RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Lehrstuhl für Bildverarbeitung
Institute of Imaging & Computer Vision

RWTH Aachen University
Institute of Imaging & Computer Vision
Prof. Dr.-Ing. T. Aach

Bachelor Thesis

Interpolation of Holes in 3D Lung Surfaces

Author:	Botros, Rami
Email Address:	ramibotros@rwth-aachen.de
Supervisors:	Dipl.-Ing. Kraisorn Chaisaowong
	Dipl.-Ing. Peter Faltin

Hiermit versichere ich, dass ich die vorgelegte Studienarbeit selbständig angefertigt habe. Es sind keine anderen als angegebenen Quellen und Hilfsmittel benutzt worden. Zitate wurden kenntlich gemacht.

Aachen, den 28. September 2011

(Rami Botros)

Table of Contents

Chapter 1 Introduction

1.1. Medical Introduction

Asbestos denotes a set of minerals that were exploited for their practical chemical properties. They are actinolite, amosite, anthophyllite, crocidolite, tremolite and chrysotile. The materials have high tensile strength and they do not evaporate or burn (Alleman & Mossman, 1997). Such relative chemical inertness has broadened its industrial appeal, but has also made human exposure to it dangerously unhealthy.

The main applications of asbestos materials have taken place in the building sector and the workers in these industries have had the heaviest exposure to them (American Cancer Society - Asbestos, 2010). During the 20th century, there has been a dramatic growth in the production as well as the consumption of asbestos, which peaked around the year 1978 (See Figure 1).

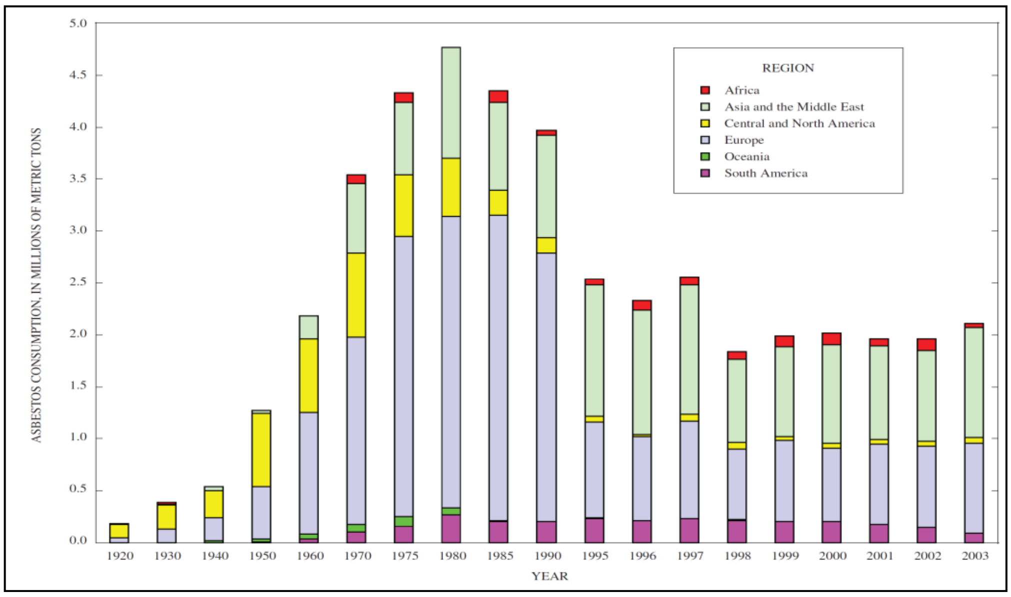


Figure 1 - Bar graph of apparent consumption of asbestos by region, 1920–2003. Source: (Virta, 2006)

Health research during the 1950s to the 1960s established a conclusive association between asbestos and lung cancer and depreciation toward its use started in the late 1970s. By 2003 the use of the material has been banned in 16 countries (Virta, 2006), including the European

Union. Because active utilization still takes place in Africa and Asia, production is still going on in many countries, despite the use restrictions. Currently, Germany is applying an assessment program to those who have been exposed to asbestos. A specific form of cancer, mesothelioma, has been statically proven to be mainly caused by asbestos exposure (Chaisaowong, Knepper, Kraus, & Aach, 2007). It is a tumor that often occurs in the pleura as the one shown in Figure 2.

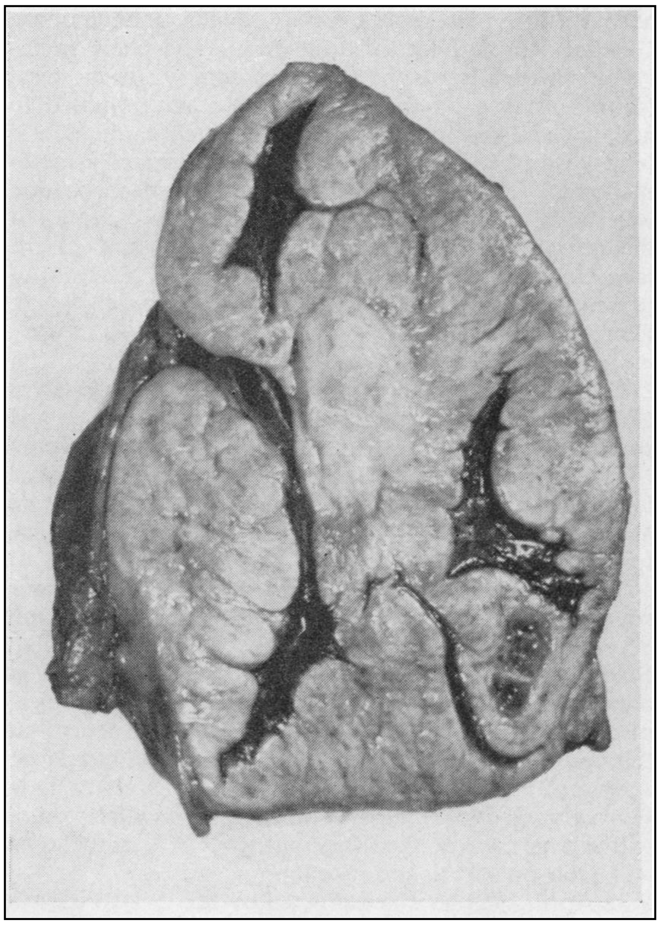


Figure 2 - Right lung affected by Mesothelioma
Source: (Wagner, Sleggs, & Marchand, 1960)

1.2. Main Topic of Research

For physicians, the assessment of lung thickenings is a tedious and time-consuming process, often with subjective outcomes. The Pleuramesothelioma project's aim is to develop a system that automates the detection of lung thickenings, as well as characterizes their medical properties. Typical properties of interest are the thickening volume or thickness. A method for thickness estimation is the interpolation of a healthy lung based on a spline model (Chaisaowong, Knepper, Kraus, & Aach, 2007).
An alternative approach for the interpolation is presented here. It consists of creating holes on the surface of the lung's mesh, where thickenings are detected, and filling them geometrically. This should give an estimation of a healthy lung surface. When comparing that healthy lung

model to the original lung data, the volumes of the thickenings can be determined. We investigate different hole filling algorithms and evaluate their suitability for lung models. A satisfactory hole filling method should be automatic, should fill arbitrary holes in lung surfaces and return a mesh that is minimally distinguishable from that of a healthy lung.

Chapter 2 Background

2.1. Technical Introduction

This work deals with two kinds of data representations for 3D structures. The first is the volumetric representation, which is made of voxel elements. The second representation is the mesh representation (geometric representation), whose building blocks are vertices and triangles.
For the purposes of this work, we define a voxel to be a unit cubic volume centered at a grid point; i.e. it is the 3D counterpart of the pixel. A volumetric data set is the aggregation of equality-sized voxels, which tessellate all of the available volume space (Kaufman, Cohen, & Yagel, 1993). Collectively, voxels specify a functional of the model across all of 3D space. Each voxel is associated with numerical values that describe model properties at its location, for example the color. However, a voxel's data entry does not indicate its position; this is rather inferred from its relative position to other voxels within the structure of the volumetric data set. The structure is illustrated in Figure 3.

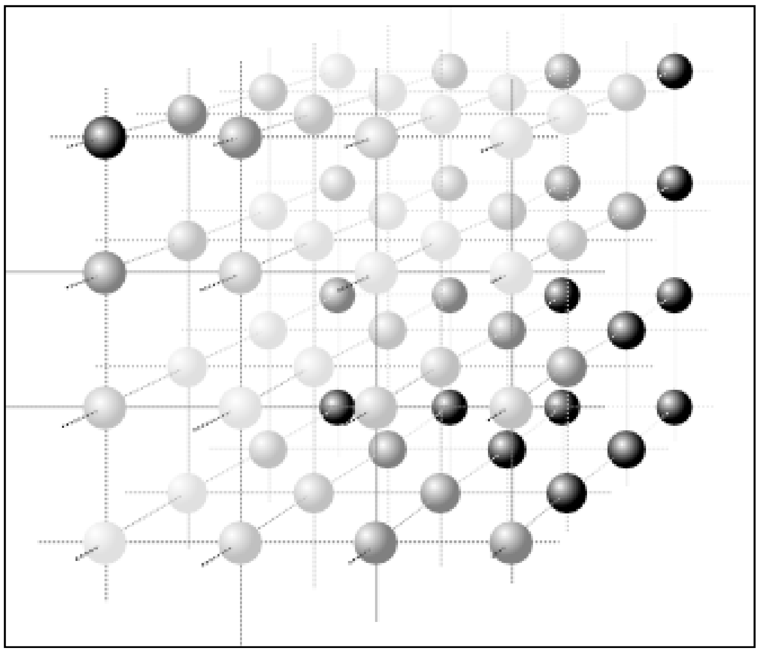


Figure 3 – A collection of voxel tessellating all of free space. Some functional is indicated by the grayscale intensity of the spheres

In our case the volumetric grid is rectilinear and each voxel contains a density value to indicate the extent to which the voxel is occupied by the model, i.e. the fraction of the voxel's volume that includes parts of the model's volume. A special case of the volumetric representation is the binary voxel representation, where the voxel data is in binary format. In this case, the value zero indicates empty space and the value one indicates space filled by the model.

