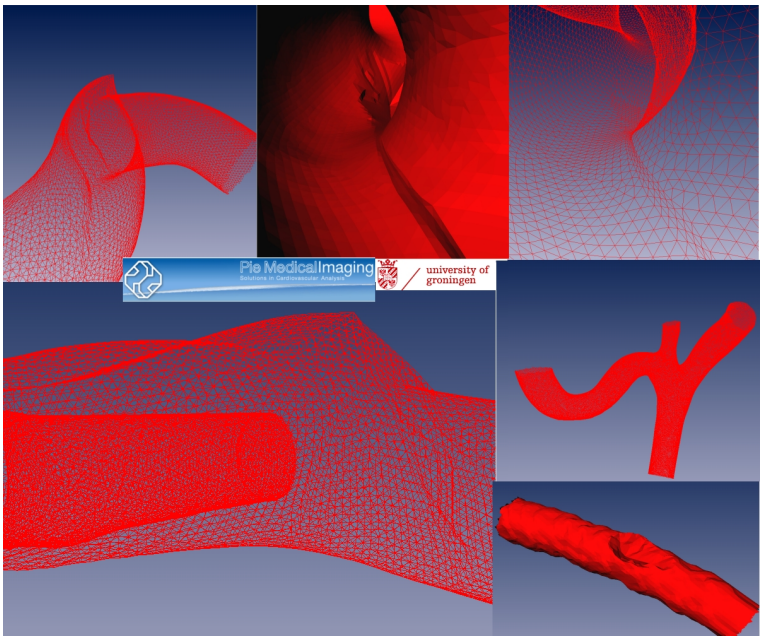
3D INTERACTIVE MESH DEFORMATION

ROELOF ANNE SCHOENMAKER



April-September 2009

Roelof Anne Schoenmaker (roelofanne@gmail.com):

SUPERVISORS:
prof. dr. J.B.T.M. Roerdink
dr. M.H.F. Wilkinson
drs. C. Consten

LOCATIONS:
Maastricht-Groningen

TIME FRAME:
April-September 2009

ABSTRACT

In this thesis a '3D interactive mesh deformation' - method tuned for vessel-trees is developed. During the literature research a number of interesting methods is introduced. One of these methods is selected, than a specific approach is deduced in theory, to satisfy our needs, and tested after prototyping and implementation. Our method is evaluated and medical issues as ageing of the vascular lumen are discussed.

'Mens-zijn, een goed mens zijn',
dat is het enig belangrijke
in deze wereld!
Maar wie interesseert het?
...
De computer houdt geen rekening met het hart.

— Phil. Bosmans

ACKNOWLEDGMENTS

I would like to thank my parents for their endless support and patience during this thesis and preceding studies.

For this thesis I would like to thank drs. C. Consten from Pie Medical Imaging b.v. (in Maastricht, the Netherlands). His guidance and support during e.g. the prototyping and implementation was of great value. Furthermore I would like to thank prof. dr. J.B.T.M. Roerdink for asking questions and correcting previous versions of this thesis.

For their interest and support I would like to thank: dr. M.H.F. Wilkinson, several persons from Pie Medical Imaging b.v. and all other people, which I have forgotten to mention.

CONTENTS

If we knew what it was we were doing,
it would not be called research, would it?

— Albert Einstein

1 INTRODUCTION

The meanings of (simplex) mesh and triangulation will be introduced in the literature research.

At Pie Medical Imaging b.v. (in Maastricht, the Netherlands) quantitative analysis software for cardiology and radiology is developed. For 3D vessel analysis applications, segmentation algorithms produce a 3D triangulation (or the dual of it, called the simplex mesh) of the vessel or vessel-tree.

In case the segmentation process results in a less successful segmentation, an easy to use correction method must be available. Since working in 3D space, an intuitive correction method would be interesting. This method should allow manipulation of the 3D object surface and the surrounding surface must preferably adapt in a natural way (in other words, it should mimic real tissue behavior).

1.1 FRAMEWORK

This master thesis concerns the development of a '3D interactive mesh deformation' - method tuned for vessel-trees. The desired C/C++ library will take as obligatory input a 3D triangulation (vessel-tree) and the wanted initial change of a specific vertex in this vessel-tree. Most likely, other input values are needed as well to gain a better result. Think for example at the original (MRI) image data of the vessel-tree or the viscosity of the vessel-wall. The output of this method should than contain a "natural" modified 3D triangulation.

Let us depict the framework for the deformation method as:

- A "black" box for the '3D interactive mesh deformation' - method.
- An input arrow for the triangulation to deform.
- A second input for the specific initial vertex and the wanted initial displacement in the triangulation.
- An input for the "natural" parameters, such as elasticity of the vessel-wall and image data of the corresponding segmentation.
- An output arrow with the deformed triangulation.

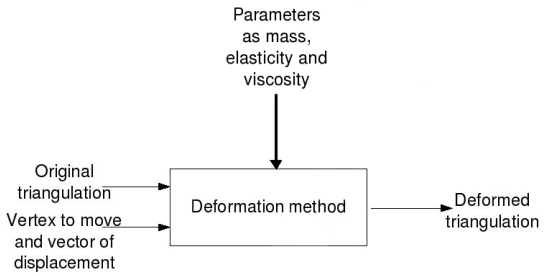


Figure 1: Framework for deformation method.

1.2 REQUIREMENTS

To decide which method, of the discussed methods in the literature research, is the most appropriate one, a number of clear requirements are needed. These essential requirements are:

- The method should correct (with help of a specific moved vertex) a 3D triangulation.

The first priority is deforming a triangulation, or actually, the triangulation from the segmentation process needs to be corrected. Sometimes a match between the image data and the triangulation is not perfect, therefore a correction needs to be done.

Let us consider the idea that we do not have the image data, but that we would like to deform a triangulation. This is the key solution to our problem. If we manage to find a method that can deform a triangulation, then this is certainly a possible candidate.

The 3D triangulation should be maintained, such that the existing edges are not removed and no new edges are added. However, with this constraint, a problem might arise during the results in part v.

- This "hard correction" on the triangulation must be done on a local area of influence.

This correction, without the image data, is labeled as a "hard correction" on the triangulation. However, should we deform the total triangulation or just a (small) part?

Usually the corrections are small regions of the entire triangulation and therefore the final method should at least be able to control the "local area of influence". As a matter of fact, if we need to deform the whole triangulation, it is better to compute a new segmentation.

- The triangulation must thus be "locally" corrected, but it should mimic real tissue behavior and it needs to behave differently for vessels and bifurcations.

If we deform a local area of the triangulation, it would be good to use a correction on the surrounding surface that adapts the surface "naturally". To determine "natural" better, it can be rephrased to "mimic real tissue behavior". This statement includes thoughts about the properties of the vessel-wall, such as elasticity and viscosity.

On page 3 a bifurcation, with connected vessels (partly displayed), is shown. This sketch shows also the Carina of the bifurcation. The tissue of the Carina is stiffer and it has no (relative) flat surface as the surrounding tissue, therefore a deformation in a bifurcation (especially when it starts in the carina) will behave differently than a deformation of a vessel.

- When the image data is taken into account with the corresponding triangulation, then a "soft" correction must be done on the local area of influence.

Let us now add the image data to the requirements. If the image data could be used by the final method, then the correction is not longer called "hard". This new "soft" correction uses the image data (e.g. gradient image value) for guiding the deformation process close

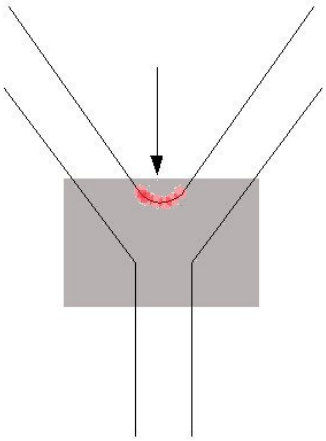


Figure 2: A bifurcation (approximately the shaded area) between vessels and with an arrow at the (red) Carina.

to the corresponding image locations, also it still should more or less mimic real tissue behavior.

With these four requirements in place:

1. The literature research and prototyping will deliver a final method, which could satisfy all four.
2. The theory of the prototype and implementation are discussed and how they relate to these requirements.

Part I

LITERATURE RESEARCH

There is nothing either good or bad -
but thinking makes it so

— William Shakespeare (Hamlet: Act II - Sc. II)

2 INTRODUCTION

Deformation methods do exist in a wide variety of flavors these days. In this literature research several ideas will be presented. However, to determine which method will solve the problem, the previous stated requirements should be met.

In order to understand the several flavors, it will be good to refresh or up-date the geometric knowledge. Therefore this literature research is starting with some basic geometric knowledge. After these geometric definitions, some initial ideas to correct triangulations are proposed. Also methods which could be used in deformable models are discussed, as for example the well known Finite Element Method and Free-Form deformation methods.

Finally an interesting method is selected, which depends on the drawn conclusions (with help of the requirements) from the researched literature.

3 GEOMETRIC BASICS

To understand the several flavors of the methods to come, two fundamental geometric structures are introduced here. The triangulation is first and afterwards the simplex mesh.

3.1 TRIANGULATION

The general assumption of a triangulation is that it exists of vertices, edges and faces. A well know type of a 2D triangulation is the Delaunay triangulation (a so-called balanced triangulation) as proposed by Delaunay in [10].

vertices ⇒ points
edges ⇒ lines
faces ⇒ triangles

The vertices of 2D triangulations are depending on the x- and y-coordinates of the used vertex set. An edge is the connection between vertices and a face in a triangulation is a triangle, thus with 3 vertices and 3 edges. A 3D triangulation is generally the same, although now a vertex carries x-, y- and z-coordinates.

A very simple 3D triangulation might be a triangle-based pyramid, but also the surface of more elaborate 3D objects can be triangulated. Such 3D objects are cubes, cylinders and spheres, as well as the vascular lumen (see figure 3 on page 8 for two examples).

Formally a 2D Delaunay triangulation is a planar graph ([10] and [4]) with:

- A vertex $p_i = [x_i \; y_i]$, with coordinates $x_i, y_i \in \mathbb{R}$.
- A vertex set
$$P = \bigcup_{i=1}^{N} p_i,$$
where N is the number of vertices in the set.
- A set of valid non-crossing Delaunay edges. A Delaunay edge can only be valid if and only if, the two vertices of the edge are on the boundary of a circle. In this circle no other vertex of P should exist, which assures that there are no crossing edges in the triangulation.
- A set of valid faces. A valid face (triangle), which consists of 3 vertices and 3 edges, is only valid if those vertices are on the boundary of the circle and no other vertex of P is inside this circle.

In 3D the Delaunay properties are also valid; however each vertex now becomes $p_i = [x_i \; y_i \; z_i]$ and uses the coordinates $x_i, y_i, z_i \in \mathbb{R}$. Each 3D face has a normal to determine how it is oriented, in other words, which side of the triangle is part of the inside space and which is part of the outside space of the 3D object.

3.2 SIMPLEX MESH

The 2D Delaunay triangulation has the Voronoi diagram as its dual and this dual is also known as the 2-simplex mesh. The k-simplex mesh is

Figure 3: Sketch of a triangle-based pyramid and a possible surface triangulation of a 3D cylinder.

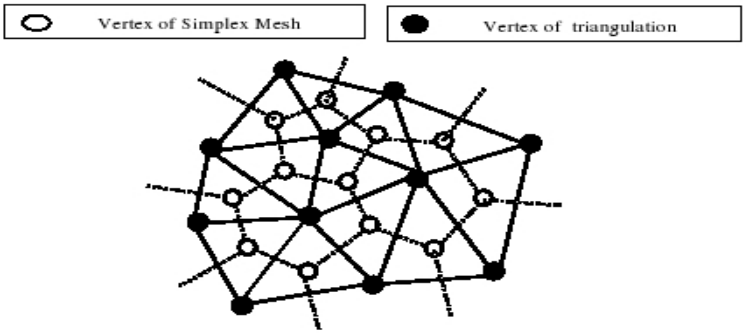


Figure 4: Taken from [11], which is showing the dualism for $k = 2$.

introduced by Delingette in [11], where $(k+1)$ denotes the number of connected edges to a simplex mesh vertex.

The dualism of a k-simplex mesh and a k-triangulation could be illustrated for example with the case $k = 2$ (see figure 4), where:

- Triangles of a 2-triangulation $\leftrightarrow$ Are vertices in a 2-simplex mesh.
- Vertices of a 2-triangulation $\leftrightarrow$ Are corresponding to a 2-face in a 2-simplex mesh.

This is valid in 2-dimensions, however not in higher dimensions. In higher dimensions it is not possible to generate a triangulation directly from a simplex mesh. On the other hand **a k-simplex mesh can be generated from any k-dimensional given triangulation**, as proposed in [11], which might be useful for our wanted purpose.

4 INITIAL SUGGESTIONS

Let us consider two initial ideas for deforming 3D triangulations (and from the last chapter, it is also known that it could be valid for meshes).

4.1 MOVE THE VERTEX

The most basic black box interpretation of figure 1 (see page 1) takes the two crucial inputs:

- The triangulation to deform.
- Vertex to move (including wanted displacement!).

Moves the given vertex of the input triangulation to the desired position, and the method puts the changed triangulation on the output.

Although this method is very simple and can correct the triangulation locally, the following issues arise:

- The deformation is not able to mimic tissue behavior, because only one vertex is moved.
- Neither is it able to select difference in behavior for vessels and bifurcations (it is one 'fixed' movement).
- Also it does not incorporate the image data for the "soft" correction.

Therefore this suggestion is not suitable at all for the wanted purpose.

4.2 AVERAGE CHANGE METHOD

The second idea for constructing a deformation method, that might satisfies more or all requirements, is our so-called Average Change Method. The idea is to average the change over the triangulation, where the decreasing value of the initially changed (moved) vertex propagates through its neighbors.

An attempt for averaging out the displacement of the initially moved vertex could be:

Notice that the $*$ is an element-wise multiplication!

$$p(t+1)_j = p(t)_j + 0.5 * (p(t)_i - p(t)_j) \tag{4.1}$$

where:

- $p(t)_i$ is the fixed 3D position of the moved previous ring neighbor vertex (at the start it is the center vertex).
- $p(t)_j$ is the current 3D position of the neighbor vertex.
- $p(t+1)_j$ is the new 3D position of the neighbor vertex.

In figure 5 a partly result of a triangulation at time step $t = 1$ is presented (red) and also the corresponding triangulation part (yellow) before the initial move (at $t = 0$) of the center vertex. In this example

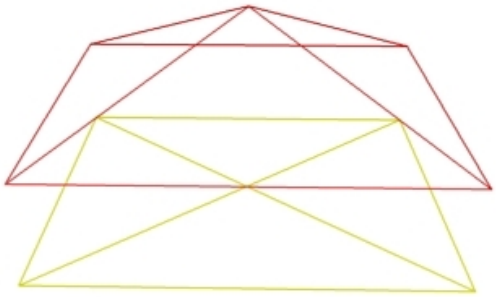


Figure 5: Average Change Method example result.

only the z-direction is changed of the center vertex, but also the x- and y-direction of the neighbors will move a little bit.

Due to the special design of the formula, it can be used for circular propagation (outward direction). When all neighbors are done on the first circle around the initial center, all these ones could be used as "centers" and the outward second (from initial center!) circle could be used as their neighbors and so on. However, somehow the process needs to be stopped, which could be done with some kind of threshold or perhaps with a fixed number of propagation levels. Also some other considerations, such as organizing which neighbor belongs to which center should be taken into account.

4.3 CONCLUSION

The first suggestion did not suited the wanted requirements, but is the second suggestion applicable?

The Average Change Method might be a possible solution for our "hard" correction. However, it is not a medically motivated deformation of the local area (there is no difference in deformation between a dent in some vessel and a bifurcation!) and it certainly doesn't take the image data into account for the "soft" correction.

For a first try this method comes closer than the initial suggestion, however it misses out on the already mentioned issues. Moreover, numerous papers are hinting at the subject of deformable models. These methods are using for example forces- or energy-based equations, with inter alia various Element Methods for solving these equations. Let us therefore now look into more elaborate existing methods.

5 DEFORMABLE MODELS

In order to understand some of the flavors of so-called deformable models, the papers of [27] and [30] are reviewed. These papers give an overview of several deformable model ideas in e.g. medical imaging analysis, with for example:

- **Energy-Minimizing Deformable Models;** to minimize the energy, deformable contour models (such as snakes) are used. More information about such snakes, snake pedals and other active modeling techniques can be found for example in [21], [25], [27], [28] and [47].
- **Dynamic Deformable Models;** this type of models unifies the description of motion and shape. The resulting method is useful for moving medical images, because the deformable objects are analyzed in e.g. frequently bending motion.
- **Probabilistic Deformable Models;** when prior knowledge exists in terms of probabilities, then this type might well do the job. In general the probabilistic framework provides also error and uncertainty measurements (just as snakes), for example, how well it fits into the image data.

