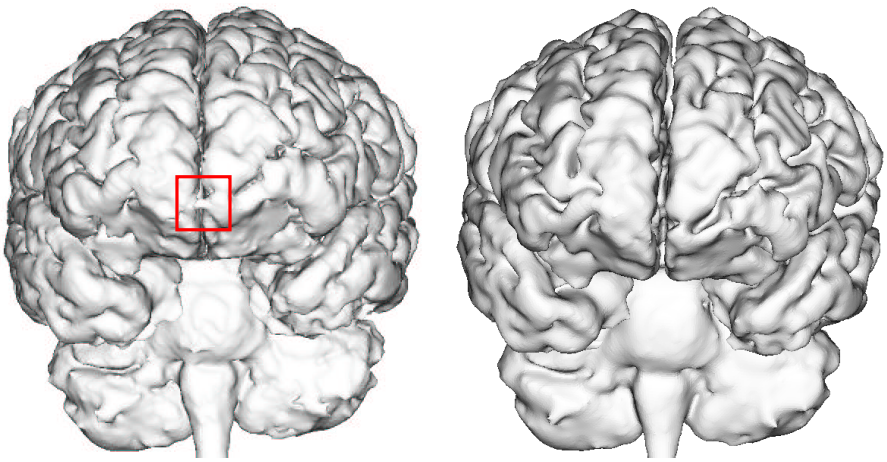

Figure 8.2: *Our algorithm removes the holes and preserves the complex shape of the brain folds. The brain folds on the isosurface of the output image (right image) have the same appearance as the isosurface of the input image (left image). However the holes, such as the one shown in the red square, are removed on the output isosurface.*

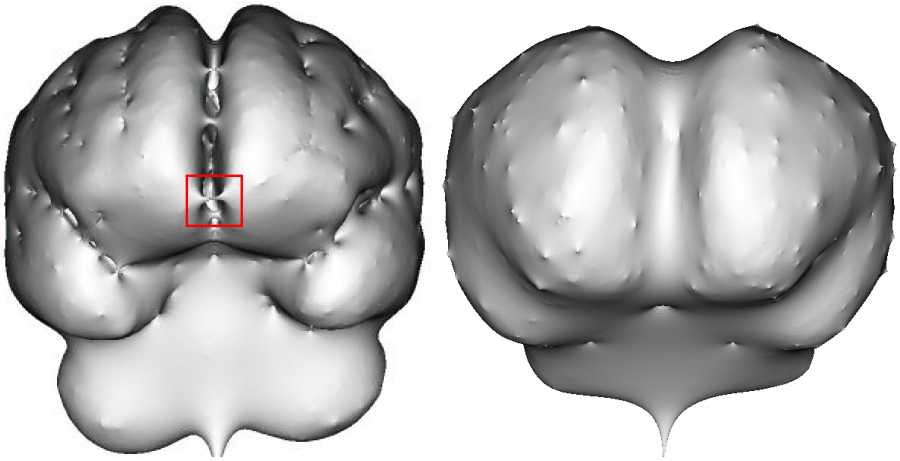

Figure 8.3: *With the removal of holes, the brain surface can be inflated to visualize the regions inside the brain folds. The left image shows that because of holes, such as the one in the red square, the input isosurface cannot be completely inflated. However the right image indicates that, with the application of our algorithm, the output isosurface can be inflated.*

folds close together. The application of our algorithm removes these holes, and thus the brain surface has the correct representation of a single folded sheet. The right image shows the inflated output isosurface. The entire cortical surface can be visualized on this inflated surface mesh.

## 8.4 Visualization of the modified voxels

To remove the 91 holes from the input image, our algorithm modifies the value of some voxels in the image. We visualize the modified voxels to show their size relatively to the brain volume, and their distribution in the brain.

### 8.4.1 Differences in the volume

*Figure 8.4* illustrates the modifications for a front view (left) and a lateral view (right) of the brain. The images shows a rendering of the modified voxels in color, and a semi-transparent rendering of the brain surface. The blue and red spots corresponds to the two types of modifications: changing the value of a voxel from background to object, or changing the value of a voxel from object to background. These modifications respectively insert the voxel into the brain volume, or remove the voxel from the brain volume.

We can see that the modified voxels form a number of clusters in the image. Compared to the size of the brain, the size of the clusters is relatively small. Besides, they approximate a uniform distribution over the image. Therefore, this indicates that the modifications to the image do not cause major changes to the shape of the brain.

### 8.4.2 Differences on a plane

*Figure 8.5* shows the difference between the input and the output images for the 30th and 35th planes extracted from the 124 planes of the brain image. The voxels with a different value between the input and output images are displayed with the same color-coding as *Figure 8.4*: red for values changed to object, blue for values changed to background. We observe that the modified voxels only change small connected areas, in different regions on the planes. Therefore the modifications do not significantly affect a given plane.

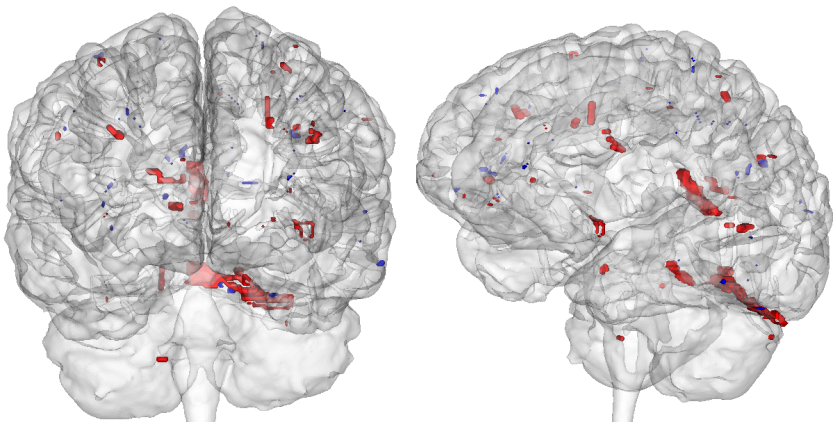

Figure 8.4: *To remove the holes from the images, our algorithm only modify a small number of voxels in the image. Besides they appear uniformly distributed over the brain volume. The images renders the modified voxels in color superimposed on a semi-transparent brain surface on a front view (left) and a lateral view (right). The voxel values changed from background to object are colored in red, while the voxel values changed from object to background are colored in blue.*