The second type of data format for 3D models, the triangular mesh representation, is defined by a set of vertices and a set of triangles. A vertex is a representation of a point in space defined through three coordinates. A triangle is defined through the three vertices which it connects via straight edges. Two adjacent triangles share a common edge. An edge that is not shared between two triangles, and is instead owned by one triangle, is a boundary edge; vertices on it are called boundary vertices. Boundary triangles are those who own one or more boundary vertices.

One-ring triangles of a vertex are all those triangles which own it as one of their respective three corners. Similarly, one-ring edges of a vertex are all those edges which connect with it. One-ring vertices of a given vertex, a.k.a its neighbors, are all vertices on its one-ring edges except itself.

Moreover, a triangle normal is the normal to the plane on which the triangle resides, and a vertex normal is the average of the triangle normals of all its one-ring triangles. These concepts are illustrated in Figure 4.

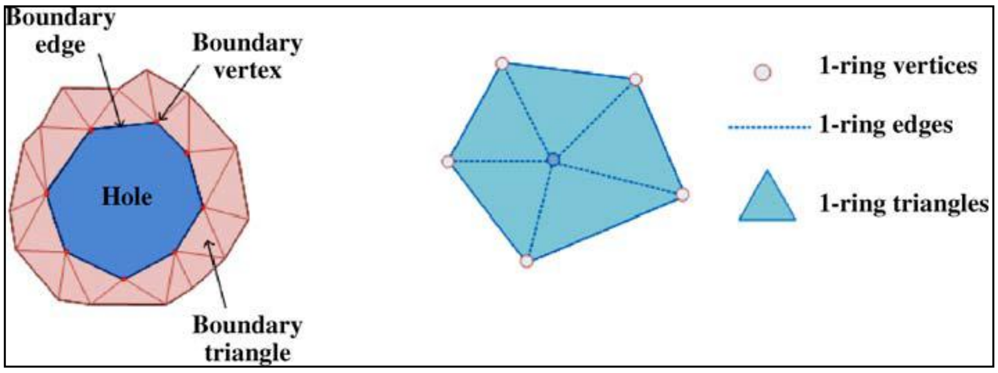


Figure 4 - Basic nomenclature related to mesh models.
Source: (Zhao, Gao, & Lin, 2007)

Our definition of a hole follows that of (Zhao, Gao, & Lin, 2007): It is a closed contour that is formed by boundary edges. We generally assume that meshes are oriented, manifold and connected. Consequently, the holes will have no islands in them. Also, two separate holes will have not vertices in common. Note that such topological definition of a hole is not always compatible with the conventional conception of a hole (Liepa, 2003).

Given that the original data is available in voxel format, we need a method that converts a volumetric representation of a model to an equivalent geometric one. For this purpose, we utilize isosurfaces. An isosurface is a level set of a function, whose domain is 3D space. It is therefore a surface consisting of all points where the function is equal to a certain constant value. Theoretically, the outer surface of a binary volume can be obtained as the isosurface of a median value between the two binary values in the data set (Lempitsky, 2010). If the binary values consist of ones and zeros, this median value can be 0.5.

2.2. Literature Review

Most works on hole-filling fall into two data-format-based categories: There are voxel-based approaches and mesh-based approaches.

One volumetric approach (Curless & Levoy, 1996) interpolates holes using space carving and iso-surface extraction. The hole-filling is integrated within a surface-reconstruction process using distance functions. The method always produces a water-tight surface, but the results may be less plausible than other smooth methods. (Davis, Marschner, Garr, & Levoy, 2002)

Another volumetric approach is that of (Davis, Marschner, Garr, & Levoy, 2002) , which works via isotropic diffusion of volumetric data, which is performed until the fronts meet. The data is represented as a voxel data set which contains values of some signed distance function. The diffusion process consists of alternate blurring (iterative Gaussian convolution of the distance function) and compositing, which take place only in the vicinity of holes. A downside of this method is that it might cause changes in the original mesh.

Volumetric hole interpolation addresses generally holes that have complex topological configurations (for instance, holes with islands), which are more significantly more difficult to handle with mesh-based approaches. They are, nevertheless, time-consuming and may result in incorrect topology in case of large holes (Brunton, Wuhrer, Shu, Bose, & Demaine, 2009). Because the lung thickenings may span large surfaces, we expect large holes to occur frequently and require adequate method for such cases.

In addition, numerous methods have been developed that work directly on the geometric representation of meshes. (Carr, Fright, & Beatson, 1997) use radial basis functions to compute an implicit surface covering the hole. This representation stems from scattered data and ensures smooth reconstruction of hole regions. While this method works well for convex surfaces and can handle irregular holes, difficulties arise when the surface is too complex to be described by a single-valued function. Moreover, because an RPF is computed for the full surface, the complexity of the algorithm depends on the size of the original surface, even if the holes are small.

It is common for geometric approaches to perform a triangulation in order to cover the holes. For instance, (Dey & Goswami, 2003) use a Delaunay-triangulation-based method without adding new vertices. This method can only fill small holes properly, because the extrapolation can only be performed in the vicinity of the boundary. A similar outcome is given by (Barequet, Dickerson, & Eppstein, 1998), where a divide-and-conquer algorithm is used to obtain a minimum-area triangulation of the hole boundary (also without adding new vertices). Moreover, in the works of (Zhao, Gao, & Lin, 2007) and (Liepa, 2003) complete accounts on hole-filling procedures are given. These start with planar patching of the hole, where new vertices are added to the patching mesh until the vertex density of the surrounding mesh is matched. This step is followed by a fairing step, where vertices are repositioned in order to make the patch match the shape of the surrounding mesh. We investigate these two works in detail.

(Liepa, 2003)'s triangulation algorithm is influenced from (Barequet, Dickerson, & Eppstein, 1998) and extends it to a min-max triangulation that takes into account the dihedral angles between adjacent triangles. The aim of such extension is to handle hole boundaries with geometric features that are orthogonal to the triangulation. A comparison of triangulation results for such cases is shown in Figure 5.

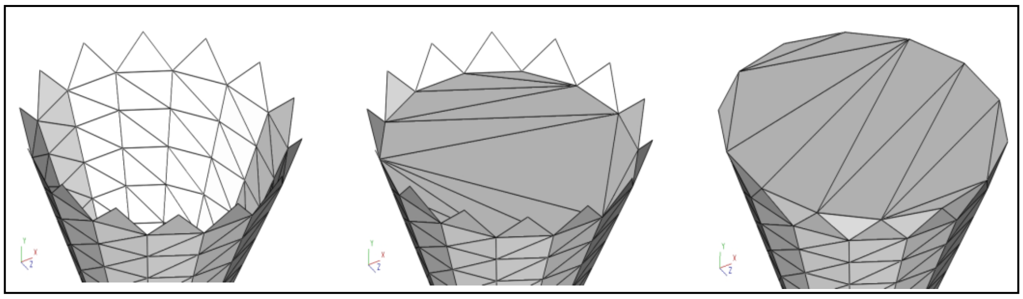


Figure 5 - Hole boundaries with orthogonal features (left), minimum-area triangulation results (middle), min-max triangulation results (right)
Source: (Liepa, 2003)

The next step is a refinement of patching mesh. In order to achieve a triangle density close to that of the surrounding surface, edge-length data of boundary vertices are obtained and diffused towards the interior. The amount of vertices and triangles to be added should depend on this data. The last step is a fairing process that should smoothen out the whole resulting mesh, i.e. the composition of the patching mesh with the original mesh. Through solving a linear system of equation (that is constructed using the so-called Umbrella Operator), the vertices of the patching surface are relocated to collectively minimize a fairing function.
(Zhao, Gao, & Lin, 2007) fill holes using a similar procedure. A fast Advancing Front Mesh method is used to reach a triangulation with an adequate triangle density. This is done by repeatedly advancing the hole boundary via the addition of new triangles in its vicinity. The shape and alignment of the new triangles is determined by the angle between the existing boundary edges. This is also followed by a fairing step, but this time it works on achieving smooth changes between vertex normals across the mesh.
Our application requires the filling of large whole in a small amount of time, therefore we have chosen to take the geometric approach. Our work is based mainly of that of (Liepa, 2003) and (Zhao, Gao, & Lin, 2007).
The preliminary conversion from volumetric to geometric data is often carried out by the standard Marching Cube algorithm of (Lorensen & Cline, 1987) or variation thereof. The algorithm takes a threshold value from the user and divides the model into small cubes, whose corners are made out of voxel centers and thus have data values associated with them. Each of these corners either has a value larger or smaller than the given threshold, making it considered “inside” or “outside” the surface to be extracted, respectively. There are 256 possible permutations for a given voxel as to which of its corners are inside or outside of the surface. For each of those combinations there is a preset polygon configuration that should be placed in the appropriate position along the cube’s edges. Although the method is efficient, it produces surfaces that “exhibit distracting aliasing artifacts” and are not smooth enough (Lempitsky, 2010).
(Lempitsky, 2010) solves this problem by reinterpreting the role of the volume data. As far as this approach is concerned, the given volume data does not define the separating surface directly. Instead, its new role is to impose hard constraints on the separating surface. Instead of performing the Marching Cubes algorithm directly on the given volume data, a new non-binary volumetric function is obtained, on which the Marching Cubes algorithm is carried out. For smoothness, the new volumetric function should minimize a so-called Energy Integral, while satisfying boundary conditions imposed by the original volume data. The surfaces produced by this method are considerably smoother yet close to the original data.

2.3. Description of Environment

The main numerical computing environment used is MathWorks MATLAB. The original voxel data was provided in MATLAB format. Self-implemented algorithms like hole creation, detection, triangulation and fairing were written in MATLAB. For the mesh fairing process, the MATLAB implementation of sparse matrices was used. For the application of the surface extraction method of (Lempitsky, 2010), the original implementation provided by the author is used. It is also written in MATLAB and utilizes the isosurface function of the software. (To our best knowledge this function is based on the Marching Cubes algorithm). The plots produced are all created in MATLAB.
For error rate investigation, a third-party software called Metro, provided by the authors of (Cignoni, Rocchini, & R., 1998), is used to compare between two surfaces. For comparison with original data, external MATLAB implementations are used to convert geometric data back to volumetric representations.

Chapter 3 Methods

Figure 6 shows a schematic of the whole procedure. First, the CT data goes through segmentation and thickenings are detected and segmented. This stage produces a volumetric model of the lung. From there, the mesh extraction should produce a geometric representation that resembles the present model. Afterwards, holes are deliberately created in the mesh in places where thickenings have been detected. The subsequent hole detection stage identifies the vertices that form the holes' boundaries.