Furthermore the authors from [27] came to the conclusion that deformable models are very interesting for medical imaging analysis. Also they concluded:

- That a deformable model is able to represent complex shapes and have an extensive shape variability of anatomical structures.
- Also these models overcome many of the limitations of traditional low-level image processing methods.

In the end all of the mentioned models boil down to some sort of action for minimizing the energy, force or motion equation(s) for the mesh or triangulation model. Actually this is most likely the way our final solution will also take, than however questions arise such as: *which equation and method is optimal for our problem?*

To find the answers on e.g. these questions and solve the current problem, some Element Methods will be discussed in the next chapter and a few other (more diverse) methods are presented after that.

6

CONSIDERING ELEMENT METHODS

The Finite Element Method is introduced e.g. in [30]. This method is often used for solving different kinds of problems, which makes it interesting to look into. Also a few other related Element Methods will be discussed in this chapter.

6.1 FINITE ELEMENT METHOD

One of the most used methods is the Finite Element Method (FEM). FEM is generally accepted as a solver for differential equations and can thus also solve the well known equation $Ax = b$, where x is usually the unknown.

To be able to use FEM or one of the next methods, the triangulations or meshes should be rewritten to a suitable equation form. Let us first consider the following general form:

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{6.1}$$

Here the symbols denoted with A are the deformation matrix. If the deformed triangulation b is known and also deformation matrix A, then the original triangulation coordinates in x can be resolved. Also the other way around works in this case; when the original coordinates of triangulation x and deformation matrix A are known, then the deformed coordinates of triangulation b can be computed. This can thus be solved by FEM; however the deformation matrix needs to be determined. (The basics of FEM and more are described quite clearly in [53].)

Internal data means here that the interior of a 3D triangulation contains vertices and edges and thus not alone a surface triangulation (as our final method would use).

In for example [30], FEM is used to solve an Energy Equation. [20] uses the matrix A as a symmetric and positive definite stiffness matrix and a Force vector b, to resolve the unknown vector x (which gives the displacements and possible rotations). The authors of [15] describe that FEM uses all data of a triangulation. This means that for a 3D deformation also the internal data is used.

In general FEM uses, as described by [37], the following steps for one resolving cycle:

1. Discretize the physical problem, such that a finite element "mesh" with vertices is constructed.
2. Choose interpolation functions, for interpolating the field variables over the element(s).
3. Find the element(s) properties, which can be done by different approaches such as: the Galerkin method (more information is found in [32]).
4. Combine local element equations for all elements to a global equation system.

5. Solve the obtained global equation system (typically sparse, symmetric and positive definite).

6. As last step it is often needed to compute additional results, which may concern, for example, in deformation problems the strains and stresses of the used objects.

Two questions that might arise on this moment are:

- *How many steps (cycles) of resolving are needed, to obtain satisfactory results?*
- *Which deformation matrices should be used, to retrieve the needed "mimic real tissue behavior" deformations?*

The latter might be solved with a Laplacian Matrix, but this is postponed to section 7.2.2 on page 18. However, the first question still needs to be answered. Nevertheless, there are several error measurements possible for FEM, which are important for tracking down possible irregularities.

FEM is interesting; however, it has also some weaknesses, such as the above mentioned repeated cycles. For each repetition the whole mesh or triangulation needs to be updated, thus redoing the complete computation step, time after time, is an important bottleneck. A possible solution is to assume a local deformation region (with small surrounding) as the whole mesh, but then the question arises: *how large should the local deformation and surrounding boundary area be taken?*

To tackle the global update bottleneck, eXtended FEM (XFEM) is introduced ([29]) and is modified for thin films and nanotubes ([34]). XFEM is developed in [3] with a minimal re-meshing for crack growing and also there exists ([29]) even a variation without re-meshing for crack growing. An example of a method that implements a (Large) FEM deformation can be found in [22], where it is incorporated into the Super-convergent Patch Recovery (SPR) method.

6.2 BOUNDARY ELEMENT METHOD

The Boundary Element Method (BEM) is another element method approach, which could solve the update problem. The authors of [9] have proposed an analysis of 3D problems with BEM; this is particular interesting because the used triangulations are in the end also in 3D. The 3D BEM version requires data, such as the material properties or surface tractions and displacements, where the changes are preferred.

For more in-depth information about BEM, [15] could be consulted.

The most important difference is that, following the authors of [15], the complete data domain Ω for FEM needs to be discretized and for BEM only the boundary $\Gamma = \delta\Omega$. In the current case the boundary might well be the complete surface triangulation as is shown by the authors of [19]. They developed a BEM which is able to produce interactive deformation on the boundary. Two of the rendered results of their method can be found on the next page.

However, in the used papers for BEM, they are concerned about more complicated object boundaries. Also the performance should be looked into, especially for the matrix equations. Nevertheless, it should be computed quicker (due to the smaller (boundary) dataset), than FEM.

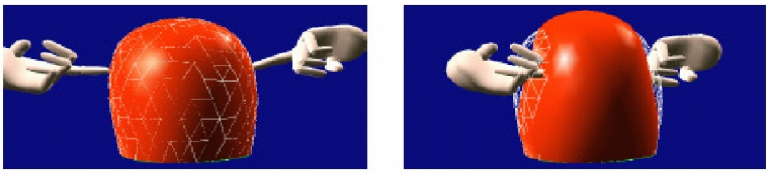


Figure 6: Render results of [19]. Left: undeformed 3D triangulation. Right: deformed 3D triangulation by the two hands.

6.3 NATURAL ELEMENT METHOD

Mesh-free or mesh-less methods are particle based methods, with for example Lagrange computations.

FEM uses a Finite Element data structure (such as a mesh) for computing each cycle, but recently some element methods have been developed as mesh-less or mesh-free methods. Interesting methods are the Natural Element Method (NEM), such as described in [41] or its variant the constrained NEM (C-NEM) from [51]. These Natural Element methods are replacing also the "shape functions" of FEM with natural neighbor functions.

The natural neighbor interpolation technique is based on the Voronoi Diagram and contains generally the following steps:

- From the cloud of vertices each Voronoi Cell T_I, corresponding to vertex v_I, from the Voronoi Diagram is constructed as: $T_I = \{v \in R^{dim} : d(v, v_I) < d(v, v_J) \forall J \neq I\}$ Where v are the vertices and when the dimension $dim = 2$, then the distance measure is: $d(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$.
- The natural neighbor - based interpolation follows now; one simple idea is the Thiessen or nearest neighbor interpolant over each Voronoi Cell:

  If $v \in T_I$, then $\phi_I(v) = 1$.

  If $v \notin T_I$, then $\phi_I(v) = 0$.
- With this interpolant only the nearest neighbor is taken into account, also numerous other (usually more usable!) interpolation techniques for NEM exist as is demonstrated in e.g. [13].

The authors of [13] made as one of the first, a comparison between FEM and NEM in 2008. A few results of their research are:

- It is a good alternative for FEM; also some programs are available online such as [42].
- Still, NEM also needs to be optimized, because computing it is a quite expensive job.

Although this seems like a promising method for the future, and with the knowledge that there are more particle methods, it surely could be a research area of interest. For the current problem, however, these NEM solutions do not preserve the existing triangulation or mesh edge-structures, which are important for further use after the deformation.

6.4 CONCLUSION

It seems to be that BEM is the most promising method for the current problem till now. However, the discussed issues should be looked into.

7 CONSIDERING OTHER METHODS

Let us consider beside BEM, a few other methods which might satisfy most or all of the wanted requirements (see section 1.2 on page 2).

7.1 GENETIC ALGORITHMS

Genetic Algorithms (GAs) are widely used, for example in mesh optimizations, but are also frequently mixed with other algorithms for performance reasons. In general, a GA is described ([1]) by the following steps:

1. Take selection from "healthy" pairs of classifiers (in the mesh-case, these are usually a large number of meshes!).
2. Generate copies of those pairs (meshes).
3. Modify the copies slightly by applying a “genetic operator” (e.g. bit-flipping in a general GA-case or changing the mesh slightly in the mesh-case).
4. Make selection of modified "unhealthy" pairs (meshes) for replacing.
5. Modify these pairs (meshes) by for example mutation of the data of these pairs (meshes).

In recent research GAs are proposed in a hybrid mix of a so-called Real Coded Genetic Algorithm (RCGA) and a Greedy Algorithm for mesh optimization. Papers that are related to this kind of hybrid combination are [43], [44] and [45].

These hybrid G(entic) A(lgorithm) GR(eedy) combinations do take in general the following steps (as in [43]) for each mesh:

1. Minimize globally the energy of the mesh, with use of a GA.
2. Minimize the resulting mesh from step 1, by a GR algorithm.
3. Adaptation of the resolution of the mesh (the resolution of the mesh is adapted to correspond to that of the image data).
4. Then the final minimization is done on the adapted mesh with a GR algorithm.

This hybrid combines the advantage for global minimization with GA, but reduces the slow local convergence of GA with help of the GR algorithm. However, also from previous research (such as [45]), it can be concluded that GAGR is still very slow in comparison with for example Dual Surface Minimization (DSM). However, DSM is only described by Tohka et all, and not by others as far as I know, which therefore might not be considered as relevant for the research field.

A different approach with a Genetic and Greedy algorithm is described by [39], which is based on Bayesian formulations. Also a

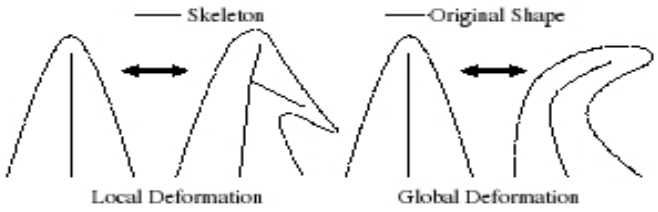


Figure 7: A local (adds a branch to the skeleton) and global deformation (skeleton and original shape are bending together) example from [50].

Bayesian stochastic mesh optimization for 3D reconstruction is proposed in [48]. Both ideas incorporate a likelihood formula which is based on the well known Bayes theorem, of course both in their own context.

Nevertheless, for GA and GAGR the incorporation of multiple meshes is simply too computational intensive for the purpose of the wanted interactive method.

7.2 FREE-FORM

Free-form methods are in different versions available, such as free-from curves ([52]), free-form surfaces ([24]), free-form patch modeling schemes ([2]) and free-form deformations for meshes as described in [50]. Free-form deformations (FFD) have global or local reshaping possibilities; a clear example of this is given in figure 7.

Free-form is generally based on Bezier curves or Bernstein-Bezier curves, where the deformation is evaluated by ([38]) the so-called "trivariate" (based on three independent values) Bernstein polynomial. Free-form methods are using two models for deforming; one is the original shape which needs to be deformed, and the other is the control mesh (in 2D called skeleton) which is used for the deformation.

These free-form methods seem to be very promising, also in 3D. Let us therefore look into a 3D free-form method, which might be interesting for solving the current problem. Also a number of associated methods will be discussed, such as Laplacian, Dirichlet and multi-resolution related free-form methods.

7.2.1 *Free-Form in 3D*

In [50] a global deformation of 3D meshes with FFD is proposed. Moreover in [23] several approaches are presented, such as global deformation and local deformation in indirect mode, and also both deformations can be handled in direct mode. The difference between the two modes is simple to state, when the indirect mode is used the user can only modify the control mesh B and not the original shape M. However in direct mode the user will modify the original shape "directly" and the system produces a modified control mesh. At the end both modes contain a control mesh B' that is modified, from this the modified original shape M' is computed. Two nice examples of results are shown in figure 8, which are both taken from [23].

The control mesh is usually created by the user, such that the topology and density (which controls the number of vertices and faces used) can

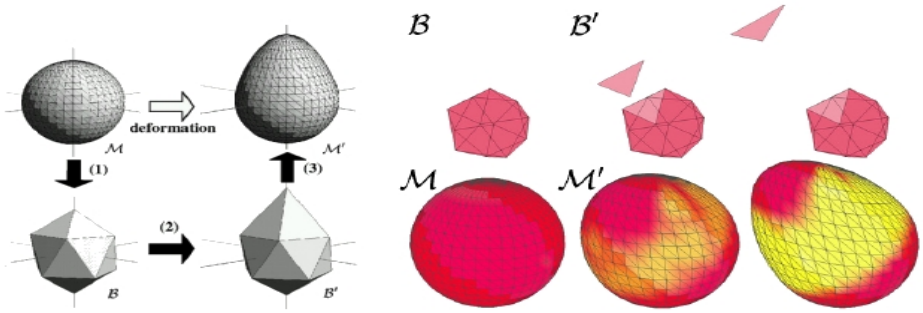


Figure 8: Left: The general modification process from [23]. Right: An example of deforming a sphere.

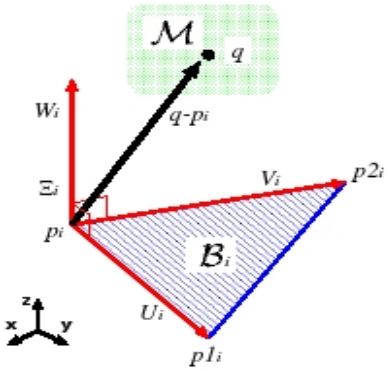


Figure 9: Illustration of parameterization, which is taken from [23].

be chosen. It is also possible to create (self-intersecting) disconnected triangles as control "mesh".

The theoretical steps from [23] are generally described as:

- **Parameterization;** the parameterization of an original shape vertex q, by a control triangle from B, which includes q, yields in [23] the following equation:

  $$q = p_i + u_i(p1_i - p_i) + v_i(p2_i - p_i) + w_i \frac{(p1_i - p_i) \times (p2_i - p_i)}{\sqrt{|(p1_i - p_i) \times (p2_i - p_i)|}}$$

  Where the vertices p_i, $p1_i$ and $p2_i$ are vectors, and vertex q is parameterized by local coordinate system $\Xi_i = (u_i, v_i, w_i)$. An illustration of the parameterization is shown in figure 9.

- **Weight Calculation;** a number of, for example, linear functions are available to compute the weight k_i of the local coordinate system. This weight indicates how relative its influence is against other local coordinate systems, which is important for the mapping process.

- **Modification of control mesh;** the example deformation on the right in figure 8 shows the idea of modifying the control mesh, by moving one triangle (with multiple original vertices) of the complete control mesh in the outward direction.

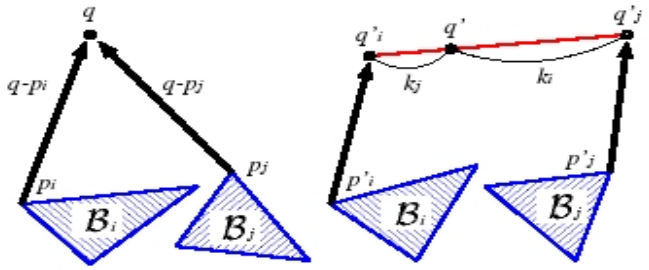


Figure 10: Mapping taken from [23].

- **Mapping;** after the modification, the mapping can be done with the formula:
  $q'_i = p'_i + u_i(p1_i - p_i)' + v_i(p2_i - p_i)' + w_i(\frac{(p1_i - p_i)\times(p2_i - p_i)}{\sqrt{|(p1_i - p_i)\times(p2_i - p_i)|}})'$
  When the weights are incorporated, q' becomes:
  $q' = \frac{\sum_i k_i q'_i}{\sum_i k_i}$
  This mapping is also depicted in figure 10.

7.2.2 *Laplacian*

Laplace and Dirichlet were famous mathematicians from the 18^{th} century and their discoveries are the fundaments of many theories and methods nowadays. Original methods with, for example Laplacian, are defined as continuous formulae. However this is not feasible for (limited) meshes, therefore discrete methods ([49]) are developed. Discrete methods which incorporate Laplacian or Dirichlet formulae are presented in respectively [40] or [18]. Laplacian methods are general good in changing local parts, without changing far away vertices. Also mesh optimization could be done with help of the Laplacian matrix as in [31]. Let us therefore now investigate what these ideas could add to free-form deformation methods.

A method that describes a Dirichlet approach for noisy 3D data is presented by Ilic and Fua in [18]. The general idea is the same as the above free-form methods, but the main difference, the authors are stating, is that their function has the ability to place control vertices at arbitrary positions against the regular used lattice (mesh) structure. Therefore any complex shape can be modeled ([18]), as long as it fits into a deformable triangulated model.