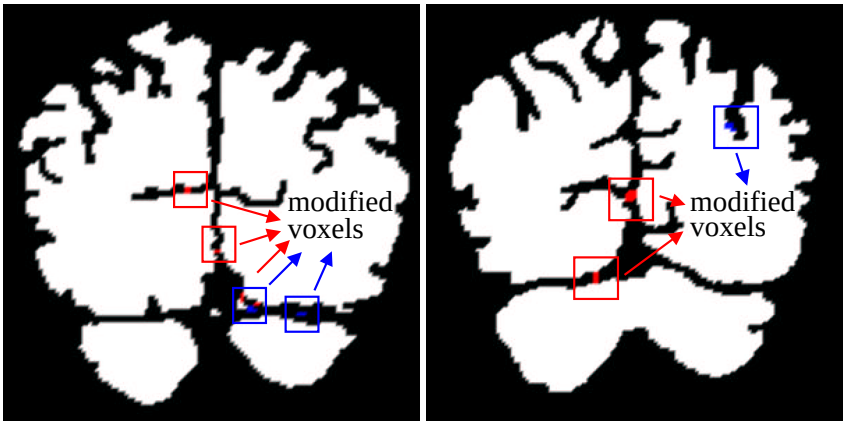


Figure 8.5: *Within a given plane, few voxels are modified and they do not describe large connected areas. The left and right images are respectively the 30th and 35th cuts extracted from the 124 cuts. The modified voxels are colored in red when the voxel value is changed to the object, and in blue when it is changed to the background.*

| | number of voxels | fraction of the image |
|---|---|---|
| voxels in image | 8.126.464 | 1.0 |
| voxels modified | 1486 | $1.8\ 10^{-4}$ |
| new object voxels | 1320 | $1.6\ 10^{-4}$ |
| new background voxels | 166 | $0.2\ 10^{-4}$ |
| largest single modification | 299 | $0.4\ 10^{-4}$ |

Table 8.2: *The number of modified voxels is relatively small compared to the total number of image voxels. The algorithm has modified 1486 voxels, i.e. a fraction of* $1.8\ 10^{-4}$ *in the* $8.1\ 10^{6}$ *voxels in the image. Among the 1486 voxels, 1320 correspond to a value changed to object, while 166 correspond to a value changed to background. The largest correction corresponds to the modification of 299 voxels.*

## 8.5 Statistics

We now report some statistics about the number of modified voxels, and compare these figures with the total number of voxels in the image. These statistics assess the success of our algorithm: the simplification of topology under the constraint to preserve as much as possible of the input brain segmentation.

### 8.5.1 Number of modified voxels

*Table 8.2* shows that the number of voxels modified by our algorithm is relatively small compared to the total image voxels. The input image has 256 x 256 x 124 = $8.1\ 10^{6}$ voxels. The algorithm has modified 1486 voxels, i.e. a fraction of $1.8\ 10^{-4}$ of the image voxels. The modified voxels can be classified into the voxels changed to object and the voxels changed to background. The first set has 1320 voxels, while the second set has 166 voxels. For every hole, a number of voxels are modified. For one hole, this number has reached 299 voxels, which is a relatively large value. This indicates that another solution could be better.

### 8.5.2 Loops

*Table 8.3* reports some statistics about the loops used to remove every hole. It shows that 109 loops are required to remove 91 holes. This means that the correction of some holes has created new holes. Nevertheless,

| | repaired loops |
|---|---|
| total | 109 |
| holes detected in the graph | 84 |
| contour loops | 42 |
| Reeb loops | 18 |
| cross loops | 24 |
| holes detected in a ribbon | 25 |
| Reeb loops | 0 |
| cross loops | 25 |

Table 8.3: *More loops (109) than holes (91) were needed to remove every hole. This means that the correction of some holes created new holes, but the total number of loops is still reasonable. First to remove the 84 holes detected in the graph, we selected 42 times a contour loop, 24 times a cross loop, and 18 times a Reeb loop. Contour loops are often selected because they only modify voxels in a single plane. The fact that cross loops are generally preferred to Reeb loops justifies to search for the cross loop after the Reeb loop. Second to remove the holes inside a ribbon, we always selected a cross loop. This means that our wavefront propagation inside a ribbon could be improved to result in a shorter Reeb loop.*

these additional holes are all corrected, and the total number of 109 loops is still reasonable. Among the 109 loops, 84 are created based on the detection of a hole with the graph, and 25 are based on the detection of a hole in a ribbon.

For every hole detected in the graph, we select either a contour loop, a Reeb loop, or a cross loop based on the minimal modification in the image. Among the set of 84 loops, the selected loop is a contour loop for 42 loops, it is a Reeb loop for 18 loops, and it is a cross loop for 24 loops. The contour loop is selected with the highest frequency because it often modifies less voxels or as many voxels as the corresponding Reeb loop and cross loop. Besides more cross loops are selected than Reeb loops.

On the other hand every hole detected inside a ribbon is corrected with a cross loop. The Reeb loop we compute is always longer than the cross loops in our experiment.