At that point the hole filling procedure can begin. The first step is the planar patching of the hole, i.e. laying out a relatively planar surface that covers it. This surface should have a triangle density similar to that of the surrounding mesh. During the research, two parallel methods were used and compared. The method of (Liepa, 2003) is a two-step method: It starts with a hole triangulation step, which does not add new vertices to the mesh. Therefore, in order to match the triangle density of the surrounding mesh, a subsequent refinement step has to be performed. The Advancing Front Mesh method, on the other hand, achieves the adequate triangle density in a one-step process.

In both cases the final fairing method is the same. It should result in smooth composition of the patching and the surrounding mesh. In case needed, this result can be converted back into voxel format. However, such conversion to voxel representation is not guaranteed to be free of errors. Such errors should be taken into account.

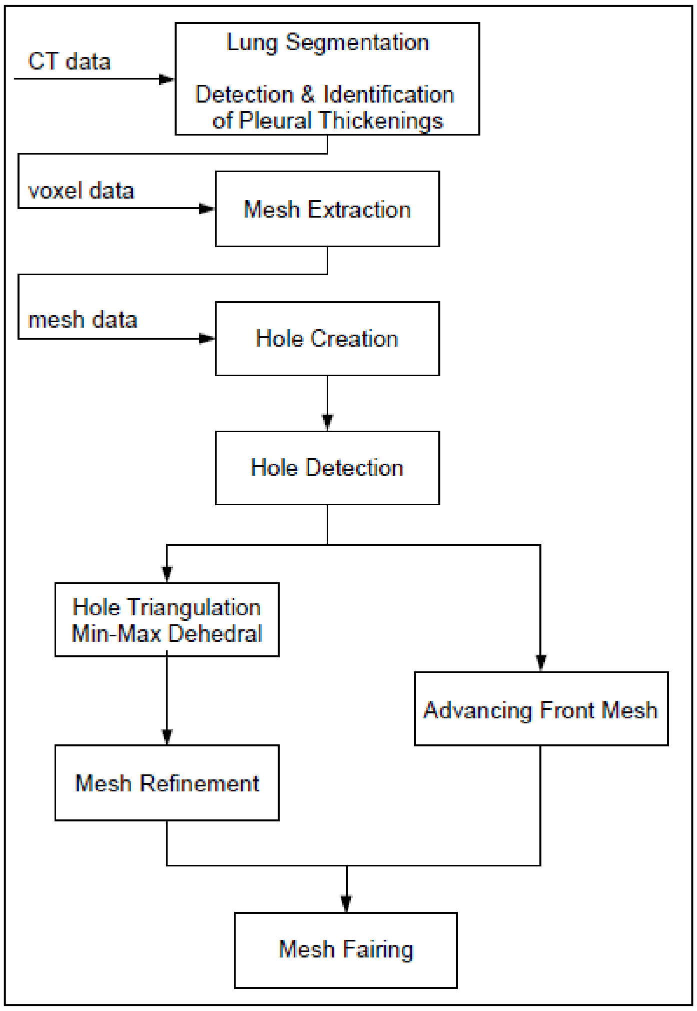


Figure 6 – Overview of the methods used

3.1. Voxel Data from CT Data

The initial data is provided as complete, unsegmented CT image data. In the previous work of the project, (Chaisaowong, Bross, Knepper, Kraus, & Aach, 2008), a lung segmentation procedure is described to extract the essential volumetric data of the lung from the CT data. The first step of this procedure is a two-step segmentation using supervised range-constrained Otsu thresholding. Thresholding is a simplistic segmentation process, where pixels are segmented into foreground and background classes, depending on how their associated data fields (e.g. intensity value or color at a given pixel) compare with a set threshold. The quality of the thresholding process depends on the proper choice of the threshold value: Increasing it enlarges the false-negative error rate and decreasing it enlarges the false-positive error rate (Chaisaowong, Knepper, Kraus, & Aach, 2007). Generally, Otsu thresholding methods use statistical methods to predetermine an appropriate threshold, or a range thereof. In our case, a priori knowledge about a histogram of the background itself is used for the purpose of setting upper and lower limits to the threshold range. This thresholding method, along with 3D connected component labeling, is used to extract the pulmonary organs out of the initial, wide-ranging CT Data in two steps: Firstly, the thorax is identified (Result: Figure 7, top), and then the lungs are separated from the thorax. Afterwards, the trachea and bronchi are selectively eliminated by modeling the gray-levels of the organs as a mixture of Gaussian distribution and using the Expectation-Maximization algorithm. Then the left and right lungs are separately identified and can be represented individually. (Result: Figure 7 – bottom).

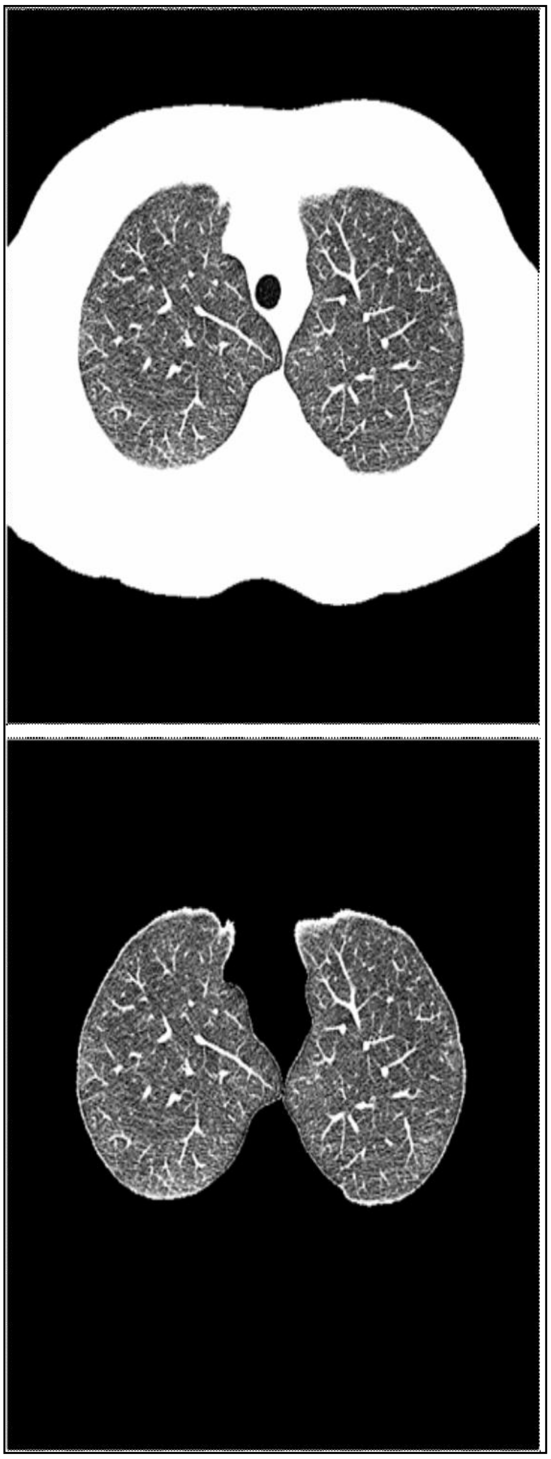


Figure 7 - Result of lung segmentation. Above: Result of thorax extraction. Below: Result lung extraction and right and left lung separation
Source: (Chaisaowong, Knepper, Kraus, & Aach, 2007)

3.2. Extraction of Mesh from Voxel Data

The surface extraction method described here is identical to the one found in (Lempitsky, 2010). We redefine the binary volumetric representation to be $V(x, y, z) \rightarrow \{-1,1\}$. The goal is to obtain a function $F(x, y, z) \rightarrow \mathbb{R}$. This output should have the constraint generated by

$$\forall\ (x, y, z) F(x, y, z) \cdot V(x, y, z) \geq \min_{(a, b, c) \epsilon B} \sqrt{(x-a)^2 + (y-b)^2 + (z-c)^2},$$

where B is the set of voxels that are in 26-connectivity to nodes of the different values in V. Note that the right-hand side of the inequality denotes a minimum distance. These constraints are imposed while minimizing the functional

$$\int_\Omega \left(\frac{\partial^2 F}{\partial x^2}\right)^2 + \left(\frac{\partial^2 F}{\partial y^2}\right)^2 + \left(\frac{\partial^2 F}{\partial z^2}\right)^2 d(x, y, z),$$

which has the discrete approximation

$$\sum_{x,y,z} \left[\big(F(x+1, y, z) + F(x-1, y, z) - 2F(x, y, z)\big)^2 + \big(F(x, y+1, z) + F(x, y-1, z) - 2F(x, y, z)\big)^2 + \big(F(x, y, z+1) + F(x, y, z-1) - 2F(x, y, z)\big)^2\right].$$

The minimization is carried out by MATLAB's second-order optimizer.
In the end, the Marching Cubes algorithm is used to extract the zero isosurface of F. Its steps are as follows:

1. Build a cube out of eight voxel center points as its corners.
2. Study each corner to see whether its voxel is larger or smaller than a given threshold.
3. The combination of how each voxel is evaluated in Step 2 can be one of the known 256 combination. Look-up the current combination in the preset table and draw the triangles that correspond to it between the location of the voxels.
   Note that because of symmetry and rotation, these combination can be completely described through 15 cases only, shown in Figure 8.
4. Interpolate (through linear Interpolation) the exact location of the corners of the triangles generated in Step 3.