Also Laplacian approaches for deformation exist. Sorkine is one of the authors, who propose combinations of Laplace and free-form modeling (such as in [31] and [40]). Also the Laplace approach, as for the Dirichlet above, is using a set of control vertices.

The Laplacian Matrix, where many methods (not only free-form; think for example at the already introduced FEM) make use of, could be constructed as follows (based on [40]):

- Let us take a triangular mesh T, with vertices, edges and faces.
- The adjacency (connectivity) matrix A, where each element $A_{i,j}$ becomes:

  If vertex i *and neighbor vertex* j *are connected by an edge;*

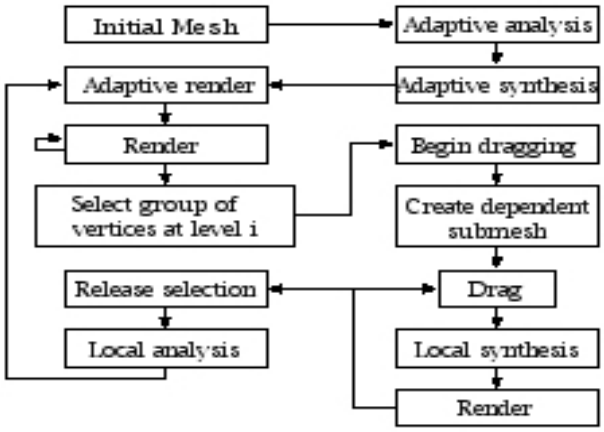


Figure 11: Global outline between the various procedures in the editing process, taken from [54].

if $(i, j) \in \mathtt{Edges}$, then $A_{i,j} = 1$.
If i *and* j *are not connected by an edge;*
Otherwise, then $A_{i,j} = 0$.

Where `Edges` *is of course the collection of all edges.*

- The diagonal matrix D with each diagonal element: $D_{i,j} = d_i$, where $d_i = \mid N(i) \mid$ is the degree of vertex i and $N(i) = \{j \mid (i, j) \in \mathtt{Edges}\}$ the collection of directly connected neighbors of vertex i.

Now the Laplacian Matrix becomes: $L = I - D^{-1}A$, where I is the identity matrix and D^{-1} denotes the inverse matrix of D. Sorkine et al. states also that the symmetric version of this matrix is likely to be more convenient in use:

$$L_{sym} = DL = D(I - D^{-1}A) = DI - DD^{-1}A = D - IA = D - A$$

The Laplacian matrix transforms the global coordinates to differential coordinates (representing the local detail and local shape description), which can than be used for further processing of the mesh. However, this new symmetric matrix is singular (its determinant is zero!), which means that when the coordinates are ready for transforming back another transform method needs to be devised.

7.2.3 *Multi-resolution*

The authors of [54] propose a multi-resolution approach for meshes, which is based on subdivisions. They explain it as a natural extension of free-form patch surface schemes (as described by [2]). This proposed method can be influenced by the user with a set of control vertices, after editing (altering vertices) also a smoothing function like Gaussian or Taubin needs to be used.

The global outline of the editing process is given by the diagram in figure 11. This process starts a cycle, when "adaptive rendering" is entered. Before the actual deformation process starts, global synthesis and analysis is done on the mesh. During the cycle only local (on the so-called sub-mesh (small part of mesh)) changes take place (synthesis and render) and when "adaptive rendering" is entered again, the

'whole' (depending on the available resources!) mesh, including the local changes, is redrawn. In [54] ideas for the implementation are presented, but are not added here to prevent introducing a number of pages which might not be very interesting for the current research.

Further details about the theory such as the used multi-resolution transform (used e.g. for smoothing) can be found in [54]. This transform is related to the well known wavelet transform, more information about both transforms can also be found in books such as [16] (although this book is written with image processing in mind, the general theory is the same).

7.2.4 *Comparison of free-form methods*

To wrap up the free-form topic, a couple of conclusions could be drawn.

Free-form methods can be controlled by either:

- A regular mesh.
- A set of vertices.

The advantages of the shown variations are:

- General: the "ease" for the user to deform a meshed object.
- Laplacian or Dirichlet approach possible, which use control vertices at arbitrary positions.
- A different and interesting approach is the multi-resolution representation for meshes.

The main disadvantage for our problem is that there are currently no ideas presented for incorporating image data. If the decision is taken to implement it in Matlab, then considering how to incorporate this into a Free-form method is important. Also it would be good to look into the possibilities for different behavior, such that the vessel deformation and the bifurcation deformation behave differently.

7.3 ADAPTIVE MODEL

Let us now consider the last type of methods. These deform a mesh or triangulation by adaptation and possibly refining the adaptive result. Several proposals about adaptive meshes are present in the research community. Here the ideas of e.g. [6], [46] and [12] will be discussed.

Adaptive hints at the use of adaptive model parameters, as for example the mass of the model vertices.

The authors of [6] propose an adaptive method for a self-optimizing mesh. Their idea is to develop a method that increases computational efficiency and accuracy for 3D deformable models. They start with an energy equation, which incorporates internal energy, energy of the external forces and energy of external constraints (such as preventing that a vertex moves away from the desired image data location).

After adapting the model, the processes of refinement and decimation are a possible enrichment for the deformed model. Due to these processes, sufficient details in the mesh are kept (perhaps adding vertices and removing redundant vertices, if they are close together). Refinement and decimation are usually done iteratively, as is shown in [6] and [12]. Two possible decimation methods, which could be used, are compared in [33].

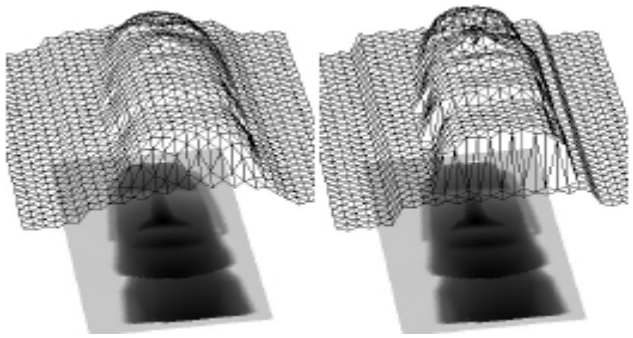


Figure 12: Left: example of an adaptive triangulation taken from [46]. Right: discontinuity included (look at the sharper edges).

In [46] the adaptive triangulation is represented by a discrete dynamical system, which is constructed from a set of movable vertices and e.g. adjustable springs. An example of a spring adjusted system is also shown in [37].

Following the information of [46], a 3D triangulation could be adapted and controlled by the set of motion equations:

$$m_i \frac{d^2 p(t)_i}{dt^2} + \gamma_i \frac{dp(t)_i}{dt} + \sum_{j \in N_i} \frac{c_{ij} e(t)_{ij}}{\|r(t)_{ij}\|} r(t)_{ij} = f(t)_i$$

Where:

- $i = 1, \ldots, N$ and N is the total number of vertices in the triangulation.
- $j = 1, \ldots, N_i$ and N_i is the number of neighboring vertices of center vertex i.
- $p(t)_i$; center vertex i in the current motion equation.
- $p(t)_j$; neighboring vertex j of i, in current motion equation.
- m_i; the mass of the time evolving 3D-vertex $p(t)_i = [x(t)_i, y(t)_i, z(t)_i]$.
- $\frac{d^2 p(t)_i}{dt^2}$; the acceleration of vertex i.
- γ_i; damping coefficient of vertex i.
- $\frac{dp(t)_i}{dt}$; the velocity of vertex i.
- c_{ij}; controls the stiffness of the spring.
- l_{ij}; the natural (or rest) length between vertices i and j.
- $r(t)_{ij} = p(t)_j - p(t)_i$; the difference between the positions of vertices i and j.
- $e(t)_{ij} = \| r(t)_{ij} \| - l_{ij}$; the actual length between vertices i and j.
- $f(t)_i$; the external force.

With this adaptive triangulation reconstruction, the authors of [46] created the 3D results of the image data (which is shown underneath the triangulation) in figure 12. Also they introduced a notion of discontinuity for this method, such that the 3D triangulation on the right image is sharper bounded as it is on the left image.

7.4 CONCLUSION

Let us conclude this chapter with the observation that GA and GAGR are using multiple meshes, which is simply too computational intensive for the purpose of the wanted interactive method. In contrast Free-Form methods usually use two meshes, but the image data might be a problem.

The adaptive model can include image data, but might get slow with all the matrix motion equations. However, [7] and [8] describe that there are performance improvements possible, for these usual matrix based models. This is especially interesting for our wanted interactive implementation.

8 CONCLUSION

Several flavors of methods for the wanted interactive method were discussed in this part, with the requirements of page 2 in mind. After careful research and thinking, the resulting comparison table is shown below for a quick overview. This table compares the methods with three key requirements and states if the method is considered as possible solution for solving our problem.

Most of the discussed methods can be discarded, as not suitable (for different reasons), for the wanted 3D interactive triangulation and mesh deformation method. Although most of the methods are labeled as global and local deformation, some caution is needed. FEM has for instance a number of problems around local area deformations to tackle.

Let us consider the following possible methods:

1. Adaptive Model.
2. Boundary Element Method.
3. Free-form.

The boundary element method and the free-form method are interesting for this thesis. However, both have concerns around performance, also it would be good to investigate how the deformation differences in behavior of bifurcations and vessels can be included. Beside the performance issues of the Free-form method, a more serious problem arises for incorporating image data. Currently I am not aware of theories or implementations for Free-form methods that integrate image data, which might be the reason to discard it.

Currently the Adaptive Model is the only method that satisfies each of the wanted requirements, also it was previously used for other human organs (such as in [8]) and the own developed Matlab prototype yielded positive results. Therefore only an adaptive method will be developed in the remainder of this thesis.

Requirement / Method	Hard	Soft	Region of correction	Implement
Av. Change M.	+	-	Global & Local	X
Finite E. M.	+	?	Global & Local	X
Boundary E. M.	+	+	Global & Local	Considered
Natural E. M.	+	-	Global	X
Genetic A. GR.	-	+	Global	X
Free-form	+	?	Global & Local	Considered
Adaptive M.	+	+	Global & Local	Considered

Table 1: Comparison of discussed methods.

Part II

THEORY

Everything is determined,
the beginning as well as the end,
by forces over which we have no control.
It is determined for insects as well as for the stars.
Human beings, vegetables or cosmic dust,
we all dance to a mysterious tune,
intoned in the distance.

— Albert Einstein

9 INTRODUCTION

An adaptive model is general based upon internal equations of:

- Energy
- Force
- Motion

External data, such as image data, could guide the deformation, also constraints, such as the initial moved "center" vertex, are important for the deformation as well. The image data should not be included for the "hard" correction on the triangulation, but is included for our "soft" correction on the triangulation. Hence, it is a nice idea that the same method can serve both corrections.

This subject is often based on so-called mass-spring models or variations of it, which is also used by Choi et al. (in [7] and [8]) for simulating interactively deforming soft tissue. They use an adaptive tissue model and the external pressure is propagating through the model with decreasing force, such as in a human liver for local deformation with a probe ([7]).

The authors of [6] propose an adaptive model for a self-optimizing mesh. Their idea is to develop a method that increases computational efficiency and accuracy for 3D deformable models. They start with an energy equation, which incorporates internal energy, energy of the external forces and energy of external constraints.

The literature research mentioned the adaptive model equation of [46] as follows:

$$m_i \frac{d^2 p(t)_i}{dt^2} + \gamma_i \frac{dp(t)_i}{dt} + \sum_{j \in N_i} \frac{c_{ij} e(t)_{ij}}{\|r(t)_{ij}\|} r(t)_{ij} = f(t)_i \tag{9.1}$$

This equation includes all neighbors j of the "center" vertex i. Our final method will contain a more specific approach, which only includes the (current) "center" vertex i and one neighbor j. Let us use therefore the above equation as a starting point towards our final one.

To reduce the neighbors down to one neighbor j, we remove the sum of all neighbors of (the current) "center" vertex i:

$$m_i \frac{d^2 p(t)_i}{dt^2} + \gamma_i \frac{dp(t)_i}{dt} + \frac{c_{ij} e(t)_{ij}}{\|r(t)_{ij}\|} r(t)_{ij} = f(t)_i$$

Let us replace $e(t)_{ij}$ and $r(t)_{ij}$, with the information of [46],

$$m_i \frac{d^2 p(t)_i}{dt^2} + \gamma_i \frac{dp(t)_i}{dt} + \frac{c_{ij}(\|p(t)_j - p(t)_i\| - l_{ij})}{\|p(t)_j - p(t)_i\|}(p(t)_j - p(t)_i) = f(t)_i \tag{10.1}$$

Where $l_{ij} = \|p(0)_j - p(0)_i\|$ is the natural or rest length of the spring. Thus at the overall time $t = 0$, the spring, and thus the triangulation, is in the rest position!

10.1 FIXING THE "CENTER" VERTEX

Let us assume now that the "center" vertex i will be moved, by the user, at the start of the deformation and is locked on that position during the further overall deformation process. This means that the equation would yield, for the acceleration and damping part, both a result equal to zero. However, there are one or more vertices that actually will move in response to the displacement of vertex i. In the current deformation part this will be neighbor vertex j. With this new knowledge, the equation is transformed in:

$$m_j \frac{d^2 p(t)_j}{dt^2} + \gamma_j \frac{dp(t)_j}{dt} + \frac{c_{ij}(\|p(t)_i - p(t)_j\| - l_{ij})}{\|p(t)_i - p(t)_j\|}(p(t)_i - p(t)_j) = f(t)_j$$

Due to the locking of vertex i, the position of i will be the same throughout the processing time t of neighbor j. Therefore we can always use $p(0)_i$ in the deformation processes of a neighbor j, which renews the equation to:

$$m_j \frac{d^2 p(t)_j}{dt^2} + \gamma_j \frac{dp(t)_j}{dt} + \frac{c_{ij}(\|p(0)_i - p(t)_j\| - l_{ij})}{\|p(0)_i - p(t)_j\|}(p(0)_i - p(t)_j) = f(t)_j \tag{10.2}$$

10.2 VELOCITY AND ACCELERATION

The velocity and the acceleration parts of the equation could be rewritten, such that

$$m_j a(t)_j + \gamma_j v(t)_j + \frac{c_{ij}(\|p(0)_i - p(t)_j\| - l_{ij})}{\|p(0)_i - p(t)_j\|}(p(0)_i - p(t)_j) = f(t)_j$$

Where $a(t)_j$ and $v(t)_j$ are:

$$a(t)_j = \frac{d^2 p(t)_j}{dt^2} = \lim_{\Delta t \to 0} \frac{v(t+\Delta t)_j - v(t)_j}{\Delta t} \approx \frac{v(t+\Delta t)_j - v(t)_j}{\Delta t}$$

$$v(t)_j = \frac{dp(t)_j}{dt} = \lim_{\Delta t \to 0} \frac{p(t+\Delta t)_j - p(t)_j}{\Delta t} \approx \frac{p(t+\Delta t)_j - p(t)_j}{\Delta t}$$

If now these two parts are placed into the equation, we get the approximation:

$$m_j \frac{v(t+\Delta t)_j - v(t)_j}{\Delta t} + \gamma_j \frac{p(t+\Delta t)_j - p(t)_j}{\Delta t} +$$
$$\frac{c_{ij}(\|p(0)_i - p(t)_j\| - l_{ij})}{\|p(0)_i - p(t)_j\|}(p(0)_i - p(t)_j) = f(t)_j$$

After rewriting the acceleration part, the final tissue deformation model equation results in:

$$m_j \frac{p(t+2\Delta t)_j - 2p(t+\Delta t)_j + p(t)_j}{(\Delta t)^2} + \gamma_j \frac{p(t+\Delta t)_j - p(t)_j}{\Delta t} +$$

$$\frac{c_{ij}(\|p(0)_i - p(t)_j\| - l_{ij})}{\|p(0)_i - p(t)_j\|}(p(0)_i - p(t)_j) = f(t)_j \tag{10.3}$$

11 CONCLUSION

This theory part has introduced the desired tissue deformation model equation and it is derived from the more general adaptive equation 9.1 of [46]. Furthermore we need to notice that the time t starts counting for each deformation of a neighbor j individually.

The image data and other issues concerning the tissue deformation model, are not introduced further in this part. More detail information will be given in the next part.

Part III

IMPLEMENTATION

Any intelligent fool
can make things bigger,
more complex, and more violent.
It takes a touch of genius,
and a lot of courage,
to move in the
opposite direction.