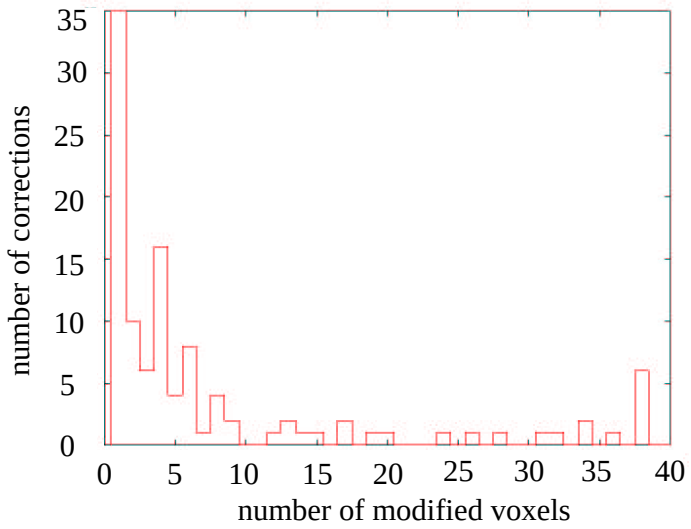


Figure 8.6: *Generally our algorithm only modifies 1 to 8 voxels to remove a hole. The histogram represents the number of modified voxels along the x axis, and the number of corrections along the y axis.*

### 8.5.3 Size of the corrections

We now observe the size of the corrections: whether the correction of a single hole requires the modification of a large or a small number of voxels. We provide two measures for the size of the corrections: the number of modified voxels, and the number of loop edges. The number of modified voxels evaluates the volume of the correction, while the number of loop edges indicates the area of the correction. We present an histogram of the number of corrections based on each of the two measures.

**number of modified voxels** *Figure 8.6* represents the number of modified voxels along the x axis, and the number of corrections along the y axis. The largest number of corrections are concentrated between 1 and 8 voxels. For 35 corrections a single voxel is modified, and for 15 other corrections 4 voxels are modified. The number of corrections drops when the number of modified voxels exceeds 8. Since the volume for one voxel is 1.5 $mm^3$, the volume occupied by 8 voxels in the brain image is only 12 $mm^3$.

The histogram shows that the algorithm removes each hole with the

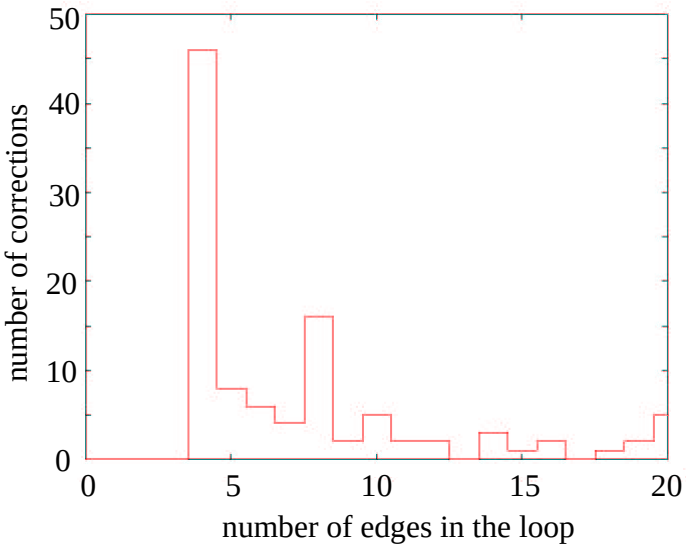


Figure 8.7: *Most of the holes are corrected with a loop that only has less than 8 edges. The histogram shows the number of loop edges along the x axis, and the corresponding number of corrections along the y axis.*

modification of only a small number of voxels. Indirectly, this indicates that the holes in the volume are small, and justifies our assumption that the modification of the region inside the holes do not significantly affect the brain volume.

**length of the loops** *Figure 8.7* shows the histogram of the number of corrections (y axis) with respect to the number of edges in the loop (x axis). When correcting a hole, the chosen loop has generally less than 8 edges. The loop chosen for 45 corrections has only 4 edges, i.e. the minimum number of edges for a loop in the image. For 15 corrections, the chosen loop has 8 edges. With a voxel spacing of 1 mm x 1 mm x 1.5 mm, the area of a loop with 8 edges is at most 10.5 $mm^2$. These small numbers indicate that the correction of a hole do not cause the modification of a large area in the image.

## 8.6 Discussion

We discuss the results presented in the previous section, and assess the assumption made for the development of the algorithm. Besides, we highlight the current limitations of the algorithm, and propose some improvements.

### 8.6.1 Assessment of the method

The results show that the algorithm achieves the goal of simplifying the topology and preserving the global shape of the brain. Indeed the 91 holes are corrected, and only a relatively small number of voxels well distributed over the brain volume are modified. Therefore our algorithm would be useful for applications that require a brain representation that is both geometrically accurate, and topologically correct (i.e. without holes).

For applications in the treatment of a patient, it is critical that the modifications to the image do not affect the interpretation of the image. The visualization of all modified voxels shows that the modifications are almost uniformly distributed over the brain volume. Consequently the modified voxels do not form large clusters that could significantly deform the representation of the brain. Therefore the doctor should be able to give a similar interpretation using either the input image or the output image. Moreover the correct topology of the brain in the output image should improve the interpretation of the image.

### 8.6.2 Further experiments

Further validation experiments are needed to validate the method. The algorithm should be applied on a significantly large number of segmented brain scans to gather more data and improve the statistics. The images from different segmentation methods could be used to find if the algorithm works better with some segmentation methods than with others.

In new experiments, we would like to investigate the combination of our automated topology simplification with automated segmentation methods. The combination could provide an automated solution in many applications where the time to manually segment a brain scan and correct the holes would be prohibitive. The automated segmentation and topology correction could be particularly useful for intra-operative imaging.

### 8.6.3 Comparison procedure

To compare different topology simplification algorithms, we suggest to apply the algorithms to a common set of brain images. With a sufficiently large set of images, every algorithm could be assessed with reliability and the results could be compared.

To assess the quality of the algorithms, a doctor could review the results. The anatomical knowledge of the doctor would be very valuable to decide which correction is preferred to perserve the anatomy of the brain. This experiment could be useful to find the algorithm that gives the best result for a specific applications.

Alternatively, we could artificially generate images with holes. We observe that white noise in an image tends to create a large number of holes. We could thus apply different levels of noise to an image, and compare how the algorithm removes the holes in the image.