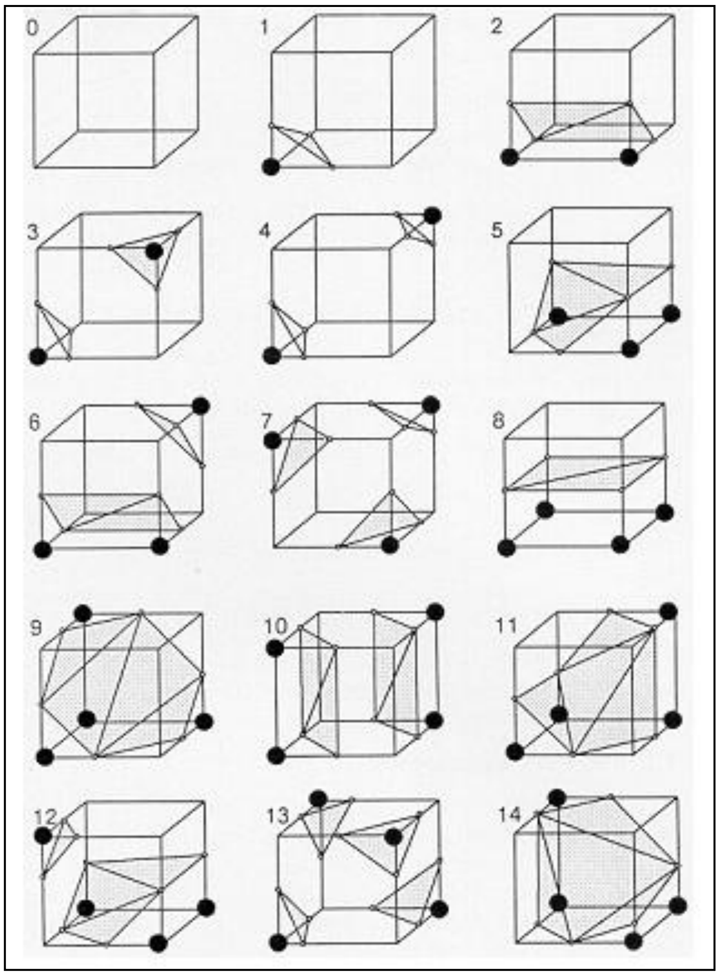


Figure 8 - Fifteen unique combinations based on the corners that surpass the threshold value.
Source: (Lorensen & Cline, 1987)

3.3. Hole Identification and Filling

A preliminary step is the creation of holes in the lung mesh. This creation is expected to be aided by the automatic detection of pleural thickenings of (Chaisaowong, Bross, Knepper, Kraus, & Aach, 2008).All vertices of the areas identified as part of thickenings should be removed from the mesh, along with the triangles that own them. For testing purposes, holes were created in arbitrary healthy locations and the result of the hole filling was compared to the original lung.
We describe the process of hole identification and filling for a single given hole, which can easily be extendedly applied to multiple holes in a given mesh.

3.3.1 Hole Identification

The purpose of the hole identification step is to obtain the contour of the hole boundary. The problem of hole detection can be formulated as the detection of a series of connected boundary edges (Liepa, 2003). A vertex is identified as a boundary vertex if the number of its one-ring triangles and one-ring edges are not equal (Zhao, Gao, & Lin, 2007). Given that the

hole boundary is continuously connected, the problem is simplified once a seed boundary vertex is found: The algorithm can trace other connected boundary vertices until closed loop of them is formed. In the application at hand, the seed is assumed to be given, since the hole creation is a deliberate process, whose location is known.
The full algorithm is carried out as follows:

1. Define $V_{boundary}$ as a set identified boundary vertices. Initialize it with a given seed boundary vertex.
2. Define v_{last} as the last element in $v_{boundary}$. Define the set one-ring vertices of v_{last} as V_{ring}. If $|V_{boundary}| > 2$ and $|V_{boundary} \cap V_{ring}| > 0$, the hole identification is complete.
3. Define $V_{candidates} = V_{boundary} \backslash V_{ring}$. Look in $V_{candidates}$ for boundary vertices (i.e. vertices whose number of one-ring triangles and one-ring edges are not equal). If more than one is found, choose the one which is connected to v_{last} through a boundary edge. This case is illustrated in Figure 9, where v_1 should be chosen. Define the chosen vertex as v_{new}.
4. Append v_{new} to the end of $V_{boundary}$ and go to Step 2.

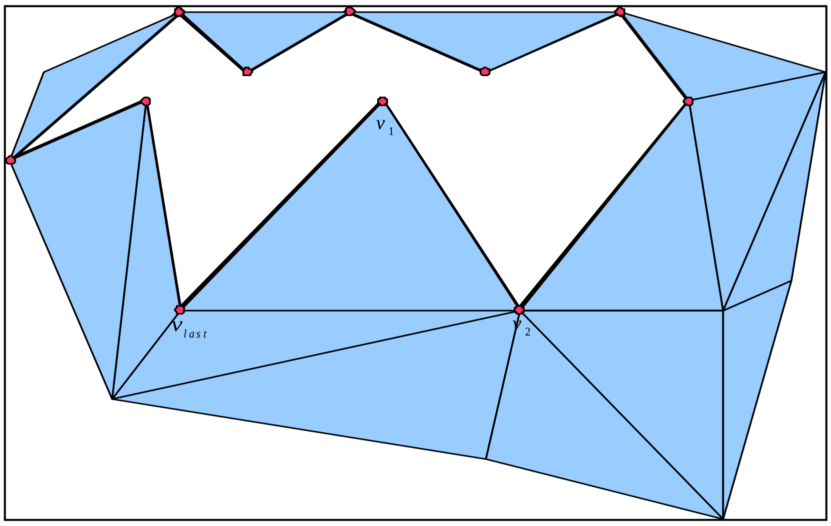


Figure 9 - A case where there are two candidate boundary vertices.

3.3.2 Planar Patching of the Hole

1) *Min-Max Dihedral Hole Triangulation and Refinement*

This algorithm from (Liepa, 2003) is based on a minimum-area triangulation method found in (Barequet, Dickerson, & Eppstein, 1998), which is in turn based on a planar triangulation procedure found in (Deza & Rosenberg).
A triangulation of a hole is a collection of triangles that connect the hole boundary vertices such that:

- Each boundary vertex is owned by at least one triangle.
- Every boundary edge is owned by exactly one triangle (Barequet, Dickerson, & Eppstein, 1998).

- A triangle's edge that has not originally been a boundary edge of the hole is owned by exactly two triangles (Barequet, Dickerson, & Eppstein, 1998).

If a weight can be defined for each triangle of a hole triangulation, a minimum triangulation problem is formulated as follows:
Find a triangulation of the hole that minimizes the total sum of the triangles' weights. This problem can be shown to be NP-complete (Liepa, 2003).
A simple choice for weight is the triangle area. In our case the weight is defined to be a pair of values. The first represents the triangle area and the second is the maximum dihedral angle between it and its existing adjacent triangles.
The problem is solved through a brute-force method, which is performed in dynamic programming manner, i.e. by breaking the problem into simpler subproblems. The sum of the triangle weights is recursively expressed in terms of a triangulation that involves a subset of the boundary vertices. The base weight is that of three boundary vertices; it can be easily evaluated because it only involves a fixed triangle area and maximum dihedral angle.
The necessary subsequent refinement step results in a breaking down of the triangulation mesh into smaller triangles, such that the triangle density of the patching mesh matches that of the surrounding mesh.
Each boundary vertex is associated with a scale attributed that is the average length of its one-ring edges. The refinement algorithm consists of a repeated process of alternation between adding new vertices and relaxing new triangles:

- New vertices are added as centroids to existing triangles if the scale attribute of all of its vertices are small enough.
- Relaxing triangles is an edge swap that might happen for two adjacent triangles. The condition for the edge swap is that a non-shared vertex lies inside the circum-sphere of the triangle that does not own it. This condition is illustrated in Figure 10.

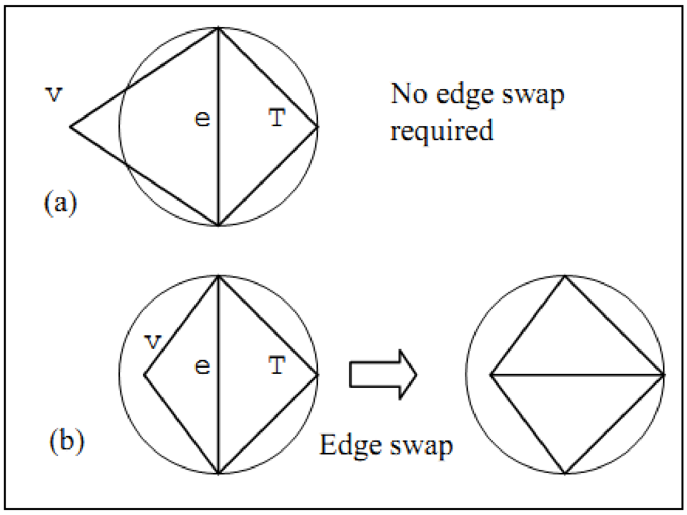


Figure 10 - Edge swap condition.
Source: (Liepa, 2003)

The exact algorithm is described in (Liepa, 2003). Note that the triangulation algorithm is an $O(n^3)$ algorithm.

2) Advancing Front Mesh

An alternative method for planar patching has been investigated. To so-called Advancing Front Mesh (AFM) method is a one-step procedure, which starts by adding new vertices to the mesh right away. New triangles are iteratively created next to the hole boundary. As a result, the hole boundary is moved towards the interior of the hole. The decision-making for the shape of the new triangles relies on the angles between adjacent boundary edges. Starting from the edge pair with the smallest angle, new vertices and/or edges are added based on that angle, as shown in Figure 11. The angle range from left to right is: $\theta \leq 75°$, $75° \leq \theta \leq 135°$ and $135° \leq \theta$.

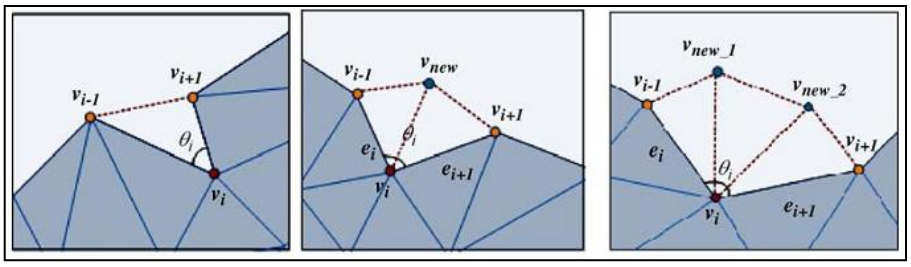


Figure 11 - The addition of new edges and/or vertices based on the angle. Source: (Zhao, Gao, & Lin, 2007)

The exact rules for the creation of new mesh elements are:

- If $\theta \leq 75°$, an edge is created between v_i and v_{i+1}.
- If $75° \leq \theta \leq 135°$ a new vertex v_{new} is created such that
  - $\measuredangle(v_{i-1}, v_i, v_{new}) = \measuredangle(v_{i+1}, v_i, v_{new})$
  - $|v_{new}v_{i-1}| = |v_{new}v_{i+1}|$
- If $135° \leq \theta$ two new vertices v_{new1} and v_{new2} are created such that
  - $\measuredangle(v_{i-1}, v_i, v_{new1}) = \measuredangle(v_{new1}, v_i, v_{new2}) = \measuredangle(v_{new2}, v_i, v_{i+1}) = \frac{\theta}{3}$
  - $|v_i v_{i-1}| \cdot \sqrt{2} = |v_i v_{new1}|$
  - $|v_i v_{i+1}| \cdot \sqrt{2} = |v_i v_{new2}|$

Note that the $\sqrt{2}$ factor used in the last rule has been empirically determined to produce visually satisfying results.
The distance between a newly added vertex and each boundary vertex is evaluated. If it is less than a preset threshold, the vertices are connected via a new edge. The method is carried on and with each iteration the hole size is decreased until the hole is completely is patched.