— Albert Einstein

12 INTRODUCTION

During the previous parts, it has emerged that the solution for the wanted 3D interactive deformation method is an adaptive model. In the theory part, the following tissue deformation model equation is derived:

$$m_j \frac{p(t+2\Delta t)_j - 2p(t+\Delta t)_j + p(t)_j}{(\Delta t)^2} + \gamma_j \frac{p(t+\Delta t)_j - p(t)_j}{\Delta t} +$$

$$\frac{c_{ij}(\|p(0)_i - p(t)_j\| - l_{ij})}{\|p(0)_i - p(t)_j\|}(p(0)_i - p(t)_j) = f(t)_j \tag{12.1}$$

Although this is the key to our solution, some additional work needs to be done before it is really usable in practice. Let us therefore first consider the implementation framework. Furthermore it would be good to look in more detail to some key ingredients of the deformation, such as image data inclusion and how the parameters should be used.

13 FRAMEWORK

From the literature research part it is known that the adaptive model methods generally solve their equations in a matrix, which slows the deformation process down considerably. Due to the wanted "local" deformation area, usually not the whole triangulation is deformed and therefore there is no need for a complete matrix update.

The global outline of our developed solution is therefore:

1. Move the given "center" vertex to the wanted place.
2. Determine to which direct neighbors this center vertex is connected.
3. Move the first encountered neighbor according our equation and use, if available, the guidance of the corresponding image data.
4. Repeat step 3 with the next neighbor, until no neighbors are left.
5. Select new neighbors of the used neighbors, such that the next "ring" away from the center can be deformed.
6. Go back to step 3, with these new neighbors until the given/needed area (number of rings) is deformed.
7. The model is deformed.

Sometimes the given area is bigger, then the actual deformed area.

With this global and simple idea in our minds, it is time to implement the framework. Also it should be kept as simple as possible, such as Albert Einsteins quote at the beginning of this part already suggested.

13.1 INPUT AND OUTPUT

This implementation will deform a triangulation and not directly a mesh. Although the mesh could be indirectly deformed, by converting the deformed triangulation into a mesh. To make clear what is needed as input, before performing a deformation and to output a deformed triangulation, is shown by figure 1 on page 1.

Remember that in the literature research a transformation between a triangulation and mesh was suggested.

In figure 1 the inputs have different places. At the front side of the deformation framework box the following inputs are stated:

- **Original triangulation** is the triangulation that will be deformed.
- **Vertex to move and displacement vector** are the wanted center vertex to move and a vector containing the wanted displacement of that vertex, they are crucial for the deformation framework.

The "natural" input parameters at the top of the deformation framework box are:

- **Mass** is the mass of the vertices (see chapter 14).
- **Elastic** is the elasticity of the vertices (see chapter 14).
- **Viscous** is the viscosity of the vertices (see chapter 14).

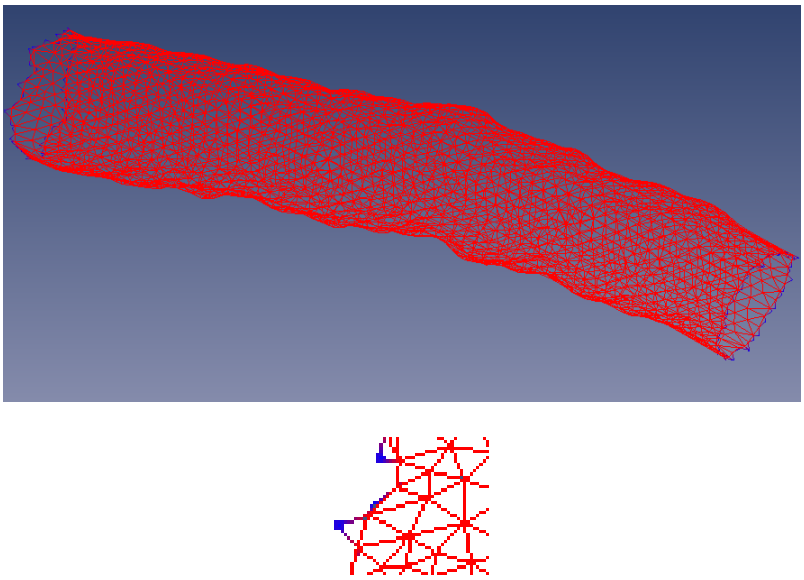


Figure 13: On top the complete triangulation of a vessel is shown. Below a part of the left end of the vessel is shown, to illustrate how a part of a triangulation could look like.

- **Level Of Influence (LOI)** is used to determine the maximum area of deformation (or actually the number of rings away from the "center" vertex). If LOI is bigger than the actual needed deformation area, it will stop earlier to save valuable processing time.
- **Fixed/Dynamic** more information is found in section 13.3.
- **Image data** is the gradient MRI image data from the concerned patient.
- **Cube size** is the size of the direction computation cube (see section 13.4).

The inputs and the obligatory deformed triangulation output are known, now the framework can be filled in further.

13.2 DETERMINE NEIGHBORS

Let us consider the 3D triangulation of a human vessel, as is shown in figure 13. The enlarged part of this triangulation, shows quite clearly that each vertex in a 3D triangulation has at least two neighbors (look at the two blue thickened end vertices). And these neighbors have again neighbors and so on. To determine these neighbors could be a bit of a problem sometimes, but this is totally depending on the implementation of the triangulation. Here some thinking is important, before starting to implement.

In the current method a triangulation structure is used, which interconnects the given vertices, edges and triangles. Therefore it is easy to find the neighbors; If the initial vertex to move is known, then we simply follow the connected edges to the neighbors. All these neighbors are than deformed by the method and the next ring of neighbors after that and so on. Also it would be good to keep track of the already

moved vertices, such that each vertex is only moved once. This can be maintained by the method during the deformation process.

13.3 FIXED/DYNAMIC

The previous mentioned Fixed/Dynamic parameter takes an important place during the actual moving process of each vertex. This parameter is used to determine which kind of displacement (fixed or dynamic) is wanted for $p(t+2\Delta t)_j$ and $p(t+\Delta t)_j$. The resulting vertex displacement is updated during the moving process of each vertex.

The fixed version is used to compute a new vertex for the considered neighbor vertex as follows:

$$p_{new} = p_{current} + (factor * displacement) \tag{13.1}$$

Notice that the $*$ is an element-wise multiplication!

Where:

- p_{new} is the new 3D-position of the considered neighbor vertex.
- $p_{current}$ is the current 3D-position of the considered neighbor vertex.
- $factor$ is a 3D-vector which includes e.g. the normalized displacement direction.
- $displacement$ is the initial 3D-displacement vector given by the user.

Notice that the initial displacement vector elements actually should be treated as absolute values, because the factor defines the direction and this vector only influences the size of that direction.

This results in:

$$p(t+\Delta t)_j = p(t)_j + (factor * displacement) \tag{13.2}$$

$$p(t+2\Delta t)_j = p(t+\Delta t)_j + (factor * displacement) \tag{13.3}$$

Before investigating what the $factor$ vector actually contains, we should have a look at the dynamic version:

$$p_{new} = p_{current} + (factor * (p_{"center"} - p_{current})) \tag{13.4}$$

The only difference is the exchange of the displacement part, from the fixed version, by the more dynamic displacement, which results in:

$$p(t+\Delta t)_j = p(t)_j + (factor * (p(0)_i - p(t)_j) \tag{13.5}$$

$$p(t+2\Delta t)_j = p(t+\Delta t)_j + (factor * (p(0)_i - p(t+\Delta t)_j) \tag{13.6}$$

13.4 IMAGE DATA

During prototyping in Matlab, also image data has been used to guide the process closer to the desired tissue behavior. This has been done to satisfy the requirement for the “soft” correction of the segmentation. Actually in Matlab the factor is implemented as:

$$\text{factor} = \frac{0.4 * \text{Normalized\_Direction} + 0.6 * \text{External\_Direction}}{100} =$$

$$0.004 * \text{Normalized\_Direction} + 0.006 * \text{External\_Direction} \quad (13.7)$$

Where:

- Normalized_Direction is a normalization of the displacement direction 3D-vector between the neighbor and the "center" vertex, such that we get for each vector-element a value in the range of -1 to $+1$.
- External_Direction is a normalization of the gradient image direction, which will follow shortly.

Remember that this "factor" stays a 3D-vector, such that the previous formulae should multiply element-wise. The normalized direction vector is simply stating in which direction the vertex needs to move for the deformation process. It is also updated during the moving process, to see if the direction is still right or if it needs to be changed.

The normalized external direction vector is computed within a small direction computation cube around the center vertex, from the 3D gradient image data, in the following way:

1. Check if there is gradient image data available. If not available, return a 3D-vector with zero’s, to state that the "central" position in the “cube” needs to be used.
2. Check if the direction computation cube size is entered.
3. Create the direction computation cube (such as on the left in figure 14) around the current vertex, such that the vertex is the center of the cube (or at least contained in the blue center image voxel).
4. Locate the highest gradient image value in the cube, because these are close to an image boundary.
5. Compute the direction vector (such as on the right in figure 14) to this gradient value from the center of the cube.
6. Now the direction vector is normalized and presented as the external direction vector, which contains again element values in the range of -1 to $+1$.

Notice that the cube size should always be odd and bigger than 1! The current vertex could otherwise not be placed in the center voxel of the cube.

A voxel is a volumetric pixel, usually depicted as a cube.

Multiplying element-wise these two normalized direction vectors with their own different fraction, gives the image data 60% chance to pull the vertex to the "ideal image" position and 40% chance to pull it in the direction of the "ideal move" position. To prevent that the new neighbor position is moved quickly in a certain direction, we introduced a division by 100 for a slower progression towards the final position.

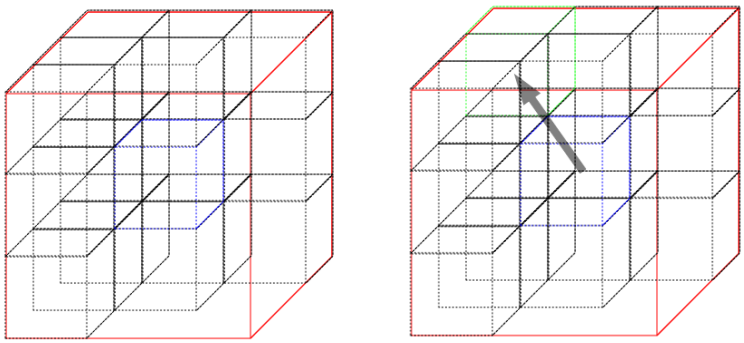


Figure 14: Left: sketch of a direction computation cube with selected center voxel (blue) and cube size 3. Right: sketch of the same direction computation cube, but now with a direction vector from the blue voxel center to the green voxel center.

14 PARAMETERS

From theory we know that parameters such as mass, viscosity and elasticity are important parameters for the developed equation. Let us therefore start here to explain these three in more detail.

14.1 MASS, ELASTICITY AND VISCOSITY

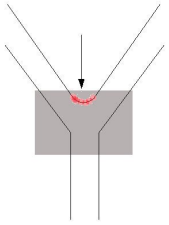
Using different ranges in values for these three parameters might become a problem if a value is so huge, that it influences the deformation process considerably. It would be better to take similar ranges to prevent this from happening. In general the value range will than be normalized.

Suppose that we assume a normalization from 0 until 1 for the three parameters. This will be fine for the mass and elasticity parameter, because we have in general no negative mass or elasticity values. However, for the viscosity range this is sometimes a different story.

Let us consider, to illustrate the choice of the viscosity range, a mass of 0.1, elasticity values of 0.5, 0.9 and 0.99, using the dynamic version of the method and the same initial displacement. In figure 15 three plots are shown with the corresponding elasticity values and horizontally a viscosity range between -1 and $+1$. The plots describe the different behavior for the negative and positive viscosity range against the total number of deformed x-, y- and z-coordinates (thus the actual impact of the deformation) of a vessel-triangulation.

From theory we know that damping, or in our case the viscosity, decreases the flexibility of a mass-spring model (changing the traveling speed of a vertex), when the value goes away from zero (no damping). This behavior is sometimes illustrated by taking a negative value range, which delivers (see also figure 15) a smooth curve with a decreasing number of deformed x-, y- and z-coordinates. Sadly, the negative range, does not give the expected rigidness to the concerned neighboring vertices at a viscosity of -1. Actually we only would expect a few vertices to move with a viscosity of $(-)1$, which is due to the fully available damping. Therefore we decided that also our value range should become positive, thus ranging from 0 until 1.

14.2 GLOBAL AND INDIVIDUAL PARAMETERS

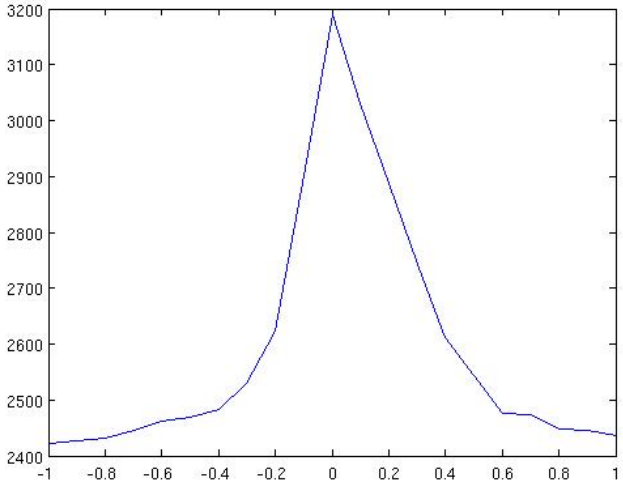


A bifurcation (approximately the shaded area) between vessels and with an arrow at the (red) Carina.

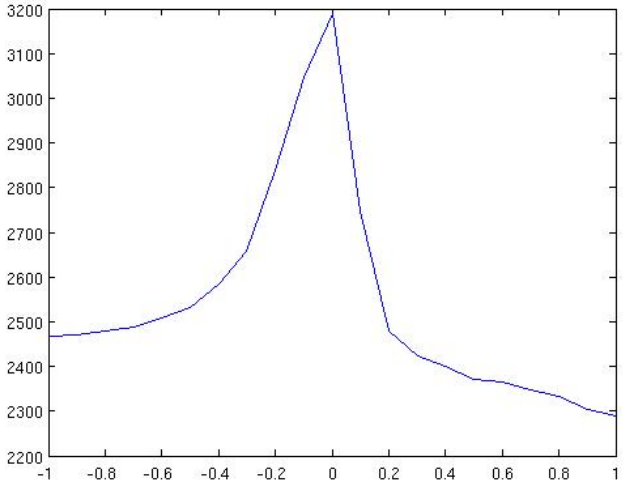
During the theory part it is shown that each vertex has its own mass, viscosity and elasticity parameter. However, to simplify the whole process in the Matlab prototype, global values for these parameters where used. This global parameter setup is also available in the C/C++ implementation, but with certain models problems may arise. Let us look to the given vessel and vascular lumen tree in figure 16.

When the segmented vessel is deformed, it would be appreciated if all the vertices have the same values for the three parameters. The deformation behavior of a vessel is therefore relatively the same over the entire vessel, but is this idea also valid for a tree and especially a bifurcation?

Elasticity = 0.5



Elasticity = 0.9



Elasticity = 0.99

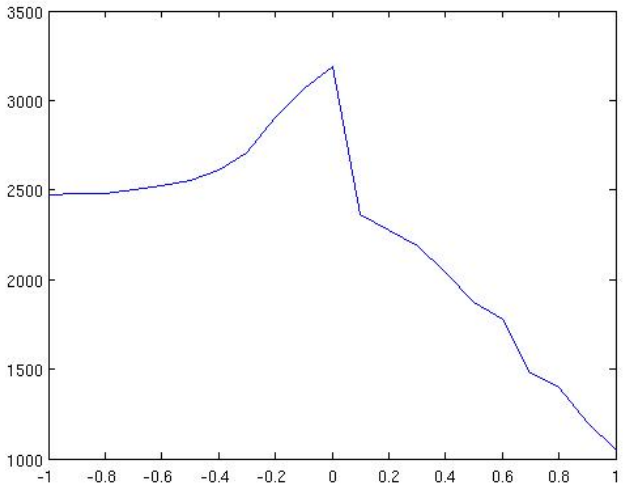


Figure 15: Viscosity (horizontal) versus the number of deformed x-, y- and z-coordinates (vertical).