The advantage of the later comparison procedure is the objective measure of the ground truth. To compare the algorithms, we could measure the similarity between the image without noise and the application of an algorithm on a noisy image. The number of different voxels could be a reasonable metric to assess the quality of the algorithm. We could also compare the isosurfaces extracted from both images, and apply a metric such as the Hausdorff distance. The Hausdorff distance sums the distances from every vertex from one isosurface to the other.

### 8.6.4 Selection of the best loop

We now discuss the results shown in *Table 8.3* first for the holes detected in the graph, and then for the holes detected in a ribbon. For a hole detected in the graph, the table justifies our method to successively search for two contour loops, one Reeb loop, and one cross loop. To remove the hole, we select the loop that modifies the smallest number of voxels in the image.

*Table 8.3* shows that a contour loop is generally selected. This can be explained as follows. For very small holes, the contour loop often has only four edges. Since only one voxel would be modified with such a small loop, the other candidate loops, the Reeb loop and the cross loop, are not computed. Besides, for a contour loop, only voxels in a single plane are modified. Therefore the number of modified voxels is often smaller than arbitrarily oriented Reeb loops and cross loops.

However, taken together the Reeb loops and the cross loops are se-

lected as often as the contours loops (42 Reeb loops and cross loops, and 42 contour loops). This justifies the benefit of searching for a better loop than a contour loop. More cross loops than Reeb loops are chosen since they are built with less constraints, and are thus generally shorter. Therefore this justifies that we search for a cross loop, which is based on the Reeb loop.

For a hole detected in a ribbon, only cross loops were selected. This indicates that our computation of the Reeb loop could be improved. To create shorter Reeb loops we could modify the face by face traversal so the fronts only have a few edges along the propagation.

### 8.6.5 Other criteria for the hole removal

In this work, we have assumed that the best hole corrections should modify the smallest number of voxels. This assumption is motivated by the fact that we want to preserve the global shape of the brain in the segmented image.

Nevertheless, other criteria could offer interesting solutions. Some corrections could be better in some regions of the brain based on the brain anatomy. As proposed by Ségonne et al [Ségonne *et al.*(2003)], we could use a criterion based on some a priori information. For every voxel in the image, we need a map of probability that the voxel belongs to the object or to the background. Then for every loop we have localized, we compute the probability inside the correction region. A correction in a region with low probability of being object or background would be preferred to a correction in a region with a high probability. Hence the modified volume could be located in a region where the modifications have a smaller impact on the brain anatomy.

For other applications, it is important to preserve the smoothness of the surface. For instance, many Computer Graphics applications require the parameterization of the surface onto a plane. The parameterization is improved when the holes in the surface are removed and when the surface is smooth. Therefore to select the best correction we could use a criterion based on the minimization of the curvature of the resulting surface.

As an approximation to the smoothness of the output surface, we could test the shape of every correction. For instance, we could measure an aspect ratio based on the ratio of the volume over the area of the correction. A smooth correction would have a small value for this aspect ratio. This criterion would be less complex than extracting the output isosurface and computing its curvature.

### 8.6.6 Dependence on the direction of propagation

The direction of propagation of our algorithm was chosen in the z direction to minimize the access to the data. Generally the volumetric images are stored one plane after the other. Therefore it is less expensive to access the voxels plane after plane, rather than at arbitrary locations in the volume. A sequence of planes could be loaded in memory and all the voxels in these planes would be read before the next sequence of planes is loaded.

However a potential dependence of the algorithm on the direction of propagation should be evaluated. We propose the following procedure to measure this dependence based on a permutation of the image axes. First we would apply our algorithm to the input image. Second we would permute the x axis and the z axis, apply the algorithm, and permute again the x and z axes. Third we would permute the y axis and the z axis, apply the algorithm, and permute again the x and z axes. Therefore the output images would be in the same reference frame. We would then test if the modifications to the image are different between the two permuted images and the non-permuted image.

We believe that the dependence on the direction of propagation would not be significant since our algorithm is based on the localization of arbitrarily oriented loops. Indeed we search for the shortest loop along the isosurface in any direction. It can happens that two candidate loops are equally good. In this case the algorithm would select the first loop it has computed. For instance the Reeb loop and the cross loop in the input image could respectively correspond to the cross loop and the Reeb loop in the permuted image. One loop will be selected in the first image, while the other will be selected in the second image. Therefore we could notice a small dependence on the direction of propagation.

### 8.6.7 Correction of large holes

Generally, the algorithm removes each hole with the modification of a small set of voxels. However, for one hole, it has required to modify 299 voxels, i.e. a volume of 447 $mm^3$. Since this modification forms a large cluster in the image, an alternative solution could be better. The software could be adapted to switch to an interactive mode when such a large hole is discovered. Alternatively we could suggest to leave large holes in the image, and define different patches of the surface around this hole. Then every patch could be flattened to visualize the brain surface.

### 8.6.8 Interactive topology simplification

Further developments could allow an interactive correction of holes in the image. To give more control to the user, the algorithm could propose to the user a candidate solution for removing a hole. The user would then validate the proposition or apply a different correction. The user could opt for a different solution when a longer loop than the loop proposed follows more accurately the anatomy of the brain.

Since the small holes are generally numerous and their automated correction cannot be improved, it would be wise to only let the option to the user to find another solution for large loops. To define when the correction should be interactive, the user could define a criterion such as the maximum loop length or the maximum number of modified voxels. Above this threshold the correction would be interactive. This parameter could be added to our implementation with only a small modification.

## 8.7 Conclusion

In the present chapter, we have proven that on an experiment with a segmented brain scan our algorithm removed all the $91$ holes in the image by only modifying $1486$ voxels, i.e. a fraction of $1.8\ 10^{-4}$ of the total number of voxels in the image. The $1486$ modified voxels are distributed over the image by small clusters with generally less than $8$ voxels. The comparison of the isosurface from the input image and the output image indicates that the complex shape of the brain folds is preserved. Besides the corrections are small and do not modify large regions in the image. Hence the algorithm is useful for medical applications that require both an anatomically accurate and a topologically correct representation of the brain.

# Chapter 9

# Conclusions and Perspectives

The previous chapters described our algorithm for the simplification of topology in volumetric images, and showed that it effectively removed every hole in a segmented brain scan.