3.3.3 Mesh Fairing

The last step smoothens the patching mesh with respect to its surrounding mesh. The algorithm described here is based on the minimization procedure described in (Kobbelt, Campanga, Vorsatz, & Seidel, 1998). It minimizes the thin-plate energy function through the following definition of the discrete Laplacian, a.k.a. the Umbrella Operator at a given vertex:

$$U(v) = \frac{\sum_i \omega(v,v_i) \cdot (v_i - v)}{\sum_i \omega(v,v_i)},$$

where v_i are the one-ring vertices of v and $\omega(v_1, v_2)$ is a weight function associated with the edge that connects v_1 and v_2. Notice that the expression is a weighted averaging of the differences between v and each of its neighbors. The weight of each neighbor for the averaging is determined by $\omega(v_1, v_2)$. Generally, the Umbrella operator returns a vector that describes the average deviation from tautness of v from its one-ring vertices (Liepa, 2003). An example is shown in Figure 12.

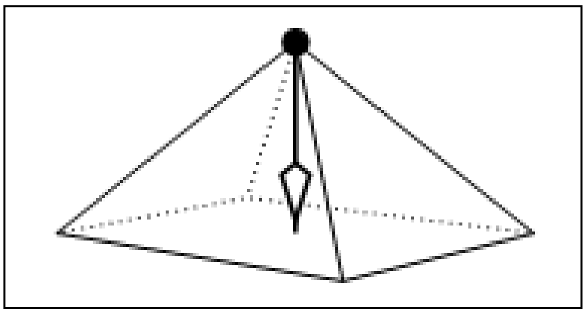


Figure 12 - Illustration of the Umbrella Operator of a vertex
Source: (Desbrun, Meyer, Schröder, & Barr, 1999)

The second-order umbrella operator is defined by reapplying the Umbrella Operator onto itself yielding:

$$U^2(v) = \frac{\sum_i \omega(v,v_i) \cdot (U(v_i) - U(v))}{\sum_i \omega(v,v_i)}.$$

If this operator is zero, that would mean that “the deviation from tautness at the vertex v is equal to the average deviation from tautness of its neighbors” (Liepa, 2003). This is equivalent to the mesh being smooth at that vertex. Thus, a linear system of equations is constructed, where for each vertex in the patching mesh the second-order umbrella operator is set to zero.
Fortunately, the resulting LSE is sparse, making it efficiently solvable without requiring large memory space or long computation time.
The choice of the weight function ω can be as simple as

$$\omega = 1.$$

The scale-dependant umbrella operator uses

$$\omega(v_1, v_2) = |v_1 - v_2|$$

and deals with irregular hole shapes better, causing less distortion (Desbrun, Meyer, Schröder, & Barr, 1999). In order to maintain the linearity of the equations, we consider the distances between vertices to remain approximately constant during the fairing process and do not include them as variables in the LSE. Another omega function that is investigated is

$$\omega(v_1, v_2) = \cot(\measuredangle(v_1, v_{k1}, v_2)) + \cot(\measuredangle(v_1, v_{k2}, v_2)).$$

The definition of these vertices is shown in Figure 13. This function achieves good triangle shapes as well as smooth patches in the case of surrounding mesh with high curvature.

Moreover, a self-implemented weight function is examined:

$$\omega(v_1, v_2) = \begin{cases} \dfrac{\cot\left(\measuredangle(v_1, v_{k1}, v_2)\right) + \cot\left(\measuredangle(v_1, v_{k2}, v_2)\right)}{\min\limits_{v_b \in V_B}\left(|v_2 - v_b|\right) \Big/ \min\limits_{v_b \in V_B}\left(|v_1 - v_b|\right)} & \text{if } v_2 \notin V_B \\ \\ \dfrac{\cot\left(\measuredangle(v_1, v_{k1}, v_2)\right) + \cot\left(\measuredangle(v_1, v_{k2}, v_2)\right)}{\alpha} & \text{if } v_2 \in V_B \end{cases},$$

where V_B denotes the set of boundary vertices of the hole and α is a control parameter that should be specified manually. This function is a scaled version of the previous one. The scaling depends on the minimum distances of v_1 and v_2 from the hole boundary. It scales the weight up to reward the case when v_1 is closer to the boundary than v_2. In the case where v_2 is itself a boundary vertex, the division-by-zero error is avoided and the control parameter α is used as a divisor. When used in the weighted averaging of the Umbrella Operators, this function causes the neighbors that are closer to the boundary to have a stronger effect on the vertex on which the operator is performed. Consequently, the gradients at the boundary have a stronger effect during the fairing and diffuse inwards inside the patch. This effect can be made stronger or weaker by decreasing or increasing α, respectively.

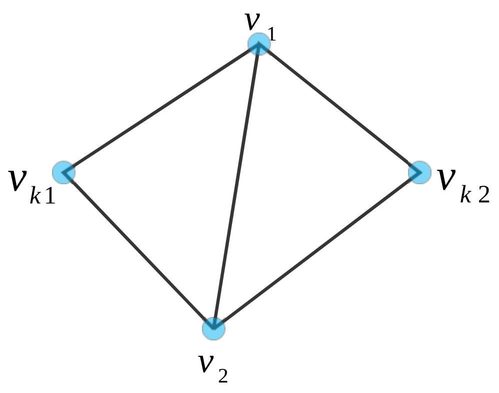


Figure 13 – Definition of vertices to define an omega function

Chapter 4 Results

4.1. Processing Times

Different steps in the used procedure have different costs regarding required CPU time. A relative comparison between the processing times for different steps for the study case shown in Table 1 identifies the typically expensive steps of the procedure. The three most expensive steps are: Mesh Extraction (Section 3.2), Method 1 of Planar Patching (Section 3.3.2) and Mesh Fairing (Section 3.3.3).

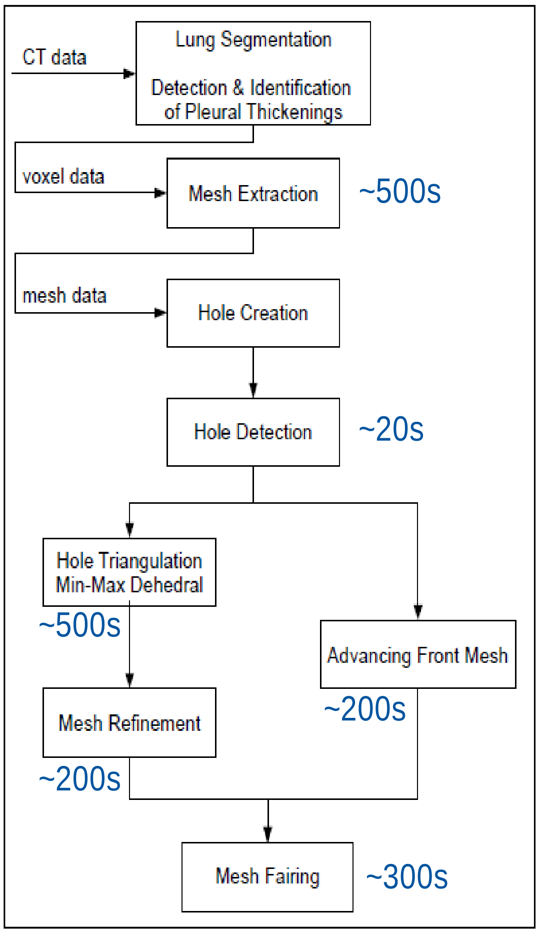


Table 1 – Comparison of process times

4.2. Surface Extraction

Two mesh extraction techniques are examined: The first is the direct application of the Marching Cubes algorithm, while the second is the redefinition of the voxel data followed by the Marching Cubes algorithm. An example for visual results for the two techniques is shown in Figure 14. On the left, the results of direct application of Marching Cubes is shown, while for the case illustrated on the right, the preliminary redefinition step has been applied beforehand. Through the camera lighting effect, noisy artifacts are made visual.