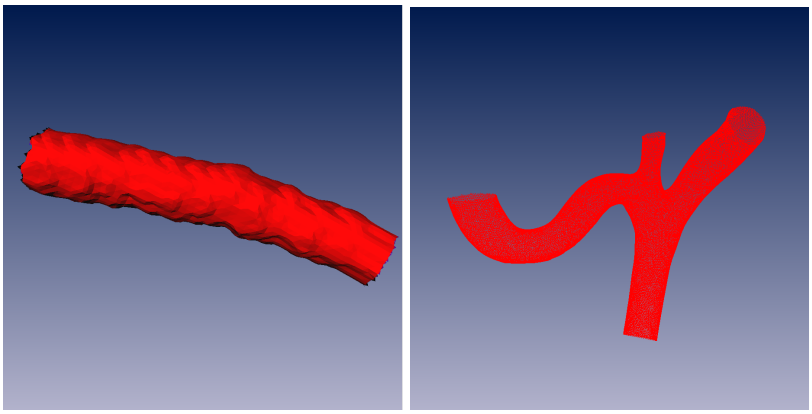


Figure 16: Left: a relatively smooth segmented vessel (triangles are shaded, to see the model of the vessel clearer). Right: a very smooth segmented tree (triangulation view).

The answer is yes and no. The tree contains also sections, which can be treated as a separate vessel. In these parts a global parameter setting can be done, but the deformation should stop before it comes close to a bifurcation. The Carina of a bifurcation is usually less flexible, than the surrounding tissue. On this kind of places it would be very nice to fine-tune the parameters of the individual vertex. Or if one branch of a bifurcation should not be included and a smaller level of influence (LOI) is not possible, we could change there the mass, elasticity and viscosity values to 0 for an optimal rigid behavior.

15 CONCLUSION

From this part we could conclude that the tissue deformation model equation is embedded in a framework and that the parameters should be in the range:

- 0 to 1 for mass.
- 0 to 1 for elasticity.
- 0 to 1 for viscosity.

Also we should notice that this method will yield different deformation results for several resolutions (number of vertices) of the same vessel. Although it is quite obvious, because the method is based on an equation of two vertices at each time, some people might overlook this and are lost how to interpret these differences in behavior.

Part IV

MEDICAL MOTIVATION

He would like to have active muscles,
but knows that electric motors do not
have the correct characteristics.

— R. A. Brooks

16 INTRODUCTION

From the previous parts, it is known that our method needs certain parameters. For the parameters mass, elasticity and viscosity, a specific range or value should be found. It is already known to us, that the normalized versions of these values should be in the range of:

- 0 to 1 for mass.
- 0 to 1 for elasticity.
- 0 to 1 for viscosity.

However, it would be very interesting, if it can be narrowed down. This would help the user of the method considerably. Let us therefore search for suggestions in e.g. medical related papers, such that we can find characteristics for the three parameters.

17 MOTIVATIONS FROM THE MEDICAL DOMAIN

The best way to narrow down the range of the normalized values, is to make realistic decisions about them. These decisions are taken with medical research in mind. Let us first consider the elasticity parameter.

17.1 ELASTICITY AND MASS

The elasticity of a vascular vessel wall is mainly due to the elastin and collagen fibers. Collagen fibers are playing also an important part in the viscosity of the vascular wall, which is due to their energy dissipation. Therefore it can be concluded that these fibers not only store fiber elastic energy, but also have energy dissipation by fiber slippage.

Generally it is known that diamonds are not elastic, but that nylon and human blood vessels are quite the opposite. To capture these differences in one parameter, we use the modulus of elasticity from table 2.

If we assume that for our purpose the natural diamond is yielding the highest modulus of elasticity and thus has the least flexibility in our method, then, after normalization, the diamond stiffness results for our purpose in:

$$c_{ij} = 1 - \frac{1200}{1200} = 0$$

The arterial segments do now get results between:

$$c_{ij} = 1 - \frac{0.004}{1200} = 1 - 0.000003333 = 0.999996667$$

and

$$c_{ij} = 1 - \frac{0.0003}{1200} = 1 - \frac{0.0003}{1200} = 1 - 0.00000025 = 0.99999975$$

Parameters / Material	GPa	$x10^6$ dynes/cm^2
Carotid artery	0.0003 - 0.0008	3.0 - 8.0
Thoracic aorta	0.0004 - 0.001	4.0 - 10
Abdominal aorta	0.0004 - 0.0015	4.0 - 15
Iliac artery	0.0008 - 0.004	8.0 - 40
Femoral artery	0.0012 - 0.004	12 - 40
Nylon blend	0.9 - 3.5	-
96% silica glass	68	-
Natural diamond	700 - 1200	-

Table 2: Comparing different materials with the general accepted modulus of elasticity (in GPa, taken from [26]) and including, for the complete picture, the circumferential incremental elastic modulus (in $x10^6$ dynes/cm^2, taken from [5]) for the arterial segments.

However, not all vascular vessels of a human are included and neither the age of the vessels is taken into account. Although we will be able to control e.g. the aging by the "viscosity", it would be good to place our elasticity not to close to 1 thus lowering the approximations to 0.99.

To find information about viscosity (such as in [14] for several human vessels) in other objects is actually quite difficult, usually it is computed of liquids and rarely from more or less solid materials. Let us therefore postpone the viscosity range narrowing to the evaluation part, where maybe testing can help us determine it better.

A narrowing of the mass parameter range can be done, with knowledge received during, for example, physics or biology courses. It is generally accepted that vascular tissue is closer to the weight of a feather, than to the weight of a stone, car or building. Therefore, it makes sense to reduce the mass to approximately 0.1.

17.2 CONSIDERATIONS

Although we already have approximated values for the mass and elasticity, it would be good to take a closer look into a few real medical issues. Let us therefore consider the:

- **Types;** in the human body there are several types of vessels, such as the thoracic aorta and carotid artery. These types behave all differently ([14]) and also their bifurcations (if applicable) will behave different as well. From biology books, such as [35], it can be extracted that a vein carries blood from the capillaries towards the heart and an artery carries blood from the heart to the body. This last type is usually more elastic than the veins, because the arteries need to smooth out the pulsatility of the heart along the body. Also the pressure of the blood on the vascular wall will be higher in the arteries than in the veins.
- **Sizes;** in [14], size is related to the values of the elastic index and viscous index. The brachial artery has a smaller mean diameter than the femoral artery. This results for the brachial artery (with respect to the femoral artery) in a lower elastic index and also in a lower viscous index, but for other types it could be different (see [14] for more details).
- **Ages;** also it is important to consider the age of a person. Usually there is a relation between the age and stiffness of the vascular wall. During the life of a human, the body starts to degrade after a certain age. This is also true for the vessels, which are used continuously. Through this continuously process, the vessels are getting stiffer over time.
- **Diseases;** in [36] and [35], also several diseases are presented. Think for example at diseases as arteriosclerosis (hardening of the arteries) or acute pulmonary hypertension (sudden increase of the blood pressure). These diseases will change the actual characteristics of the viscosity and elasticity of the vascular wall.

Let us notice here that the normalized viscosity value range could be used for the differences in flexibility of the considered vessel(-tree) in the evaluation part, such that we can consider e.g. different diseases, age-groups and types of vessels.

18 CONCLUSION

In this part we discussed a medical related approach to narrow down our normalized parameter ranges. For our mass and elasticity parameter we approximated the values of:

- Elasticity to 0.99.
- Mass to 0.1.

For the viscosity we concluded that the normalized value range could be used in the evaluation part, such that we can simulate the deformation of the tissue closer to the related age or e.g. disease of the human involved. However, usually we do not know the person and/or disease behind the received vessel(-tree) triangulation (or MRI-data), therefore it might be better to collect an approximated value from our evaluation part. The simulated results with the approximated viscosity value should be reviewed by e.g. Pie Medical Imaging and if it is considered as satisfactory, we can conclude that this approximation is acceptable for our method.

Part V

EVALUATION

The only source of knowledge is experience.

— Albert Einstein

19 INTRODUCTION

In the previous parts:

- We adopted the adaptive model idea.
- Derived our specific deformation approach in theory.
- Created a Matlab prototype and a C/C++ implementation from the theory.
- Image data could be included into the framework of our method, but is not obligatory.
- Concluded that we could use the "natural" parameters $mass = 0.1$ and $elasticity = 0.99$.
- Left the normalized range $(0 \dots 1)$ from the viscosity open, such that we are able to show e.g. different age-groups of patients and diseases.
- And also the level of influence (LOI) could be set by our method.

From the above we already could conclude that the requirements from section 1.2 on page 2 are met. However, what are the results of our Matlab prototype and C/C++ implementation?

20 MATLAB PROTOTYPE

During the prototype results we will use two different vessels as illustrated in figures 17 and 21, also we will use a bifurcation as in figure 24. First we will discuss a deformation, where the center vertex displacement is not determined by the image data and secondly a center vertex displacement which is determined by the image data. We conclude the prototype results with bifurcation deformations.

20.1 VESSEL RESULTS

The center vertex vessel of figure 17 (as will be indicated in the top image of figure 27 on page 64) is moved outwards, such that we can try to fill the local dent there.

If we would use a quite extreme displacement vector, then:

1. In figure 18 the top image shows a deformation with a viscosity of 0, a LOI of 10 and a displacement vector $0, 0, -15$. Here it can be seen quite clearly that the LOI of 10 is not enough, because of the discontinuities (the green straight lines) to the remainder of the triangulation.
2. The bottom image in figure 18 has a LOI of 100, but uses further the same parameters as the top image. Here it can be seen that the LOI is large enough to accommodate a smooth deformation.
3. In figure 19 the images of a deformation with a viscosity of 0, a LOI of 100 and a displacement vector $0, 0, -5$ are shown. Here we can see that the deformation keeps filling the dent, but using a less extreme deformation which results in a smoother triangulation than the ones in figure 18.

The above deformations are done without image data guidance and using the fixed displacement option, they are created in the approximated times:

1. 94 milliseconds (is stopped earlier by the small LOI than 2)
2. 1211 milliseconds
3. 1199 milliseconds (is stopped earlier than 2. This is due to a stopping criterion that signals an earlier (before LOI is reached!) stable state of the triangulation, because the displacement was less extreme.)

We might conclude so far that a bigger LOI is better for a smoother deformation, although it will consume more time!

Let us now consider a deformation, which is closely related to the natural length of the used triangulation. The displacement vector $0, 0, -1$ is such an example of deformation with the size of the natural length. With this vector the deformations became closer to the original triangulation and therefore less time was needed.

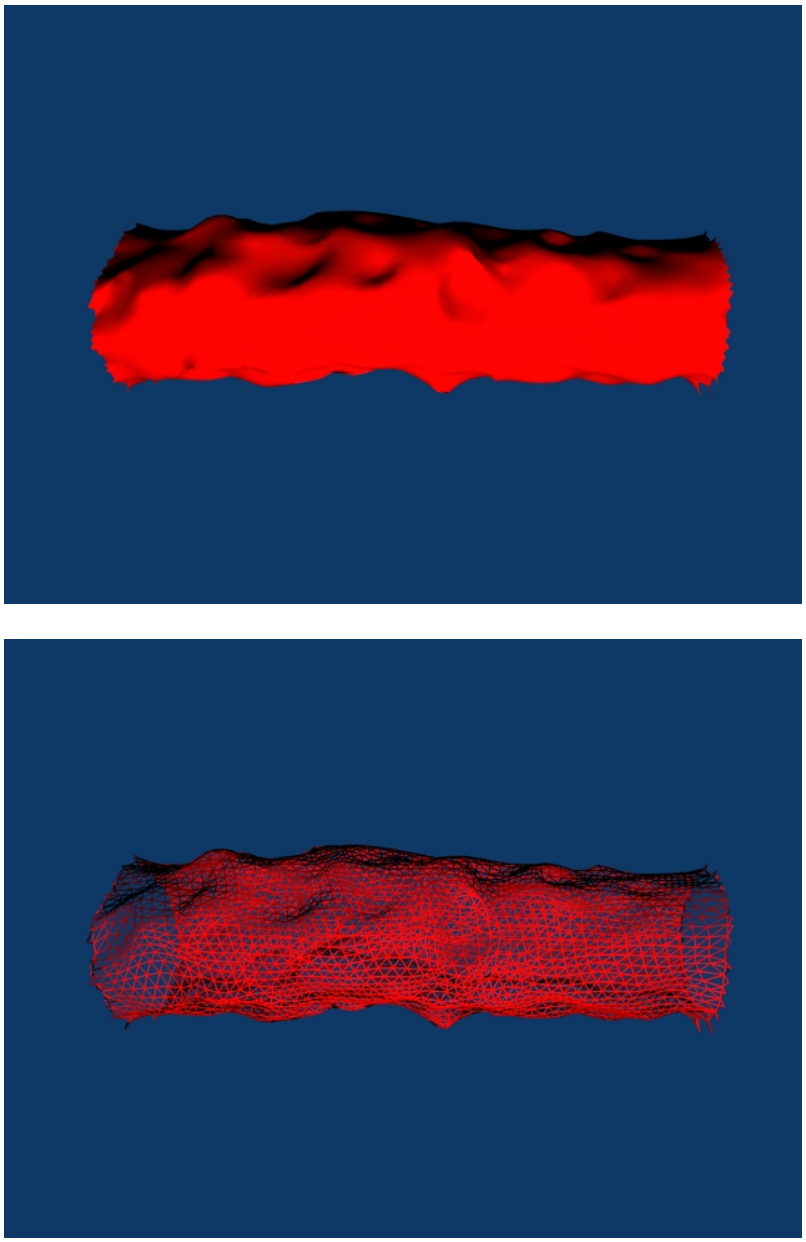


Figure 17: Top: original shaded vessel triangulation with 3412 vertices. Bottom: original vessel triangulation.

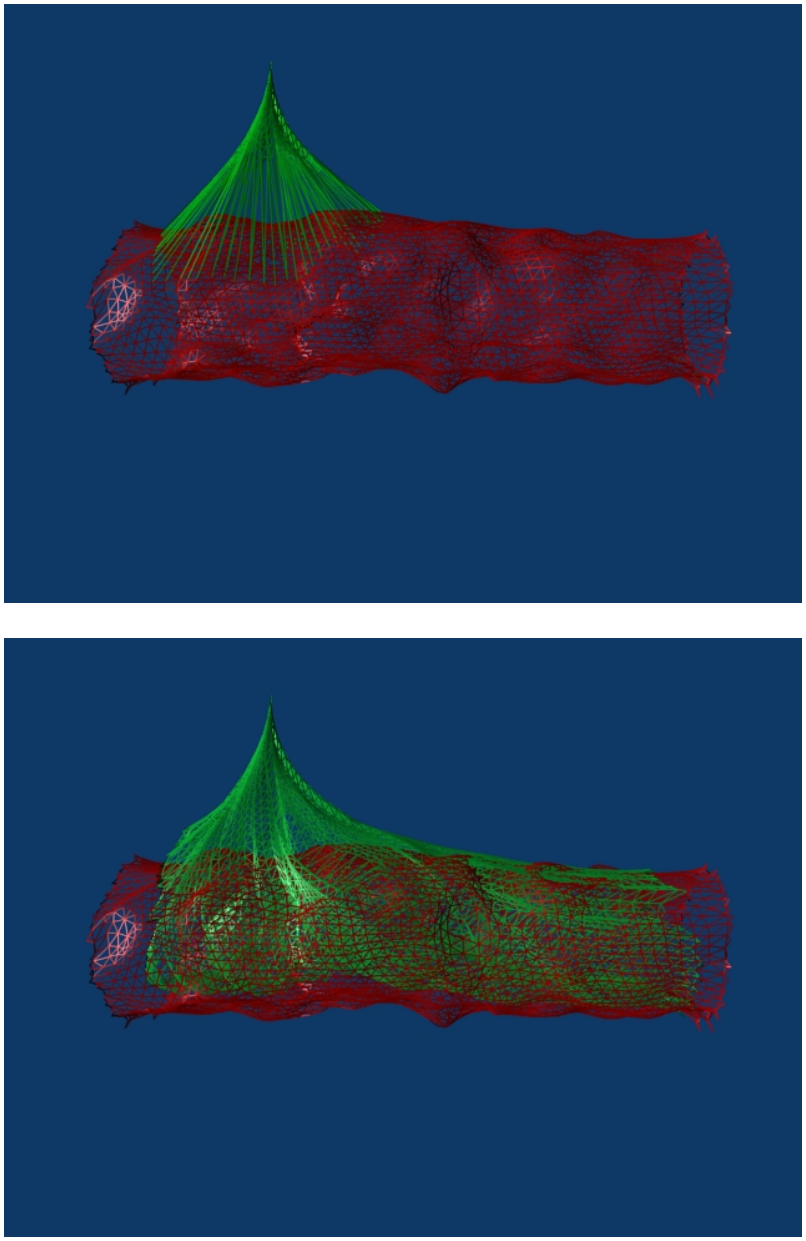


Figure 18: Top: deformation with displacement vector $0, 0, -15$ and a LOI of 10. (Red is the original triangulation and green is the deformed triangulation.) Bottom: same deformation, but with a LOI of 100.