In this final chapter, we conclude this thesis and summarize the presented contributions. Then we open some perspectives for this work and suggest further research directions.

## 9.1 Conclusions

Common imaging and segmentation techniques today allow easy visualization of the anatomy of the patient with high geometric accuracy. However this representation is not sufficient, especially for Neuroscience applications where a representation of the brain without incorrect holes is important. The holes in the brain segmentation limit the accuracy of many Medical Imaging operations such as the mapping of the brain functions, the visualization of the brain surface, or the comparison of brain surfaces.

This thesis has aimed at automatically correcting the holes in segmented images while respecting the need for an anatomically accurate representation. Considering this application objective, this thesis has effectively developed and presented an algorithm that automatically corrects the holes in segmented images. The segmented images can be obtained by any manual or automated segmentation technique.

With this goal, the algorithm combines in a novel approach different concepts from Topology with a variety of discrete representations of the image. First, the representations of a wavefront and a Reeb graph are the basis to efficiently detect the topology on an isosurface of the image. Then,

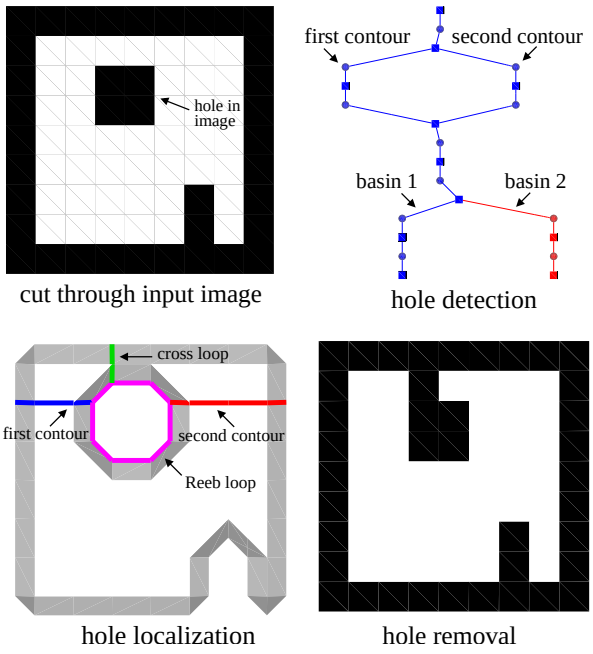


Figure 9.1: *Our algorithm removes every hole in three steps: hole detection, hole localization and hole removal. The first image shows a hole in the input image. The other images indicate the hole detection with the graph, the hole localization on the isosurface, and the hole removal in the image.*

a distance based wavefront accurately localizes the topology on this isosurface. Finally, the duality between the isosurface and the voxel representation is used to accurately simplify the topology in the image. Since a new isosurface is extracted from the corrected image, this surface is guaranteed to be free from self-intersections.

*Figure 9.1* illustrates the three steps of our algorithm. The first image shows a cut through the input image. The second image illustrates the hole detection in the graph when two contours from the same basin merge. The hole is then localized with four loops on the isosurface as indicated in the third image: two contour loops, one Reeb loop, and one cross loop. Finally the hole is removed within the image as shown in the fourth image by modifying the voxel inside the cross loop in an operation called rasterization.

The algorithm simplifies the topology of the image by removing every

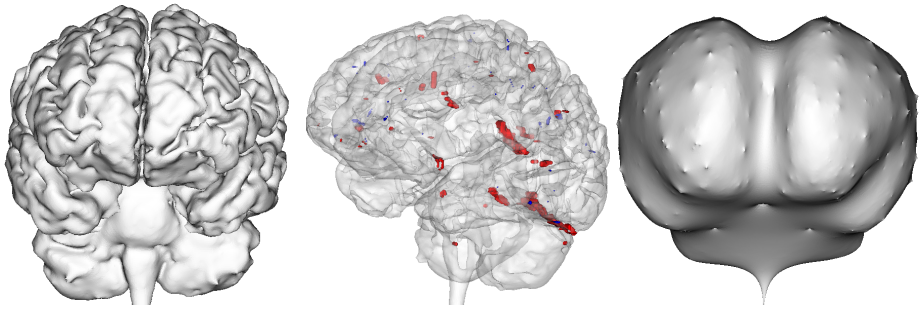

Figure 9.2: *Our algorithm removes every hole in a brain segmentation and preserves the appearance of the brain folds. The middle image shows in blue and red the few modifications to the image. The right image indicates that the isosurface of the corrected image can be inflated to visualize the entire brain area.*

hole one by one. It alternates between the progressive detection of holes and their correction until the entire image has been processed. First a hole is efficiently detected in a single exploration of the image, while other techniques often require several explorations. Then the hole is corrected based on the modification of image voxels inside a loop, with more accuracy than existing techniques. Depending on the smallest modification applied to the image, the correction can either fill or break the hole. Besides it is not constrained to lie into one plane.

### 9.1.1 Limited memory requirements

At every step of the algorithm, we limit the region of the image required in memory. First the hole detection needs one slice of the image. Second the hole localization searches in a limited region around the hole. Finally the hole removal only operates inside the localized loop. Therefore the amount of data in memory does not depend proportionally with the size of the image. Consequently the algorithm can process images that do not entirely fit in the main memory.

*Figure 9.2* illustrates our results on a brain segmentation. The left image indicates that the isosurface of the corrected image accurately represents the brain folds. The middle image shows that our algorithm only modifies a few voxels (blue and red) in the image. Since the isosurface of the corrected image does not have any hole, it can be inflated for the visualization of the entire brain surface as shown in the right image.