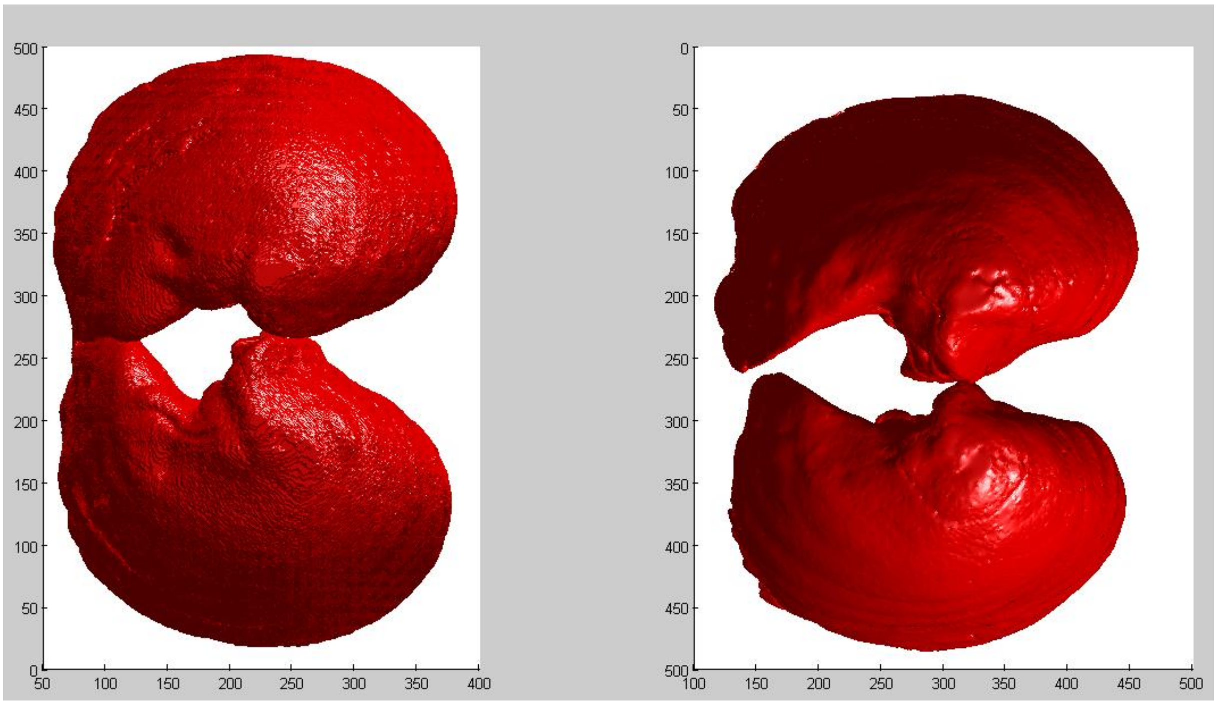


Figure 14 – Extracted meshes.
Left: Direct Marching Cubes; Right: Voxel data redefinition before Marching Cubes

A good mesh extraction algorithm should cause the extracted mesh to closely imitate the outer-most surface of the volume data. In order to evaluate this measure, the extracted mesh is converted back to voxel format and the result is compared to the original voxel data.
For evaluation purposes, a conversion back to voxel format needed to be performed. In brief, the conversion is done by splitting each triangle, until its longest edge is smaller than the size of half a voxel. This is followed by setting the voxel located around each of the small triangle’s corner to one. Finally the resulting voxel data is filled from inside using Matlab’s “imfill” function, such that all voxel inside the surface are set to one. Note that this conversation algorithm does not exactly negate the effect of mesh extraction. Therefore the obtained voxel data is expected to differ from the original voxel data in a systematic way. The original mask is found to have grown outwards.
The numerical results obtained from a case study on one lung model are shown in Table 2. The formula used to obtain the quantity labeled Different Voxels Rate is

$$\frac{\sum_{i,j,k} mask_1(i,j,k) \veebar mask_2(i,j,k)}{|mask_1|}$$

The formula used to obtain the quantity labeled Growth Rate is

$$\frac{\sum_{i,j,k}\big(mask_1(i,j,k) \veebar mask_2(i,j,k)\big) \wedge \big(mask_1(i,j,k) == 0\big)}{\sum_{i,j,k} mask_1(i,j,k) \veebar mask_2(i,j,k)}$$

Compared Masks	Different Voxels Rate	Growth Rate
Original vs. Marching Cubes	0.505%	100%
Original vs. Smooth Mesh	0.503%	98.27%
Marching Cubes vs. Smooth Mesh	0.217%	0.453%

Table 2 – Case study results for mesh extraction

4.3. Hole Filling

4.3.1 Planar Patching

Two parallel methods have been examined for the planar patching step. Table 3 shows a comparison of the results in an example case, where a sphere's hole was patched. The elapsed computation time and the number of produced triangles are indicated in the table.

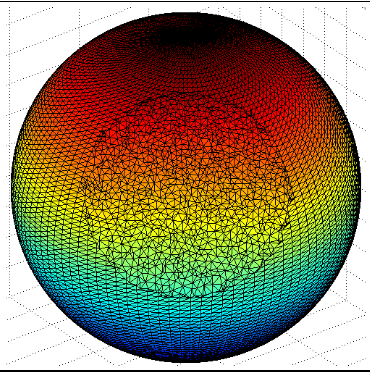
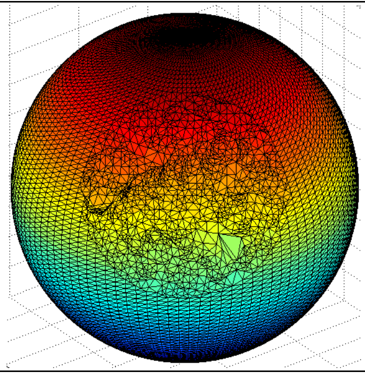
	
Method 1	Method 2
130 seconds	52 seconds
2502 new triangles	2402 new triangles

Table 3 – Example comparsion between two planar patching methods

4.3.2 Final Results

The complete algorithm was tested on a set of sphere meshes as well as lung meshes. A satisfactory hole-filling algorithm should produce a mesh that closely resembles the original mesh. In order to assess this measure for the methods used here, holes are created in known meshes and the results of the filling are compared to the original meshes. When working with lungs, holes are deliberately created in healthy parts of the lung. This facilitates an evaluation the algorithm's ability to restore a healthy model out of lungs with holes, and stands in contrast with the actual application, where holes would be created on thickened parts to eliminate them.

To obtain a numerical evaluation to the meshes' similarity, we use the tool called Metro described in (Cignoni, Rocchini, & R., 1998). Two measures are acquired from the tool. The first is the average error length, which is calculated by averaging the closest distance of each vertex on the patching mesh to the surface of the original mesh. The second quantity is the Hausdorff distance, which is the maximum of the aforementioned distances.

Table 4 through Table 8 present the results of the hole filling algorithms in different situations. Along with the figures, each table contains information about the hole diameter, the time required for computation on our system, the average error length, as well as the Hausdorff distance.

Table 4 and Table 5 show the outcome of experimenting with spheres. The former has a relatively small hole compared to the latter. Table 6 shows the results in a case of a hole on a lung surface surrounded by a relatively planar surface. Note that the cut-out surface featured an inward fold, which was later restored in the hole filling process. A case with larger hole diameter is shown in Table 7. This hole is located in a relatively convex location on the lung surface. Lastly, Table 8 shows the results of filling hole that is surrounded by complex geometry, i.e. there exists a lot of geometric variation around the boundary of the hole.

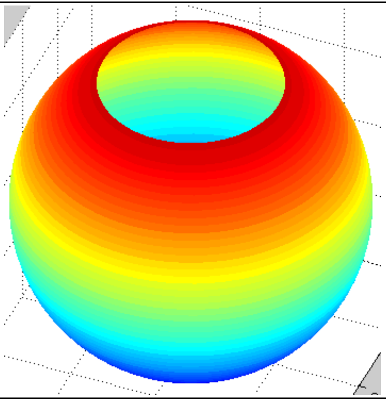
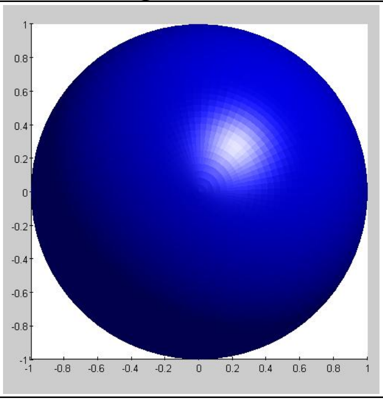
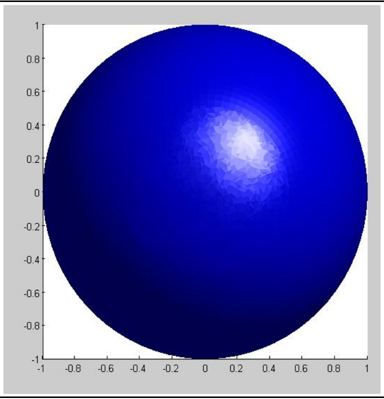
Mesh with hole	**Original Mesh**	**Patched Mesh**
		
Hole diameter	50% of sphere diameter	
Elapsed time	< 1 min	
Average error length	0.51% of sphere diameter	
Hausdorff distance	1.48% of sphere diameter	

Table 4 – Sphere with small hole

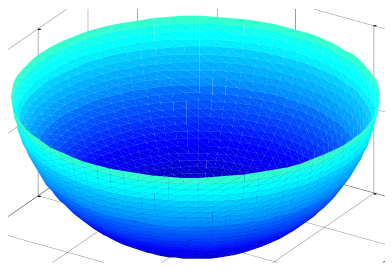
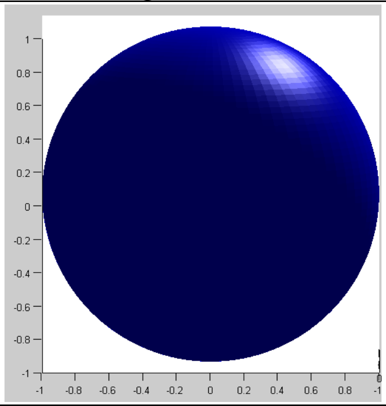
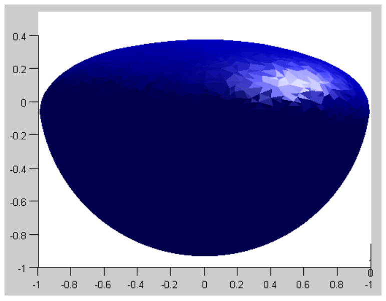
Mesh with hole	Original Mesh	Patched Mesh
		
Hole diameter	75% of sphere diameter	
Elapsed time	< 1 min	
Average error length	11.54% of sphere diameter	
Hausdorff distance	43.09% of sphere diameter	

Table 5 – Sphere with large hole

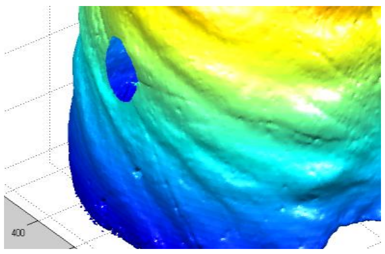
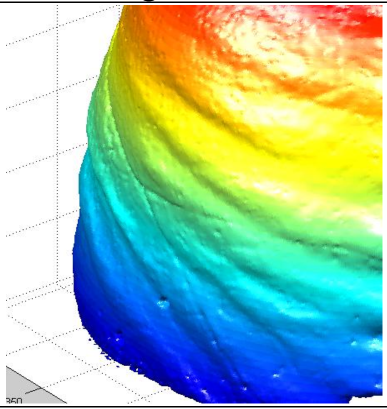
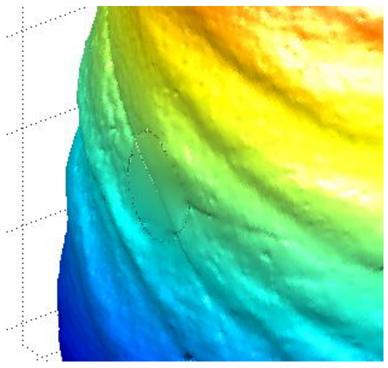
Mesh with hole	Original Mesh	Patched Mesh
		