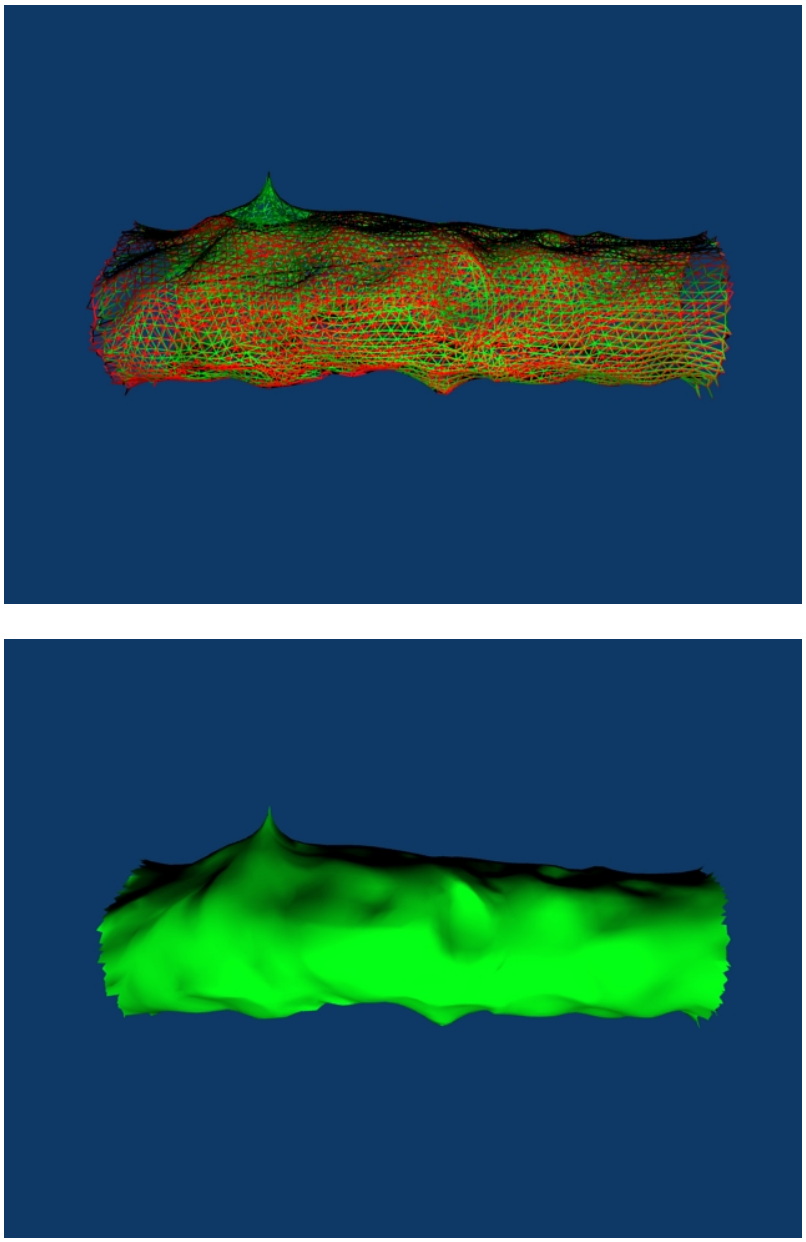


Figure 19: Top: deformation with displacement vector $0, 0, -5$ and a LOI of 100. (Red is the original triangulation and green is the deformed triangulation.) Bottom: the same deformation, but only the deformed shaded triangulation is shown.

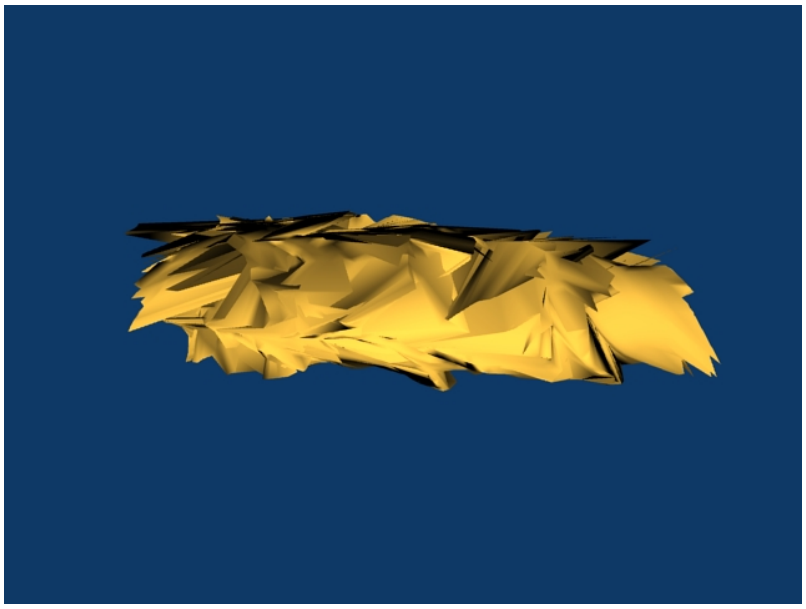


Figure 20: An unwanted deformation with displacement vector $0, 0, -1$, viscosity 1 and a LOI of 100.

The processing times of these deformations:

- With a viscosity of 0 was retrieved in approximately 1174 milliseconds.
- With a viscosity of 0.5 was retrieved in approximately 1102 milliseconds.
- With a viscosity of 1 was retrieved in approximately 1069 milliseconds.

We could conclude from this that the higher the viscosity value, the higher the damping and how quicker the deformation is done.

Suppose that we would use a viscosity of 1, even with image data guidance, the complete triangulation will be modified with a LOI of 100 and results in a sorry state. In figure 20 such an unwanted example of a fixed deformation version with a direction computation cube of 5, viscosity 1 and the displacement vector $0, 0, -1$ is shown. Nevertheless, neighboring vertices did improve their gradient values, which means that they are guided to a place nearer to an image data boundary.

20.2 VESSEL RESULTS WITH IMAGE DATA FROM START

Let us now consider the vessel triangulation from figure 21 with guidance of the image data from the start.

In figure 22 the fixed and dynamic version with a cube size of 3 and a viscosity of 0 are presented. The fixed version is created just under 1 second and the dynamic version in less than 8127 seconds (above 135 minutes!). However, the changed number of vertices are approximately the same, but the outcome of the result is not the same. The dynamic version is less smooth than the other shown two, which is not desired.

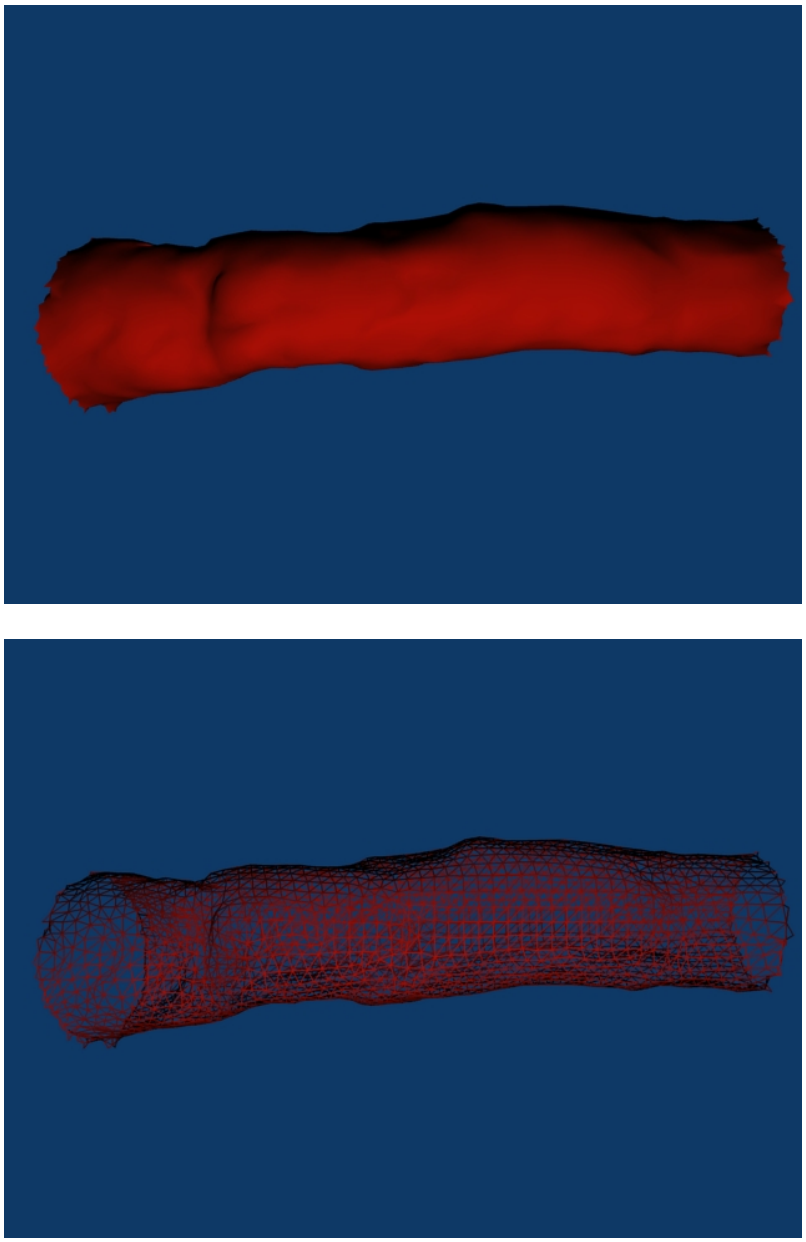


Figure 21: Top: original shaded vessel triangulation with 2504 vertices. Bottom: original vessel triangulation.

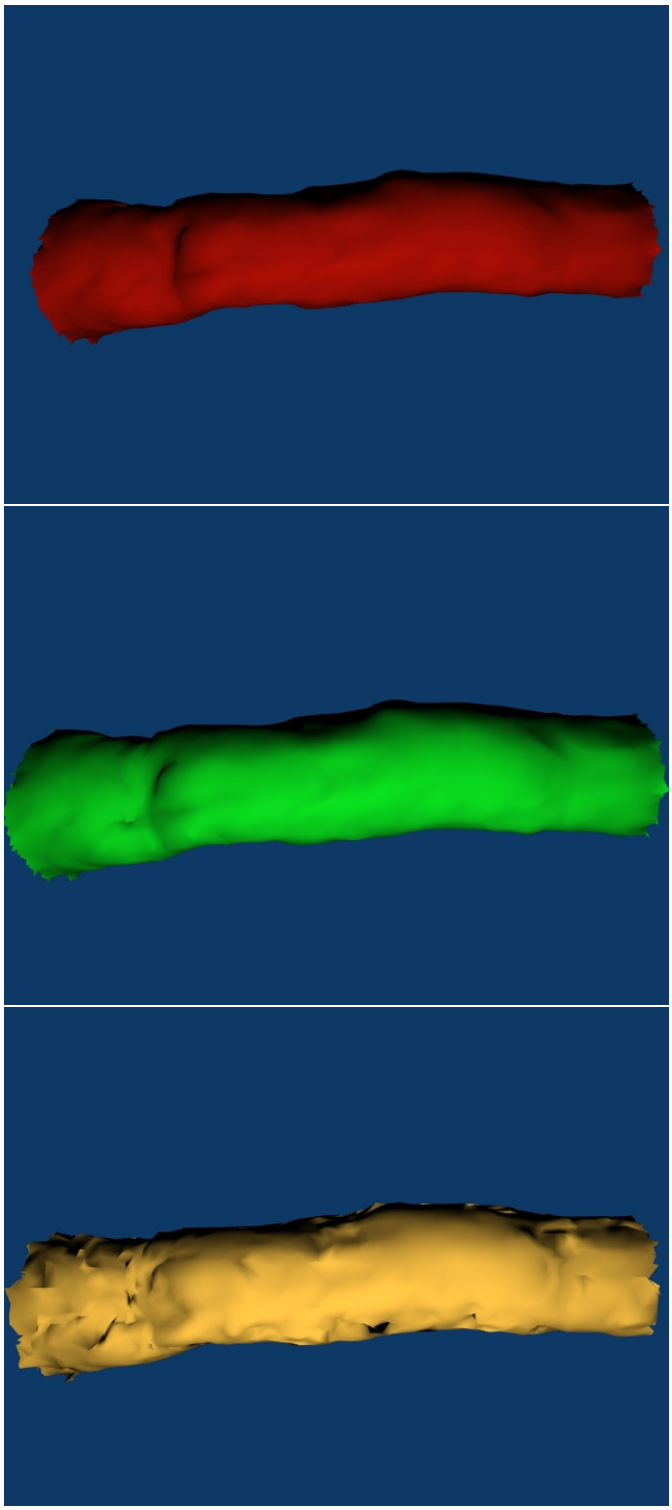


Figure 22: Shaded vessels, where top is the original shaded vessel triangulation with 2504 vertices. And the two others are the deformations with a viscosity of 0 and fixed (center) or dynamic (bottom) version.

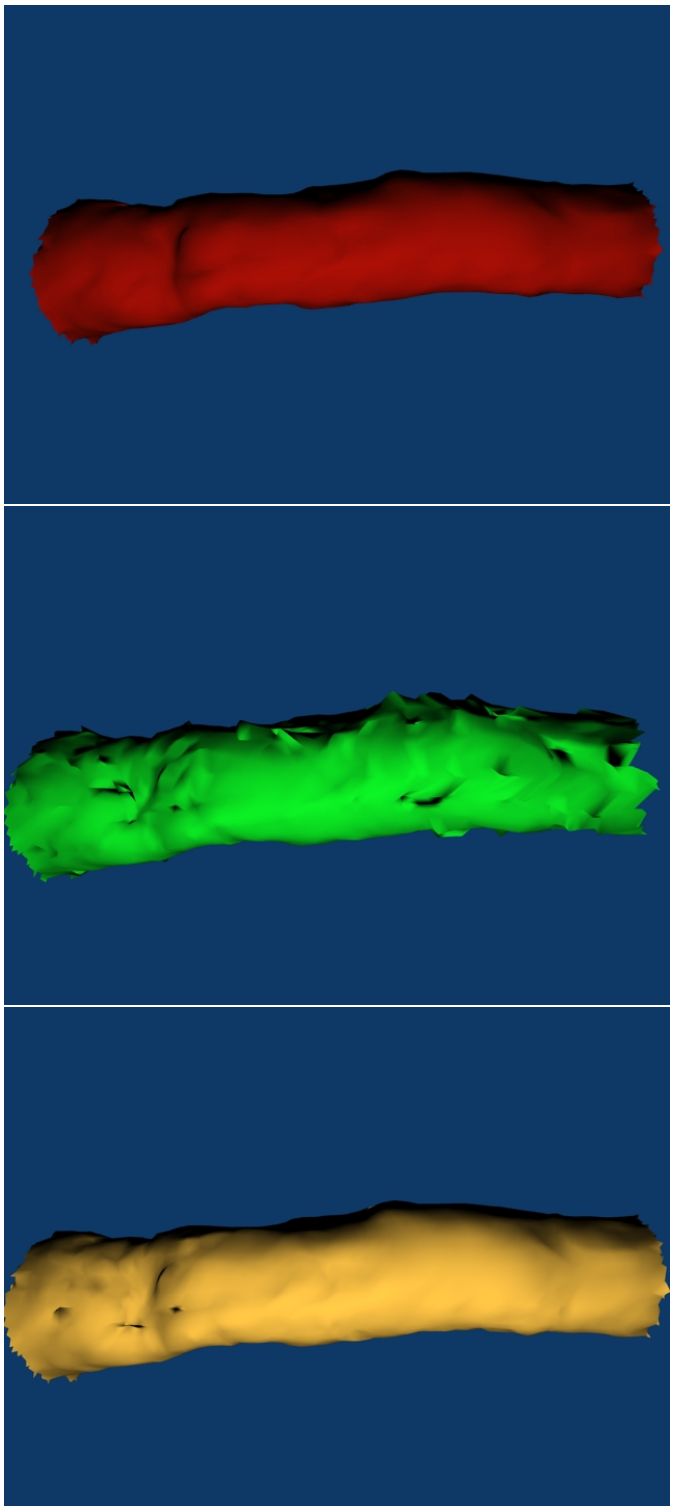


Figure 23: Shaded vessel, where top is the original shaded vessel triangulation with 2504 vertices. And the two others are the deformations with a viscosity of 0.1 and fixed (center) or dynamic (bottom) version.