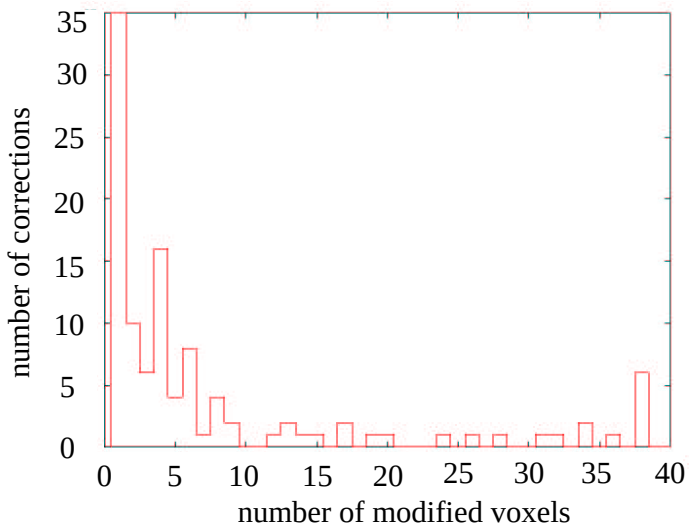


Figure 9.3: *Generally our algorithm only modifies 1 to 8 voxels to remove a hole. The histogram counts the number of corrections (y axis) corresponding to the modification of a number of modified voxels (x axis).*

Experimental results showed that our algorithm corrected every hole in a segmented brain scan with the modification of only 1486 voxels, i.e. a small fraction ($1.8\ 10^{-4}$) of the image. The practical advantages compared to existing techniques are the efficiency and accuracy that improve medical applications. The software that implements this algorithm [Jaume(2004)] is made available to help doctors with the segmentation of medical scans.

*Figure 9.3* shows the histogram of the number of corrections versus the number of voxels modified.The size of the corrections are listed along the x axis, while the number of corrections for this size is accumulated along the y axis. The first vertical bars indicate that generally to correct a hole the algorithm only modifies 1 to 8 voxels in the image.

Further experiments are needed for the validation of the method. In particular, comparison experiments with other methods would help to highlight the advantages and the limitations of our method.

## 9.2 Perspectives

Although our algorithm is targeted at the automated correction of brain segmentations, we believe that the techniques presented in this thesis could help in the analysis of other organs, and in other areas. In this section, we review some applications in segmentation, visualization, morphing, and shape analysis.

### 9.2.1 Improved segmentation

The simplification of the topology of an image could potentially help its segmentation. The segmentation of an object in an image is challenging due to noise and neighboring structures. These image artifacts often create holes inside the structure or between structures than should not be connected. Our algorithm could potentially improve the segmentation by removing the incorrect holes.

Generally the holes due to noise are small while the holes inherent to the object have a larger extent. Based on this assumption, we would localize every hole with a loop and identify a hole as incorrect if its length falls below a given threshold. Alternatively a criterion based on the number of voxels inside the loop could be used. This number would be large for a hole inherent to the object.

**Application to Medical Imaging** For many medical applications, the anatomical structure of interest has approximately a spherical topology. For instance, if the urethra and ureters could be artificially closed in the image, the bladder would have a spherical topology. Then all other holes would be image artifacts. The application of our algorithm could remove these incorrect holes.

As proposed by Jaume et al [Jaume *et al.*(2003)] the comparison of the bladder surface with topological topology helps the diagnosis of small bladder tumors. The position for the urethra and the ureters are indicated on the bladder surface as landmarks to help the comparison of different bladder surfaces.

**Interactive segmentation** The ideas from our algorithm could also be adapted to constrain an interactive segmentation software. The user would be warned if a wrong hole has just been created in the image. Besides, a possible solution could be suggested to the user to either choose

the automatic correction, or review the segmentation. This would be particularly interesting for the segmentation of brain images where the doctor segments every plane after the other and the holes are generally created across planes.

### 9.2.2 Visualization

Visualizing a complex volumetric object from its image can be confusing. Visualization on 2D cuts do not show the shape of the structure, while visualization of the boundary surface can hide most of the shape hidden within the folds of the surface. Some regions of the structure can be completely occluded even when observing the structure from different angles. Our algorithm could provide a representation of the structure that potentially improves its visualization.

**Inflation of the surface** The application of our algorithm could reduce the topology of the structure to the topology of a sphere for a more intuitive visualization. The structure could then be 'inflated' to reveal more of its surface by applying mechanical equations to every surface vertex, and letting the surface evolve until it is sufficiently unfolded. Holes on the boundary of the structure could cause major degeneracies during the inflation, and even result in an incorrect visualization. Our algorithm simplifies the topology of the image for a correct inflation of the structure.

**Mapping to a sphere** Our topology simplification could offer the visualization of the entire structure without occlusions when it is combined with a mapping to a sphere [Haker *et al.*(2000a)]. When the topology of the structure is simplified to the topology of a sphere, we can find a one-to-one mapping between every region on the structure boundary and the surface of a sphere. The folds and curved regions of the structure could be observed on the sphere with an appropriate color-coding. For instance, we could highlight the brain folds by coloring the sphere areas based on the curvature of the corresponding areas on the folded brain.

Besides, the mapping to a sphere could help to compare different objects in a common reference coordinate system. Every region of a structure mapped to a sphere can be defined by spherical coordinates. If a few landmarks are mapped at specific locations on the sphere, corresponding regions on different structures could be located with the same set of spherical coordinates. Their properties could then be easily compared.

The coherent definition of spherical coordinates on the topologically simplified surfaces could provide a coherent parameterization of organs. As proposed by Jaume et al [Jaume *et al.*(2001b)], the surface of an organ from different images could be mapped onto a sphere and compared using the spherical coordinates. Therefore the automated analysis of a database of medical scans would be improved.

**Mapping to a plane** The algorithm presented in this thesis could potentially be used to create a 2D coordinate system on the boundary of the object [Haker *et al.*(2000b)] [Zhu *et al.*(2003)]. This operation, called parameterization, defines a pair of coordinates or parameters for every position on the boundary surface of the object. Once our algorithm has simplified the topology of the object to a sphere, a unique cut on the surface is sufficient for the mapping onto the plane. The boundary of this cut can be mapped to the boundary of a circle or a square. The surface can then be 'flattened' into the circle or the square under some rules that minimize the distortion during the mapping.