Hole diameter	25 voxel edges	
Elapsed time	~15 min	
Average error length	0.71 voxel distance	
Hausdorff distance	0.22 voxel distance	

Table 6 – Lung hole with fold

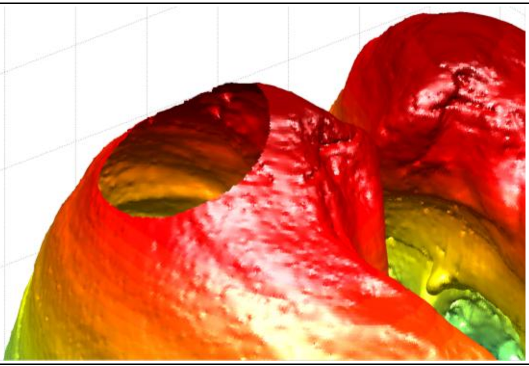
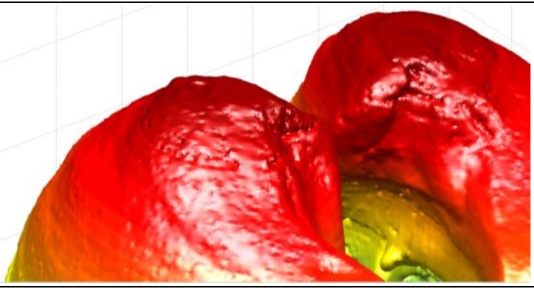
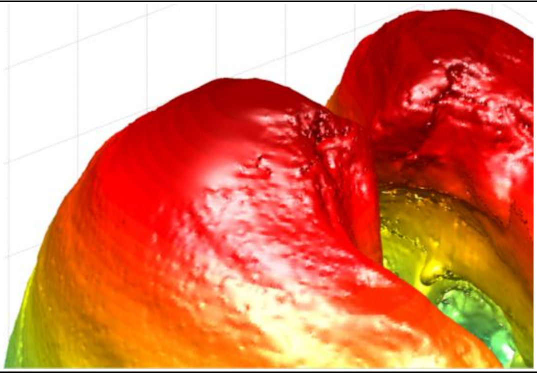
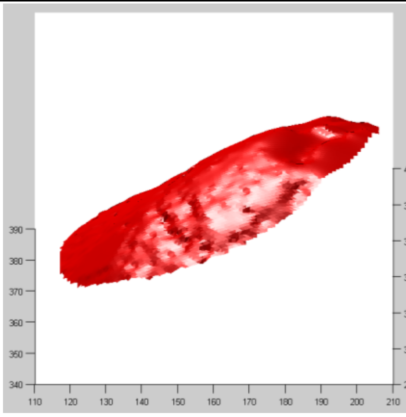
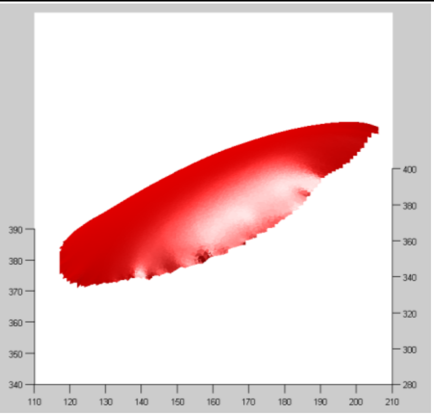
Mesh with hole	
Original Mesh	
Patched Mesh	
 340 350 360 370 380 390 110 120 130 140 150 160 170 180 190 200 210 280 300 320 340 360 380 400 Cut-out surface	 340 350 360 370 380 390 110 120 130 140 150 160 170 180 190 200 210 280 300 320 340 360 380 400 Patching surface
Hole diameter	100 voxel edges
Elapsed time	~50 min
Average error length	0.71 voxel distance
Hausdorff distance	3.25 voxel distance

Table 7 – Lung with large hole

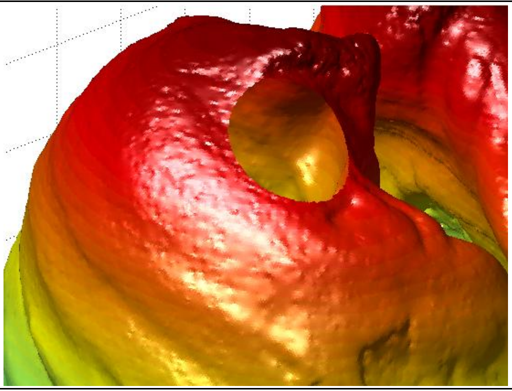
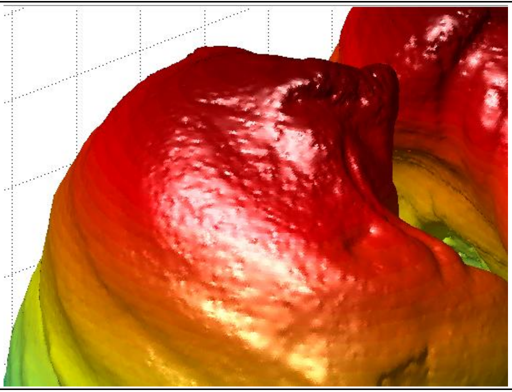
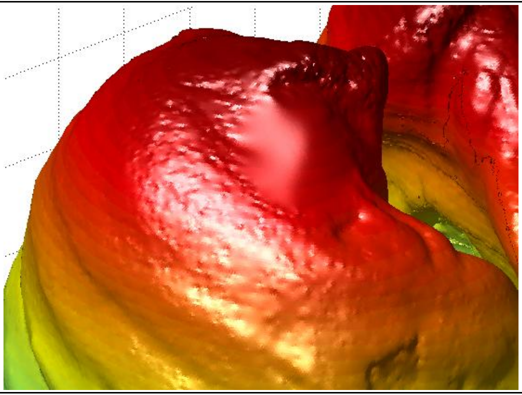
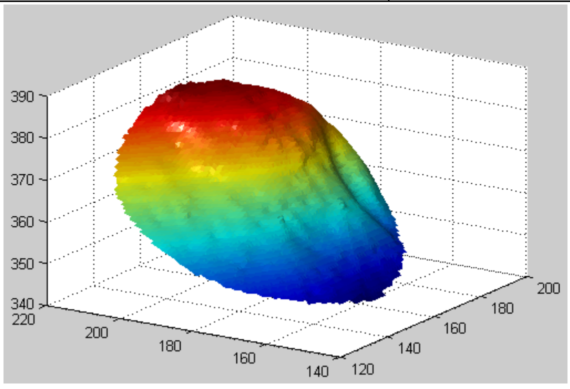
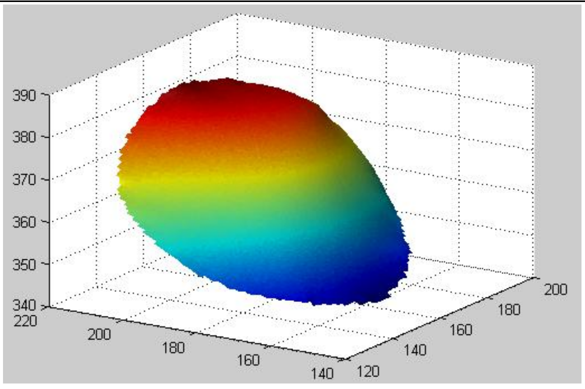
Mesh with hole	
Original Mesh	
Patched Mesh	
 Cut-out surface	 Patching surface
Hole diameter	80 voxel edges
Elapsed time	~30 min
Average error length	0.99 voxel distance
Hausdorff distance	3.81 voxel distance

Table 8 – Lung hole with complex geometry

4.4. Effects of Different Weight Functions

The Umbrella Operator U(v) uses a weight function $\omega(v_1, v_2)$. This weight to specifies a bias towards certain neighbors of v during the calculation of the average deviation of v. We have investigated different ω functions and examined their effects on the obtained patching mesh. The properties of interest are the patch's convexity as well as the shape of the mesh triangles on it. The weight functions introduced in Section 3.3.3 are tested on a sphere. The results are shown in Table 9.

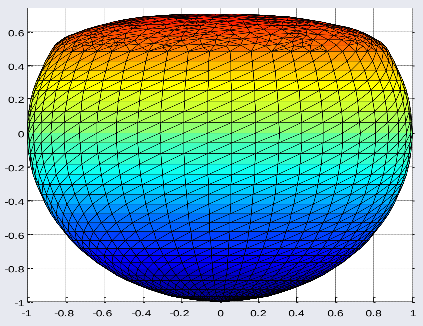
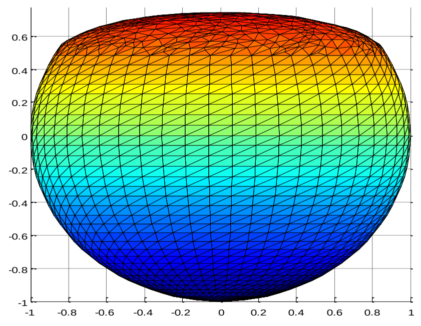
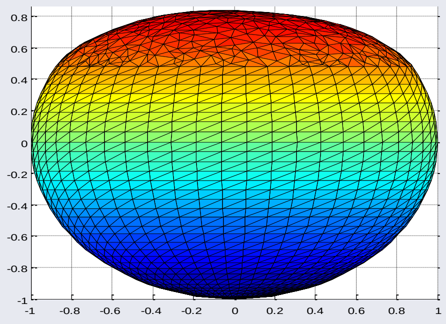
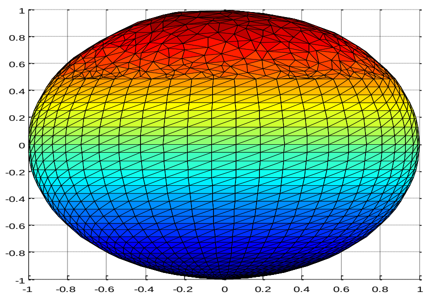
$\omega(v_1, v_2) = 1$ • Restored mesh too different from original • Tends to make triangles have similar area (Desbrun, Meyer, Schröder, & Barr, 1999)	
$\omega(v_1, v_2) = \lvert v_1 - v_2 \rvert$ • Less distortion, smooth • Restores original triangle sizes (Desbrun, Meyer, Schröder, & Barr, 1999)	
$\omega(v_1, v_2) = cot(\measuredangle(v_1, v_{k1}, v_2)) + cot(\measuredangle(v_1, v_{k2}, v_2))$ • Smoothest, best form • Triangles have better shape (Desbrun, Meyer, Schröder, & Barr, 1999)	
$\omega(v_1, v_2) = \begin{cases} \dfrac{\cot(\measuredangle(v_1, v_{k1}, v_2)) + \cot(\measuredangle(v_1, v_{k2}, v_2))}{\min\limits_{v_b \in V_B}(\lvert v_2 - v_b \rvert) \Big/ \min\limits_{v_b \in V_B}(\lvert v_1 - v_b \rvert)} & \text{if } v_2 \notin V_B \\ \dfrac{\cot(\measuredangle(v_1, v_{k1}, v_2)) + \cot(\measuredangle(v_1, v_{k2}, v_2))}{\alpha} & \text{if } v_2 \in V_B \end{cases},$ • More convex • More distortion • Triangle density lowered	