	Processing time
Original	0
Dynamic with no image guidance	8
Dynamic with direction computation cube size 3	354
Dynamic with direction computation cube size 5	403
Fixed with no image guidance	8
Fixed with direction computation cube size 3	218
Fixed with direction computation cube size 5	148

Table 3: Several processing times (in rounded seconds) of the different bifurcation deformations.

Although we could reduce the LOI, such that we get for the dynamic version a smoother result, it would be good to investigate if we could use the viscosity to gain a smoother result, with an expected lower deformation time.

If we use a viscosity of 0.1, the computing time for the:

- Fixed deformation version increases to 64 seconds and reduces the number of deformed vertex coordinate-elements by 1833.
- Dynamic deformation version decreases considerable to less than 31 seconds and reduces the number of deformed vertex coordinate-elements by 3609.

From the resulting triangulations in figures 22 and 23, it can be concluded that the dynamic version increases in smoothness as we raise the damping value and the fixed version decreases in smoothness (think also again at figure 20). Thus the fixed version wasn't such a good idea to use after all, because it uses the fixed initial displacement.

20.3 BIFURCATION RESULTS

In figure 24 most of the used bifurcation is shown. The important difference with the previous vessel deformation is that we will here demonstrate the deformation behavior in the Carina. Therefore the deformation (with a viscosity of 0.1) starts in the Carina and the initial center vertex is moved upwards between the two vessels. In figures 25 and 26 an extreme initial displacement (0, 15, 0) example is given to signal how this deformation would work.

This extreme deformation shows that a large enough LOI (1000) could more or less accommodate the changes, but it results in a deformation of almost the complete triangulation. Also we can conclude from table 3 that the fixed version is quicker, but we should remember that it is not necessarily smoother than the dynamic version.

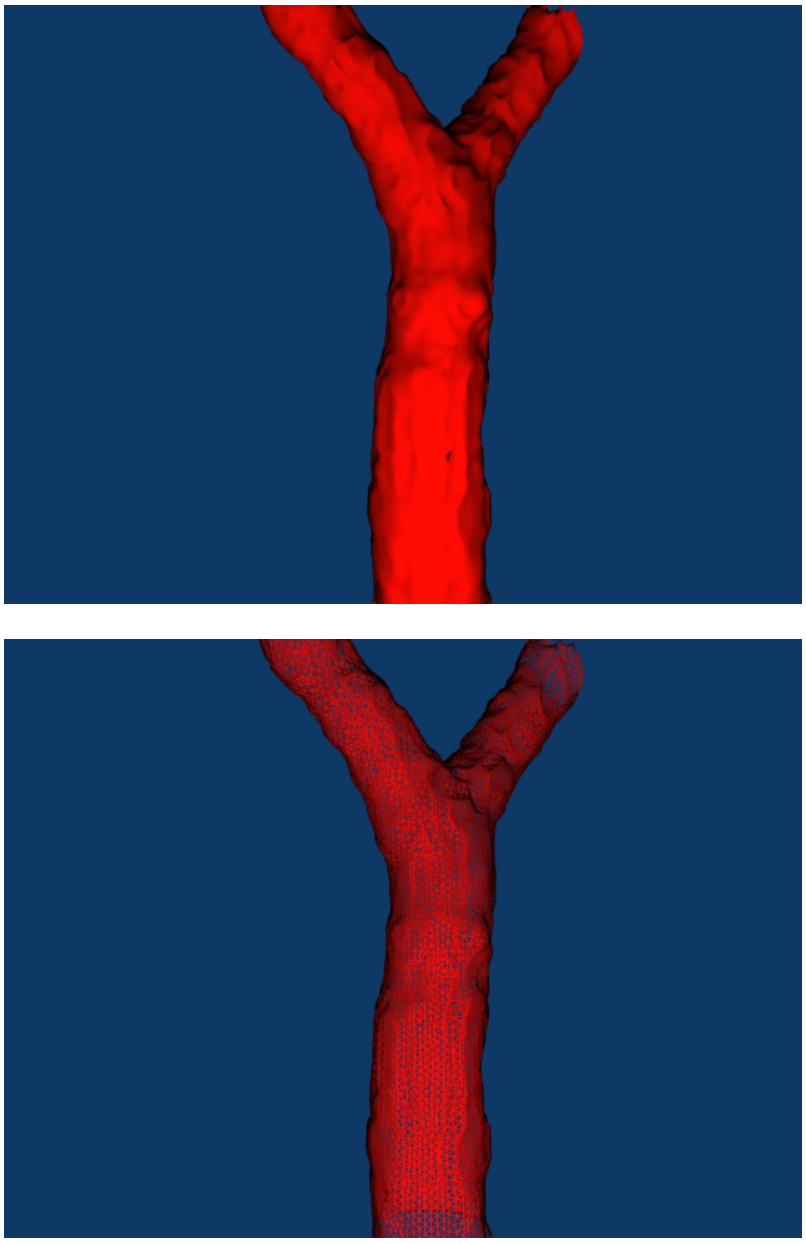


Figure 24: Top: original shaded bifurcation triangulation with 11440 vertices. Bottom: original bifurcation triangulation.

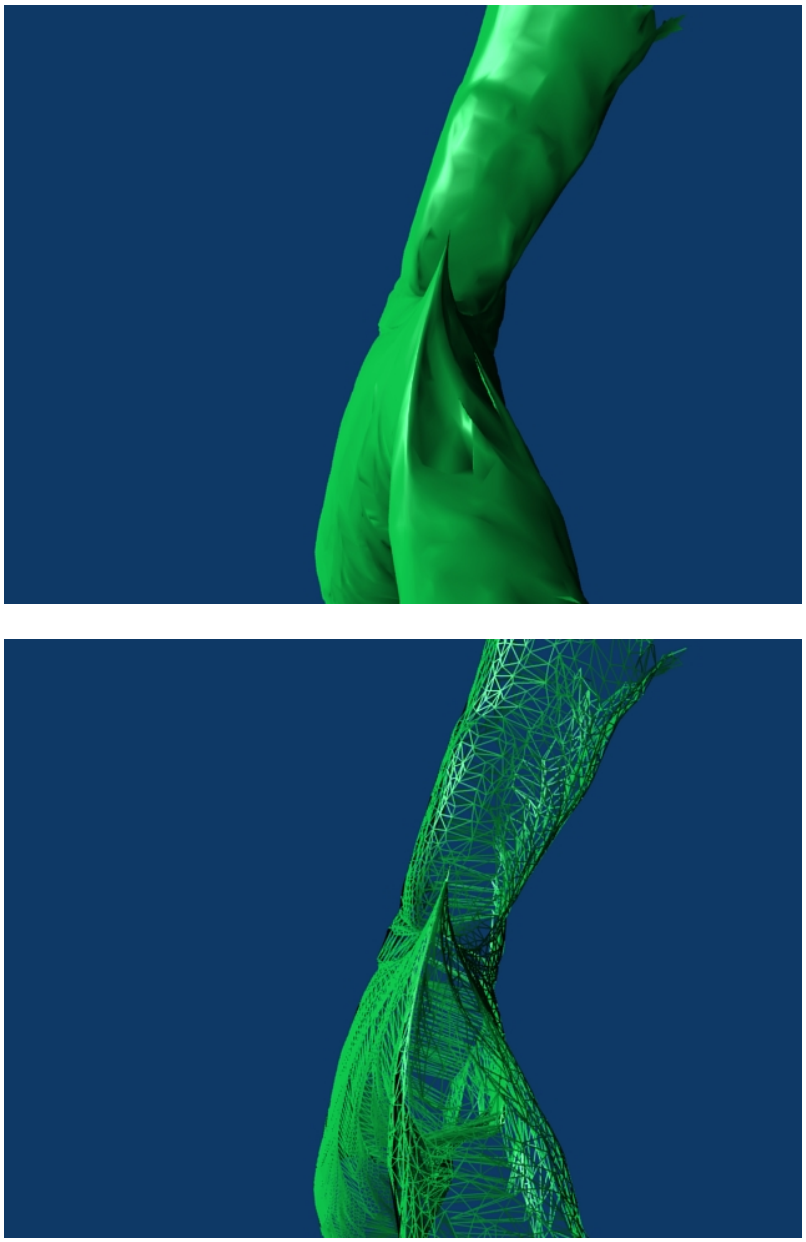


Figure 25: Top: extreme deformed shaded bifurcation triangulation with 11440 vertices. Bottom: extreme deformed bifurcation triangulation.

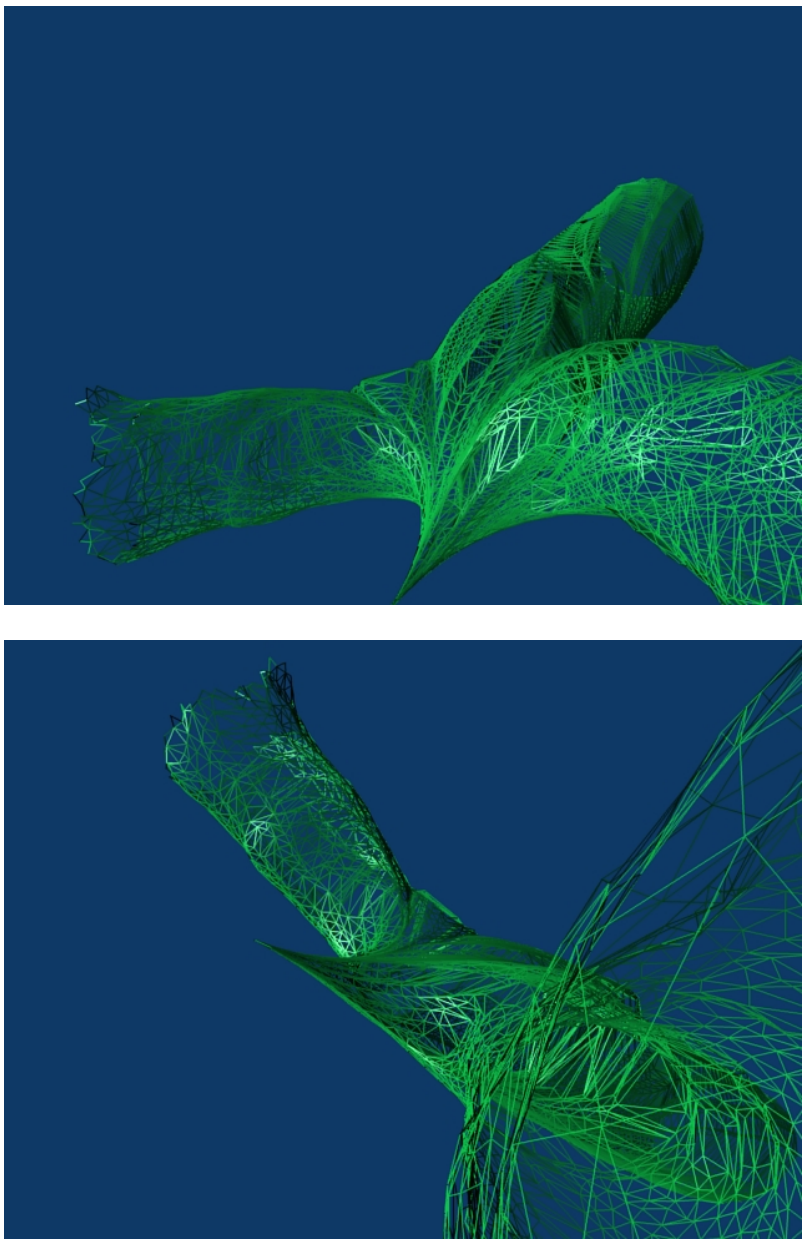


Figure 26: Two other views of the deformed triangulation of figure 25.

21 C/C++ IMPLEMENTATION

We would like to notice that the "soft" correction requirement is only implemented in the presented Matlab prototype. After discussions at Pie Medical Imaging b.v. a few in-house adjustments are made and therefore no image data (thus no "soft" correction) is used in the final developed C/C++ library. Therefore the given timetables and resulting deformations of the developed C/C++ library will differ from the results of the implementation in Matlab.

The most important adjustments are:

- Incorporating the possibility to adjust the three parameters for each vertex individually.
- Changing the "natural" parameters, such as mass, during the deformation process, to obtain a broader deformation peak instead of the small peaks from the Matlab prototype.

21.1 VESSEL RESULTS

In figure 27 the original vessel and two dynamic direction results are shown, where the center contains unwanted dents and the bottom one does not. These last two results are made with a test application that uses:

- Mass = 0.1.
- Elasticity = 0.9.
- Viscosity = 0.0 . . . 1.0 in steps of 0.1.

As can be seen, all values are rounded (down) at one position behind the comma. Nevertheless, the normalized viscosity range was not really narrowed in the previous parts of this thesis. Therefore the test application runs through the viscosity range, also it produced the tables 4 and 5 on a computer with the characteristics:

- Pentium 4, 2.60 GHz.
- 1 GB of RAM.
- Timing approximated with standard C/C++ GetTickCount function.

The two different vessel triangulations have both been treated in the same way (thus independent from their resolution!). It is an outward deformation with a simple initial displacement direction vector, where the z-coordinate contains a value of (-)5 (resp. (-)15) and the other two coordinates are zero. The time range did emerge over the different viscosity values, where the time is high the viscosity is 0.0 and where the time is low the viscosity is 1.0.

It can be concluded that the result of the triangulation without dents, thus with a viscosity of 0.6, is a good trade-off between no dents and a relative short processing time. However, we should see that the processing time also increases, when the LOI and/or displacement vector are increasing.

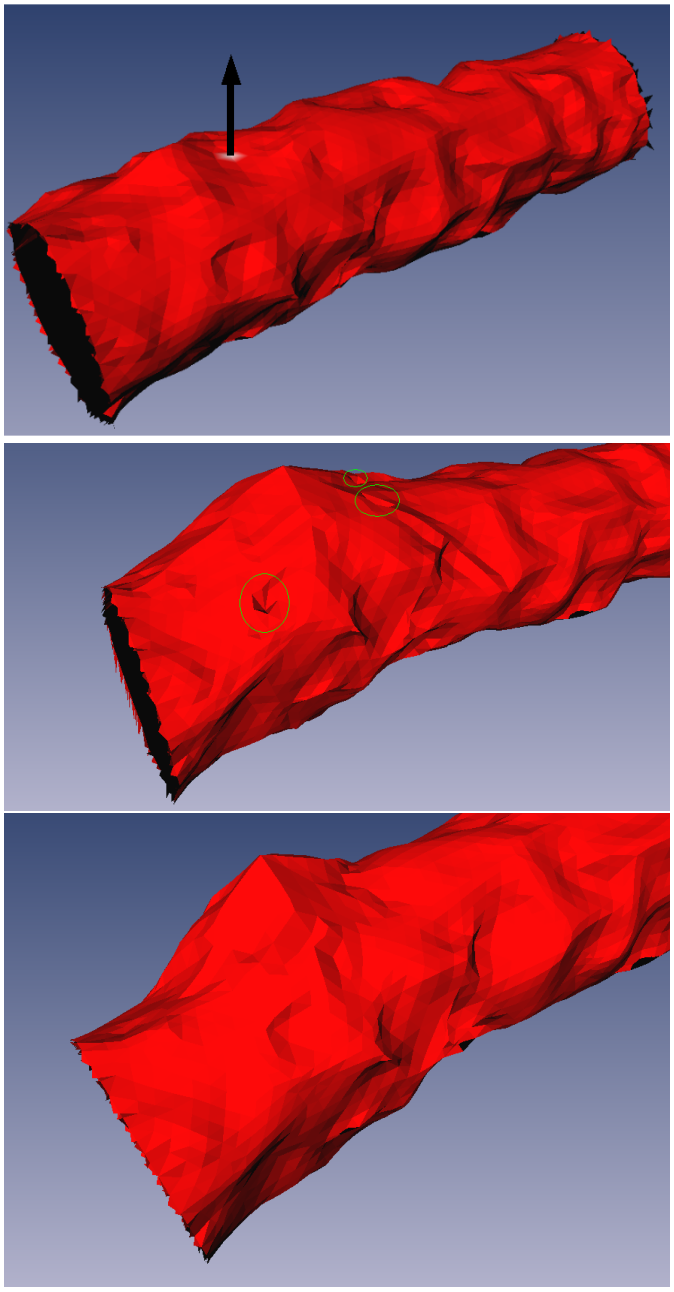


Figure 27: Top: original vessel with location and approximated direction of deformation. Center: deformation of vessel with viscosity -0.5, but with unwanted dents (see circles). Bottom: wanted (without unwanted dents) deformation of vessel with a viscosity of -0.6.