The coordinate system on the 2D representation could provide a convenient way to access every position on the surface of the object. Like the mapping to the sphere, the mapping to a plane could help in the comparison of different objects. To ensure a consistent mapping, the cut through the surface must be performed at a corresponding location for the different objects.

In Graphics, the definition of a 2D coordinate system is important to map a texture onto the surface of the structure. The texture is defined as a 2D image and efficiently represents various materials (e.g. skin, brick wall, flowers) without need for very fine geometric representation. Texture mapping requires a correspondence between the image and the triangles on the surface representation. When the surface is mapped onto the plane, the image can simply be superimposed onto the plane, and the mapping can be inverted to create a textured volumetric surface.

**Faster rendering** Our algorithm could possibly improve the speed of rendering for surface rendering. Generally the object in the image is visualized by isosurface extraction and surface rendering. An isosurface algorithm such as the Marching Cubes extracts a triangle mesh from the image. The time to render the entire mesh increases with the number of triangles in the mesh.

Many triangles are required to describe the small holes on the surface.

However these holes can barely be visualized. Moreover they can even affect the visualization of the object. Therefore removing these holes with our algorithm would reduce the number of triangles, increase the speed of rendering and improve the quality of the visualization.

Besides the number of triangles can be reduced by surface simplification. However most algorithms for surface simplification such as the Progressive Meshes introduced by Hugues Hoppe [Hoppe(1996)] are limited by the number of holes on the surface. Again it is important to remove the holes in the image. Therefore the isosurface extracted from the image can be simplified to a sufficiently small number of triangles to enable interactive rendering.

### 9.2.3 Morphing

In Graphics, 3D morphing is the operation that progressively transforms a structure into another. It is important to transform the features from one structure onto the corresponding features for the second structure. For instance, we want that during surface morphing the eyes of a character would be transformed onto the eyes of a second character. The complex topology of the structures could prevent to find a one to one mapping between the two structures.

**Application in Graphics** Our algorithm could potentially reduce the topology of the two structures to a sphere for an easier morphing. The mappings from each structure onto the sphere could define the correspondences needed in the morphing. The mapping of the first structure onto the sphere would be combined with the inverted mapping of the second structure onto the sphere. The morphing animation could then be created by interpolating between the corresponding points on both structures.

**Application in Medical Imaging** Morphing is also an important operation in Medical Imaging to match an anatomical structure from one image into another. This operation is then called *registration*, and must match the anatomical features that correspond from one image to the other. Volumetric registration techniques exist, but could be driven into a wrong matching due to the presence of convolutions in the image. Similarly to the application in Graphics, the mapping onto the sphere, after the application of our algorithm, could improve the registration.

Jaume et al. [Jaume *et al.*(2001a)] proposed a registration method to map the labels from a brain atlas to the brain image of a patient. The atlas brain surface and the brain in the second image require a spherical topology surface of the brain. The method uses a multiresolution representation that efficiently describes the spectrum of the shape at different scales. The rationale is to progressively map one surface onto the other starting from the largest scale until the smallest scale. Therefore the largest features serve as landmarks to improve the matching of the smallest features.

### 9.2.4 Shape Analysis

Shape analysis is of valuable interest in numerous fields ranging from oil exploration to computer aided diagnosis. Scanning real objects is now commonly possible with sub-millimeter accuracy. The large number of scanned objects existing today and their fine resolution require automated techniques of shape analysis.

**Application in Medical Imaging** Part of the techniques developed in this thesis could potentially help in the automated recognition of some anatomical features.

When the anatomical feature has a particular topology, we could search in the image for an object with this topology. For instance, the hole within the vertebra characterizes this structure in a medical scan. It could help in the recognition of the spine, and the structures neighboring the vertebra. Another example is the pelvis bone that exhibits a large hole in its center. Automatically recognizing this hole in different images could improve their registration. We could imagine to define a coordinate system based on the topological features of the human skeleton. The movement of the patient between two images could possibly be removed by using this anatomy based coordinate system.

**Data mining** To efficiently retrieve an object in a database, we need to define the criteria that characterize this object with the highest specificity. Some measures derived from the concepts of this thesis could help to define some shape criteria.

For instance to retrieve the image of a cup, we could search of objects with a large hole on one side. Our algorithm could provide valuable shape information to define such a criterion. It automatically creates a number of loops around the holes in the object. Geometric measures about these

loops, such the length, the area, or its position relative to the center of the object, could be interesting criteria for a data mining application.

This operation is particularly interesting to automatically characterize shapes in a large number of images. The user could then define a few characteristics for the shape she or he is looking for, and the automated search would then provide the shape that best matches these criteria.

### 9.2.5 Data compression

Finally we think that the ideas of our algorithm could probably find an application for the compression of shape representations. Since the representation of shapes is becoming more and more detailed, it is important to code them in a compressed format on the computer to allow their transmission over the network, and the loading to the main memory in a reasonable time. The simplification of the topology could lead to a more efficient coding of the shape, since less information is needed for the topology. Some very efficient coding techniques, such as spectral techniques [Karni and Gotsman(2000)] [Gotsman *et al.*(2003)], require that the structure has the topology of a sphere. Our algorithm could be applied as a pre-processing step for the simplification of the topology.

Based on the techniques we have presented in this work, we could separate the coding of the topology and the coding of the geometry. We could suggest to code every topological feature one by one, and remove them from the shape before we code its geometry. When we decode the shape, we could invert the topology code, and insert every topological feature back into the shape to recover the original representation.

The perspectives discussed in this chapter indicate that concepts of Topology can benefit various applications today. We believe that more discoveries in Topology analysis will be made in the future, and will have a major impact for daily applications.

# Bibliography


[Aktouf *et al.*(1996)] Aktouf, Z., Bertrand, G., and Perroton, L. (1996). A 3D-hole closing algorithm. In *Proceedings of 6th International Workshop on Discrete Geometry for Computer Imagery*, pages 36–47, France.

[Axen(1999)] Axen, U. (1999). Computing morse functions on triangulated manifolds. In *In Proceedings of the SIAM Symposium on Discrete Algorithms (SODA)*.