Table 9 – Results of investigated weight functions

Chapter 5 Discussion

The measured processing times are not adequate for real-time applications. However, the algorithms have been examined in a computational environment with limited programming efficiency. We expect better implementations to improve the CPU times by a factor of ten or more.

One of the slowest steps is the Mesh Extraction, where the preliminary redefinition of the voxel data is the largest contributor to the required computation. The step becomes rapidly more expensive for a larger number of voxels in the input data. Visual inspections of results show that the smoothing step is clearly effective. A smooth, noise-free mesh is necessary for the subsequent hole filling procedures, which require a smooth boundary. At the same time, it causes close to no loss in the original data – sometimes it even appears to cause less loss than the direct application of Marching Cubes. As shown in the example outlined in Table 2, the error rates between the original mask and the obtained ones is systematic. It is almost entirely produced by an outward growth of the mask. This can be attributed to the mesh-to-voxel conversion process, which creates one-valued voxels around the outermost vertices, as described earlier. In this manner, it performs a kind of “rounding up” of the surface towards the outer direction. A direct comparison between Marching Cubes with and without voxel data redefinition shows acceptably small variation, while the smooth result allows much better patching.

It is also apparent that the second of both planar patching methods is significantly more efficient, as mentioned in the literature. This is because Method 1 depends on the number of vertices in a $O(n^3)$ manner. Visual inspection of Table 3 shows a tendency for Method 1 to produce more homogenous triangle shapes with more even distribution. However, as long as the triangle density achieved by both methods is similar, this is of little importance with respect to the whole process. The reason being that the subsequent fairing step is almost completely responsible for the correct positioning of the newly-created vertices. And while the mesh fairing may be affected by the initial positions of these vertices (if they are referred to by the weight function), such effect can be eliminated by reapplying the fairing step multiple times. This iterative reapplication of smoothing is most of the time going to still be faster than using Method 1 for planar patching, esp. for large holes. Therefore, as far as our application is concerned, both methods are considered to produce comparably adequate results. Consequently, Method 2 becomes the more preferred method for its efficiency.

The complete hole-filling algorithm generally achieves acceptable error distances. We observe that it generally performs better for smaller holes. Moreover, the errors rise considerably when the hole is surrounded by convex geometry. Most of the regions of interest in lung surfaces (i.e. regions with thickenings that should be graphically removed) are relatively planar. Nevertheless, Table 9 shows adaptations of weight functions that can potentially solve convexity problems that are clear in the result of Table 5, for example.

Generally, the most preferred omega function is third one in Table 9, because of its adaptively smooth outcome, which also features the best triangle shapes. In case controllably more convexity is required, the last omega function may be appropriate. We note, however, that for this function, essential aspects of the patch are disrupted: proper triangle density and distortion-free smoothness. These effects can be repaired by reapplying the refinement (outlined in Section 3.3.21) and repeating the fairing step with a different weight function.

In Table 6, it is shown that the studied hole-filling procedure was successful in interpolating a fold that is featured in the surrounding mesh, restoring the complete fold in the patching mesh. However, Table 8 shows that more complex geometries are more challenging and might produce larger errors; the patching surface will not interpolate all the geometric features that are implied by the surrounding mesh. This is especially the case if there exist multiple, different features of that type. In other words, if geometric variation around the hole boundary is high with respect to the hole size, the algorithm might fail in interpolating geometric features such as folds, and could resort to a smooth solution that "averages out" the effects of such features.

Furthermore, the fairing algorithm only takes into account direct the surroundings hole. Specifically, only the vertices that are in 2-connectivity to the patching mesh are used to construct the LSE. Hence, geometric forms that are not sufficiently represented in the direct vicinity of the hole are not interpolated. This might be dissatisfying to intuitive expectations, when a widely predominant geometric form is expected to be interpolated, even when it is not to be observed in the direct vicinity of the hole boundary. On the other hand, this phenomenon might be beneficial in applications that would use a step-by-step procedure to eliminate thickenings.

Chapter 6 Conclusion and Prospects

We proposed a method for the assessment of the volume for lung thickenings that are featured in CT data. The thought technique is to produce a healthy lung model out of the provided one. The method consists of replacing diseased areas from lung surfaces with surface holes and then filling these hole smoothly based on the remaining surface. We evaluated the algorithm by applying it in cases where healthy parts of the lungs are removed and comparing the restored form to the original one. From there we draw conclusions about how realistic the results of the graphical “healing” process are going to be.

Overall, the algorithm is expected to perform well in most of the cases of our application. Healthy lung models can be closely approximated using the proposed method. This valuation is based on the errors measured, which turned out to be acceptable. In general, the algorithm performs specifically well if holes are small compared to the geometric variation of the surface. From another perspective, we can conclude that because most of the lung surfaces are of planar form around their thickenings, the conditions set by our application are compatible with the algorithm.

More complex geometric features require workarounds. We have shown a possible adjustment for convexity requirement. It is suggested that further research considers custom geometric requirements for hole filling results.

As for the computation time, we have gathered an alternative method to the triangulation step which is more efficient. There is room for efficiency improvement in steps such as mesh extraction and mesh fairing. In case of large thickenings, step-by-step elimination using smaller holes may produce interesting results.

List of Figures

List of Tables

Bibliography

Alleman, J. E., & Mossman, B. T. (1997). *Asbestos Revisited.* Scientific American.

American Cancer Society - Asbestos. (2010, August 11). Retrieved August 4, 2011, from American Cancer Society: http://www.cancer.org/Cancer/CancerCauses/OtherCarcinogens/IntheWorkplace/asbestos

Barequet, G., Dickerson, M., & Eppstein, D. (1998). On triangulating three-dimensional polygons. *Computational Geometry: Theory and Applications* , 155-170.

Brunton, A., Wuhrer, S., Shu, C., Bose, P., & Demaine, E. D. (2009). Filling holes in triangular meshes by curve unfolding. *Shape Modeling and Applications* , 66-72.

Carr, J. C., Fright, W. R., & Beatson, R. K. (1997). Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging* , 96-107.

Chaisaowong, K., Bross, B., Knepper, A., Kraus, T., & Aach, T. (2008). Detection and Follow-Up Assessment of Pleural Thickenings from 3D CT Data. *Proceedings of ECTI-CON , vol.1*, pp.489-492.

Chaisaowong, K., Knepper, A., Kraus, A., & Aach, T. (2007). Application of Supervised Range-Constrained Thresholding to Extract Lung Pleura for Automated Detection of Pleural Thickenings from Thoracic CT Images. *Proc. SPIE Medical Imaging: Computer-Aided Diagnosis* , 65143M.

Cignoni, P., Rocchini, C., & R., S. (1998). Computer Graphics Forum. *Blackwell Publishers , 17* (2), 167-174.

Curless, B., & Levoy, M. (1996). A volumetric method for building complex models from range images. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* , 303-312.

Davis, J., Marschner, S. R., Garr, M., & Levoy, M. (2002). Filling holes in complex surfaces using volumetric diffusion. *Processing of 3D Data Processing Visualization and Transmission, , IEEE Computer Society,* 428–433.

Desbrun, M., Meyer, M., Schröder, P., & Barr, A. H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* .

Dey, T. K., & Goswami, S. (2003). Tight Cocone: A Water-tight Surface Reconstructor. *J. Comput. Inf. Sci. Eng. 3* , 302.

Deza, M., & Rosenberg, I. G. *Combinatorics 79* (Vol. 2).

Kaufman, A., Cohen, D., & Yagel, A. (1993). Volume graphics. *IEEE Computer , 26* (7), 51-64.

Kobbelt, L., Campanga, S., Vorsatz, J., & Seidel, H.-P. (1998). Interactive Multi-Resolution Modeling on Arbitrary Meshes. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* .

Lempitsky, V. (2010). Surface Extraction from Binary Volumes with Higher-Order Smoothness. *IEEE Computer Vision and Pattern Recognition (CVPR)* , 1197–1204.

Liepa, P. (2003). Filling holes in meshes. *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* , 200-205.

Lorensen, W. E., & Cline, H. E. (1987). Marching Cubes: A high resolution 3D surface construction algorithm. In: Computer Graphics. *Computer Graphics , 21* (4).

Virta, R. L. (2006). *Worldwide Asbestos Supply and Consumption Trends from*. Retrieved August 04, 2010, from U.S. Geological Survey: http://pubs.usgs.gov/circ/2006/1298/c1298.pdf

Wagner, J. C., Sleggs, C. A., & Marchand, P. (1960). Diffuse Pleural Mesothelioma and Asbestos Exposure in the North Western Cape Province. *Br J Ind Med.* , 260–271.

Zhao, W., Gao, S., & Lin, H. (2007). A robust hole-filling algorithm for triangular mesh. *The Visual Computer , 23* (12), 987-997.