Direction	LOI	Displacement	Time at $\nu = 1$	Time at $\nu = 0$
Fixed	2	0,0,5 (0,0,15)	0 (0)	0 (0)
Dynamic	2	0,0,5 (0,0,15)	0 (0)	16 (30)
Fixed	5 (15)	0,0,5 (0,0,15)	0 (15)	0 (31)
Dynamic	5 (15)	0,0,5 (0,0,15)	0 (15)	16 (141)
Fixed	25	0,0,5 (0,0,15)	47 (47)	78 (63)
Dynamic	25	0,0,5 (0,0,15)	47 (47)	79 (219)

Table 4: Triangulated vessel with 2673 vertices and time in milliseconds at the boundaries of the viscosity interval.

Direction	LOI	Displacement	Time at $\nu = 1$	Time at $\nu = 0$
Fixed	2	0,0,-5 (0,0,-15)	0 (0)	0 (0)
Dynamic	2	0,0,-5 (0,0,-15)	0 (0)	16 (15)
Fixed	5 (15)	0,0,-5 (0,0,-15)	0 (0)	16 (16)
Dynamic	5 (15)	0,0,-5 (0,0,-15)	0 (16)	32 (234)
Fixed	25	0,0,-5 (0,0,-15)	46 (64)	93 (78)
Dynamic	25	0,0,-5 (0,0,-15)	47 (47)	203 (484)

Table 5: Triangulated vessel with 3411 vertices and time in milliseconds at the boundaries of the viscosity interval.

21.2 BIFURCATION AND TREE RESULTS

In figure 28 a vessel-tree is shown, also on the front page of this thesis a collection of deformed triangulations of this tree are presented. The deformation close-up images are taken from a deformation near the second bifurcation, which is just located inside the first bend of the long vessel.

As long as the deformation process is not passing by a Carina, the deformation just responds as a vessel would. However, there is a reasonable chance that a deformation with the C/C++ library implementation, especially if it starts in a Carina, creates a 3D triangulation with triangles which intersect each other. Actually this is sometimes already present in triangulations before the deformation, but they are usually less visible. We will introduce a possible solution in the discussion and outlook part.

In table 6 some indicating deformation times (with the same application as above) of a bifurcation are shown, where a vertex of the Carina is moved outwards. It can be noticed that the LOI is relatively high to accommodate all changes in the bifurcation, but sadly the problem of crossing triangles also did occur during these deformations.

Figure 28: A side view of a vessel-tree.

Direction	LOI	Displacement	Time at $\nu = 1$	Time at $\nu = 0$
Fixed	15	0,15,0	15	47
Dynamic	15	0,15,0	31	250
Fixed	160	0,15,0	1828	1875
Dynamic	160	0,15,0	1844	2828

Table 6: Triangulated bifurcation with 11439 vertices and time in milliseconds at the boundaries of the viscosity.

Part VI

DISCUSSION AND OUTLOOK

22 DISCUSSION AND OUTLOOK

To conclude this thesis, it would be good to discuss the results in a short overview and to look ahead for possible future work.

22.1 DISCUSSION

This thesis has given us a '3D interactive mesh deformation' - method, which is tuned for 3D vessel-trees. Our method is implemented as a library in the C/C++ programming language and as a prototype with image data guidance in Matlab. The developed method will indeed take as obligatory input a 3D triangulation (vessel-tree) and the wanted initial displacement of a specific center vertex in this vessel-tree, also the output contains a "natural" deformed 3D triangulation.

Furthermore we could determine the "natural" parameters for the prototype and the implementation as:

- A mass of approximately 0.1.
- An elasticity of approximately 0.9 (for the prototype 0.99).
- And a viscosity range between 0 and 1 (during testing of the C/C++ library implementation a more or less general value of 0.6 emerged for it).

Also we concluded that the viscosity range should not be narrowed down, such that we can accommodate e.g. different diseases and age-groups. Nevertheless, when using a bigger viscosity value and/or smaller LOI value, the deformation time will usually drop. It should thus be noticed that finding the most desired deformation of a triangulation could take a few deformations to find the optimal solution for the user.

22.2 FUTURE WORK

A problem arises in the C/C++ implementation, for deformations which contain bifurcations. It especially concerns deformations with a starting vertex inside the Carina. If the outward displacement of the center vertex is bigger than the area of the Carina can handle, problems are starting to come. The triangles start to intersect each other in the region outside the Carina area.

Let us consider a couple of solution options:

- Return to the Matlab prototype and accept the (small) peak in the first deformation run. Use the deformation method for a rerun on the deformed triangulation, with a displacement vector that reduces the deformation peak. Here we should select such a displacement that the first deformation is not completely undone.
- Remove the requirement that the triangulation property needs to be maintained at all costs. The intersecting triangle part of the triangulation can than be re-triangulated, with help of, for

example, a refinement and decimation process as is described in [6], [12] and [33].

To finish this thesis, it would be good to consider for the future:

- Incorporation of image data in the C/C++ library implementation. This might be of additional use to pursue the goal of a "perfect" segmentation.
- We might reconsider this method, such that for different resolutions (as noticed on page 41) of the same segmentation also the same deformation occurs.

BIBLIOGRAPHY

[1] Manuel Alfonseca. Genetic algorithms. In APL '91: Proceedings of the international conference on APL '91, New York, NY, USA, pages 1-6, 1991. ACM. (Cited on page 15.)

[2] Chandrajit L. Bajaj. Implicit Surface Patches (chapter 3). 1997. (Cited on pages 16 and 19.)

[3] T. Belytschko, T. Black. Elastic Crack Growth in Finite Elements With Minimal Remeshing. International Journal for Numerical Methods in Engineering, 45(5): 601-620, 1999. (Cited on page 13.)

[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational geometry: algorithms and applications. Springer Science & Business, 1997, ISBN 354061270X, 9783540612704 (Cited on page 7.)

[5] J. Black, G. Hastings. Handbook of biomaterial properties. Chapman & Hall, 1998, ISBN 0412603306. (Cited on page 45.)

[6] A.J. Bulpitt, N. D. Efford. An Efficient 3D Deformable Model with a Self-Optimizing Mesh. Image and Vision Computing, 14(8): 573-580, August 1996. (Cited on pages 20, 26, and 70.)

[7] Kup-Sze Choi, Hanqiu Sun, Pheng-Ann Heng. Interactive deformation of soft tissues with haptic feedback for medical learning. IEEE Transactions on Information Technology in Biomedicine, 7(4): 358-363, 2003. (Cited on pages 22 and 26.)

[8] Kup-Sze Choi, Hanqiu Sun, Pheng-Ann Heng. An efficient and scalable deformable model for virtual reality-based medical applications. Artificial Intelligence in Medicine, 32(1): 51-69, 2004. (Cited on pages 22, 23, and 26.)

[9] T.A. Cruse. Numerical Solutions in Three Dimensional Elastostatics. International Journal of Solids Structures, 5: 1259-1274, 1969. (Cited on page 13.)

[10] Boris N. Delaunay. Sur la sphère vide. Bulletin of Academy of Sciences of the USSR, (6): 793-800, 1934. (Cited on page 7.)

[11] Hervé Delingette. Simplex Meshes: a General Representation for 3D Shape Reconstruction. Technical Report 2214, INRIA, March 1994. (Cited on page 8.)

[12] Hervé Delingette. General Object Reconstruction Based on Simplex Meshes. Int. J. Comput. Vision, 32(2): 111-146, 1999. (Cited on pages 20 and 70.)

[13] L. Filice, I. Alfaro, F. Gagliardi, E. Cueto, F. Micari, F. Chinesta. A preliminary comparison between finite element and meshless simulations of extrusion. Journal of materials processing technology, 209: 3039-3049, 2009. (Cited on page 14.)

[14] C. Galli, D. Bia, Y. Zocalo, et al. Vascular heterografts for hemodyalisis access: analysis of elastic and viscous matching factor between human and ovine vessels. Lat. Am. Appl. Res., 37(3): 201-206, 2007. ISSN 0327-0793. (Cited on page 46.)

[15] Lothar Gaul, Martin Kögl, Marcus Wagner. Boundary element methods for engineers and scientists: an introductory course with advanced topics. Springer, 2003, ISBN 3540004637, 9783540004639. (Cited on pages 12 and 13.)

[16] R. C. Gonzalez, R. E. Woods. Digital Image Processing. Prentice Hall, Upper Saddle River, NJ, 2008, ISBN 9780131687288. (Cited on page 20.)

[17] Robert Hooke. Lectures de potentia restitutiva, or of spring, explaining the power of springing bodies. London, Printed for John Martyn, Printer to the Royal Society, at the Bell in St. Pauls Church-Yard, 1678. Currently available online at: http://posner.library.cmu.edu/Posner/, last visited 4 - 2 - 2010.

[18] Slobodan Ilic, Pascal Fua. Using Dirichlet Free Form Deformation to Fit Deformable Models to Noisy 3-D Data. In European Conference on Computer Vision, pages 704-717, 2002. (Cited on page 18.)

[19] Doug James, Dinesh Pai. ArtDefo: accurate real time deformable objects. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, pages 65-72, 1999. ACM Press/Addison-Wesley Publishing Co. (Cited on pages 13 and 14.)

[20] Carlo Janna, Andrea Comerlati, Giuseppe Gambolati. A comparison of projective and direct solvers for finite elements in elastostatics. Advances in Engineering Software, 40(8): 675-685, 2009. (Cited on page 12.)

[21] Michael Kass, Andrew Witkin, Demetri Terzopoulos. Snakes: Active contour models. International Journal of Computer Vision, 1(4): 321-331, January 1988. (Cited on page 11.)

[22] A.R. Khoei, S.A. Gharehbaghi. Three-dimensional data transfer operators in large plasticity deformations using modified-SPR technique. In Applied Mathematical Modeling, 33: 3269-3285, 2009. (Cited on page 13.)

[23] Kazuya G Kobayashi, Katsutoshi Ootsubo. t-FFD: free-form deformation by using triangular mesh. In SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications, New York, NY, USA, pages 226-234, 2003. ACM. (Cited on pages 16, 17, and 18.)

[24] Jyrki Lötjönen, Isabelle E. Magnin, L. Reinhardt , Jukka Nenonen, Toivo Katila. Automatic Reconstruction of 3D Geometry Using Projections and a Geometric Prior Model. In MICCAI '99: Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention, London, UK, pages 192-201, 1999. Springer-Verlag. (Cited on page 16.)

[25] Ravikanth Malladi, James A. Sethian, Baba C. Vemuri. Shape Modeling with Front Propagation: A Level Set Approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(2): 158-175, February 1995. (Cited on page 11.)

[26] MatWeb is a searchable database of material properties. Located at: http://www.matweb.com, last visited 4 - 2 - 2010. (Cited on page 45.)

[27] Tim Mcinerney, Demetri Terzopoulos. Deformable Models in Medical Image Analysis: A Survey. Medical Image Analysis, 1: 91-108, 1996. (Cited on page 11.)

[28] Tim McInerney, Demetri Terzopoulos. Medical image segmentation using topologically adaptable surfaces. In CVRMed-MRCAS '97: Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery, London, UK, pages 23-32, 1997.Springer-Verlag. (Cited on page 11.)

[29] N. Moës, J. Dolbow, T. Belytschko. A Finite Element Method for Crack Growth Without Remeshing. International Journal for Numerical Methods in Engineering, 46(1): 131-150, 1999. (Cited on page 13.)

[30] J. Montagnat, H. Delingette, N. Ayache. A Review of Deformable Surfaces: Topology, Geometry and Deformation. Image and Vision Computing, 19: 1023-1040, 2001. (Cited on pages 11 and 12.)

[31] Andrew Nealen, Takeo Igarashi, Olga Sorkine, Marc Alexa. Laplacian mesh optimization. In Proceedings of ACM GRAPHITE, pages 381-389, 2006. (Cited on page 18.)

[32] G.P. Nikishkov. Introduction to the Finite Element Method. Lecture Notes 2009. University of Aizu, Aizu-Wakamatsu, Japan. (Cited on page 12.)

[33] A. Noord, R.M. Aten. Geometric simplification algorithms for surfaces. School for Computing and Cognition, University of Groningen, Groningen, Netherlands, 1998. (Cited on pages 20 and 70.)

[34] Jay Oswald, Robert Gracie, Roopam Khare, Ted Belytschko. An extended finite element method for dislocations in complex geometries: Thin films and nanotubes. Comput. Methods Appl. Mech. Eng, 198: 1872-1886, 2009. (Cited on page 13.)

[35] TH. Rooswinkel, Dr. H. H. Kreutzer. Biologie van de mens. P. Noordhoff N.V., Groningen - Djakarta, 1952. (Cited on page 46.)

[36] Daniel Bia Santana, Juan Gabriel Barra, Juan Carlos Grignola, Fernando Florencio Gines, Ricardo Luis Armentano. Pulmonary artery smooth muscle activation attenuates arterial dysfunction during acute pulmonary hypertension. Journal of Applied Physiology 98: 605-613, 2005. (Cited on page 46.)

[37] R.A. Schoenmaker. Simulated Demolition by imploding: the wonders of Blender. School for Computing and Cognition, University of Groningen, Groningen, Netherlands, 2008. (Cited on pages 12 and 21.)

[38] Thomas Sederberg, Scott Parry. Free-form deformation of solid geometric models. SIGGRAPH Comput. Graph., 20(4): 151-160, 1986. (Cited on page 16.)

[39] Florent Segonne, Eric Grimson, Bruce Fischl. A Genetic Algorithm for the Topology Correction of Cortical Surfaces. In Proceedings of Information Processing in Medical Imaging, LNCS, pages 393-405, 2005. (Cited on page 15.)

[40] Olga Sorkine. Differential Representations for Mesh Processing. Computer Graphics Forum, 25(4): 789-807, 2006. (Cited on page 18.)

[41] N. Sukumar, B. Moran, T. Belytschko. The natural element method in solid mechanics. International Journal for Numerical Methods in Engineering, 43: 839-887, 1998. (Cited on page 14.)

[42] N. Sukumar. Fortran Code; Natural Element Method for Two-Dimensional Elastostatics; Theoretical and Applied Mechanics, Northwestern University, Evanston, IL 60208, June 1998: http://dilbert.engr.ucdavis.edu/~suku/, last visited 4 - 2 - 2010. (Cited on page 14.)

[43] Jussi Tohka. Global Optimization of Deformable Surface Meshes Based on Genetic Algorithms. Institute of Signal Processing, Tampere University of Technology, Tampere, Finland, 2001. (Cited on page 15.)

[44] Jussi Tohka. Surface Extraction from Volumetric Images Using Deformable Meshes: A Comparative Study. In the seventh European Conference on Computer Vision, pages 350-364, 2002. (Cited on page 15.)

[45] Jussi Tohka. Global optimization-based deformable meshes for surface extraction from medical images. Digital Media Institute / Signal Processing, Tampere Tampere University of Technology, Tampere, Finland, 2003. (Cited on page 15.)

[46] M. Vasilescu, Demetri Terzopoulos. Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision. In Proceedings of Computer Vision and Pattern Recognition conference, pages 829-832, 1992. IEEE Computer Society Press. (Cited on pages 20, 21, 26, 27, and 29.)

[47] Baba C. Vemuri, Yanlin Guo. Snake Pedals: Compact and Versatile Geometric Models with Physics-Based Control. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22: 445-459, 2000. (Cited on page 11.)

[48] George Vogiatzis, Philip Torr, Roberto Cipolla. Bayesian Stochastic Mesh Optimisation for 3D Reconstruction. British Machine Vision Conference, 2: 711-718, 2003. (Cited on page 16.)

[49] Max Wardetzky, Saurabh Mathur, Felix Kälberer, Eitan Grinspun. Discrete Laplace operators: No free lunch. In Symposium on Geometry Processing, pages 33-37, Jul 2007. (Cited on page 18.)

[50] Shin Yoshizawa, Alexander G. Belyaev, Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications, New York, NY, USA, pages 247-253, 2003. ACM. (Cited on page 16.)

[51] J. Yvonnet, F. Chinesta, P. Lorong , D. Ryckelynck. The constrained natural element method (C-NEM) for treating thermal models involving moving interfaces. International Journal of Thermal Sciences, 44: 559-569, 2005. (Cited on page 14.)

[52] Zhengyou Zhang. Iterative point matching for registration of free-form curves. Technical Report RR-1658, INRIA, 1992. (Cited on page 16.)

[53] O. C. Zienkiewicz, Robert Leroy Taylor, J. Z. Zhu. The finite element method: its basis and fundamentals. Butterworth-Heinemann, 2005, ISBN 0750663200, 9780750663205. (Cited on page 12.)

[54] Denis Zorin, Peter Schröder, Wim Sweldens. Interactive multiresolution mesh editing. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, New York, NY, USA, pages 259-268, 1997. ACM Press/Addison-Wesley Publishing Co. (Cited on pages 19 and 20.)