[Axen and Edelsbrunner(1998)] Axen, U. and Edelsbrunner, H. (1998). Auditory morse analysis of triangulated manifolds. In *Mathematical Visualization*.

[Bischoff and Kobbelt(2003)] Bischoff, S. and Kobbelt, L. P. (2003). Snakes with topology control. *The Visual Computer*. to appear.

[Davatzikos and Prince(1995)] Davatzikos, C. and Prince, J. (1995). An active contour model for mapping the cortex. *IEEE Transactions on Medical Imaging*, **14**(1), 112–115.

[Fischl *et al.*(2001)] Fischl, B., Liu, A., and Dale, A. M. (2001). Automated manifold surgery: Constucting geometrically accurate and topologically correct models of the human cerebral cortex. *IEEE Transactions on Medical Imaging*, **20**(1).

[Gotsman *et al.*(2003)] Gotsman, C., Gu, X., and Sheffer, A. (2003). Fundamentals of spherical parameterization for 3D meshes. In *Computer Graphics (Proceedings of SIGGRAPH)*.

[Guskov and Wood(2001)] Guskov, I. and Wood, Z. (2001). Topological noise removal. In *Graphics Interface 2001*.

[Haker *et al.*(2000a)] Haker, S., Angenent, S., Tannenbaum, A., Kikinis, R., Sapiro, G., and Halle, M. (2000a). Conformal surface parameterization

for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*.

[Haker *et al.*(2000b)] Haker, S., Angenent, S., Tannenbaum, A., and Kikinis, R. (2000b). Nondistorting flattening for virtual colonoscopy. In *Medical Image Computing and Computer-Assisted Intervention*.

[Han *et al.*(2002)] Han, X., Xu, C., Braga-Neto, U., and Prince, J. (2002). Topology correction in brain cortex segmentation using a multiscale, graph-based algorithm. *IEEE Transactions on Medical Imaging*, **21**(2), 109–121.

[Hoppe(1996)] Hoppe, H. (1996). Progressive meshes. In *ACM SIGGRAPH 1996*, pages 99–108.

[Jaume(2004)] Jaume, S. (2004). http://www.opentopology.org.

[Jaume *et al.*(2001a)] Jaume, S., Ferrant, M., Warfield, S., and B., M. (2001a). Multiresolution parameterization of meshes for improved surface based registration. In *SPIE Medical Imaging*, San Diego (USA).

[Jaume *et al.*(2001b)] Jaume, S., Ferrant, M., Schreyer, A., Hoyte, L., Macq, B., Fielding, J., Kikinis, R., and Warfield, S. (2001b). Multiresolution signal processing on meshes for automatic pathological shape characterization. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Utrecht (The Netherlands).

[Jaume *et al.*(2003)] Jaume, S., Ferrant, M., Macq, B., Hoyte, L., Fielding, J., Schreyer, A., Kikinis, R., and Warfield, S. (2003). Tumor detection in the bladder wall with a measurement of abnormal thickness in CT scans. *IEEE Transactions on Biomedical Engineering*, **50**(3).

[Karni and Gotsman(2000)] Karni, Z. and Gotsman, C. (2000). Spectral compression of mesh geometry. In *Computer Graphics (Proceedings of SIGGRAPH)*, pages 279–286.

[Kikinis *et al.*(1996)] Kikinis, R., Shenton, M., Iosifescu, D., McCarley, R., Saiviroonporn, P., Hokama, H., Robatino, A., Metcalf, D., Wible, C., Portas, C., Donnino, R., and Jolesz, F. (1996). A digital brain atlas for surgical planning, model driven segmentation and teaching. *IEEE Transactions on Visualization and Computer Graphics*, **2**(3).

[Kriegeskorte and Goeble(2001)] Kriegeskorte, N. and Goeble, R. (2001). An efficient algorithm for topologically segmentation of the cortical sheet in anatomical mr volumes. *NeuroImage*.

[Lorensen and Cline(1987)] Lorensen, W. and Cline, H. (1987). Marching Cubes: a high resolution 3D surface construction algorithm. In *Proceedings of Computer Graphics SIGGRAPH*, pages 163–169, Anaheim, USA.

[Mangin *et al.*(1995)] Mangin, J.-F., Frouin, V., Bloch, I., Regis, J., and J., L.-K. (1995). From 3d magnetic resonance images to structural representation of the cortex topography using topology preserving deformations. *Journal of Mathematical Imaging Visualization*, **5**, 297–318.

[Massey(1967)] Massey, W. (1967). *Algebraic Topology: An Introduction*. Brace World, Inc.

[Munkres(2000)] Munkres, J. (2000). *Topology*. Prentice Hall.

[Reeb(1946)] Reeb, G. (1946). Sur les points singuliers d'une forme de pfaff complétement intégrable ou d'une fonction numérique. *Comptes Rendus Acad. Sciences Paris*, pages 847–849.

[Ségonne *et al.*(2003)] Ségonne, F., Grimson, E., and Fischl, B. (2003). Topological correction of subcortical segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Montreal (Canada).

[Ségonne *et al.*(2004)] Ségonne, F., Grimson, E., and Fischl, B. (2004). A map framework for the topological correction of 3-dimensional digital binary segmentation. *to be submitted*.

[Shattuck and Leahy(2001)] Shattuck, D. and Leahy, R. (2001). Graph based analysis and correction of cortical volume topology. *IEEE Transactions on Medical Imaging*, **20**(11), 1167–1177.

[Wood(2003)] Wood, Z. (2003). *Computational Topology Algorithms for Discrete 2-Manifolds*. Ph.D. thesis, California Institute of Technology.

[Wood *et al.*(2003)] Wood, Z., Hoppe, H., Desbrun, M., and Schröder, P. (2003). Isosurface topology simplification. *ACM Transactions on Graphics*.

[Zhu *et al.*(2003)] Zhu, L., Haker, S., and Tannenbaum, A. (2003). Area-preserving mappings for the visualization of medical structures. In *Medical Image Computing and Computer-Assisted Intervention*.