FAKULTÄT FÜR !NFORMATIK

Sketch-based Modelling for Volume Visualization

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Computergraphik/Digitale Bildverarbeitung

eingereicht von

Nicolas Pühringer
Matrikelnummer 0525839

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer/Betreuerin: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Univ.-Ass. Dipl.-Ing. Dr.techn. Stefan Bruckner

Wien, 22.07.2009

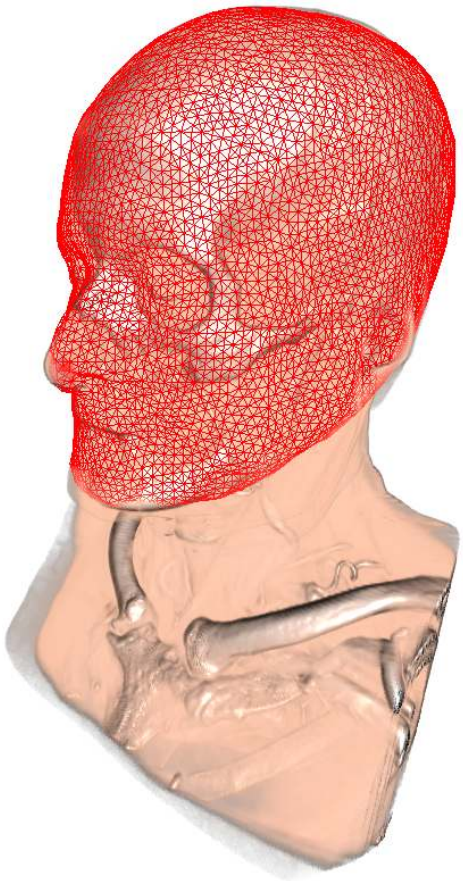
_____ (Unterschrift Verfasser/in)

_____ (Unterschrift Betreuer/in)

Nicolas Pühringer

Sketch-based Modelling for Volume Visualization

Master Thesis



supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Univ.-Ass. Dipl.-Ing. Dr.techn. Stefan Bruckner

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Erklärung zur Verfassung der Arbeit

Nicolas Pühringer, Stumpergasse 44/17, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. Juli 2009 _____

(Unterschrift Verfasser)

Abstract

In the recent years the use of touch-sensitive input devices (e.g., tablet devices) heavily increased. Especially for drawing or sketching tasks, these devices - in combination with new user interface approaches - yield many possibilities to optimize traditional workflows.

This thesis provides an approach for integrating this new user interfaces techniques into a volume visualization framework. The main goal is to account for the frequently encountered task of selecting specific structures of interest in a volume dataset which can not be separated by a transfer function setup.

First, a gesture-based user interface is incorporated to build up a fluid and intuitive workflow without disrupting the user's attention from the main working area. Further, a sketch-based modelling approach allows the user to easily generate 3D models out of 2D input sketches. These models serve as an initial selection on a structure of interest within a volume dataset. To automatically fit the generated models on the volume features, we present an algorithm for automatic deformation of mesh models on volume structures, resulting in a good approximation of the specific area.

This flexible combination of techniques allows the user to achieve selections in a volume dataset within minutes of work. These can subsequently be used for masking, cropping or segmentation operations on the volume.

Kurzfassung

Berührungssensitive Eingabegeräte, wie etwa Tablet PCs, erleben seit einiger Zeit einen starken Aufschwung. Vorallem für Skizzen- oder Zeichenarbeiten lässt sich der Arbeitsprozess durch diese neuen Eingabegeräte stark optimieren.

Diese Diplomarbeit stellt eine Integration genannter neuer Technologien in ein Volumenvisualisierungssystem vor. Ziel ist hierbei eine Lösung für ein häufiges Problem der Volumensvisualisierung: das Selektieren spezieller Strukturen die nicht durch die Transferfunktion separierbar sind.

Durch eine gesten-gesteuerte Benutzeroberfläche wird ein schneller und reibungsloser Arbeitsprozess gewährleistet ohne dabei die Aufmerksamkeit des Benutzers vom Hauptarbeitsbereich abzulenken. Weiters wird dem Benutzer durch ein intuitives Modellierungsverfahren ermöglicht 3D Modelle aus zweidimensionalen Eingabepfaden zu erstellen. Die erstellten Modelle dienen als eine Startselektion einer gewünschten Struktur im Volumensdatensatz. Diese Startselektion wird in Folge durch ein Modellverformungsverfahren automatisch an die Volumensstrukturen angepasst und resultiert in einer Selektion einer bestimmten Region des Volumens.

Durch diese Kombination von verschiedenen Techniken können Selektionen in einem Volumensdatensatz schnell und intuitiv erreicht werden. Das Ergebnis kann in Folge zur Maskierung, Beschneidung oder Segmentierung der Volumensstrukturen verwendet werden.

Contents

Chapter 1

Introduction

In science one tries to tell people, in such a way as to be understood by everyone, something that no one ever knew before. But in poetry, it's the exact opposite.

Paul Dirac

The main goal of this thesis is to achieve an intuitive and fast workflow for selecting structures of interest within a volume dataset. To achieve this, a combination of recent developments in the area of gesture-based application control, sketch-based modelling and deformable models is incorporated into traditional volume visualization. This first chapter provides an introduction on the different topics to illuminate the background of this work. Further the main goals are proposed and finally an overview on the structure of the remainder of this thesis is given.

1.1 Sketch-based Interfaces

Research on new user interfaces is a major topic in computer science. Both on the hardware and software side of the user interface chain a vast effort

Figure 1.1: Tablet and pen input device by Wacom [56].

is put into developing intuitive and easy-to-use ways to control applications and machines. Thereby the traditional mouse/keyboard interface is replaced by more natural input devices for specific tasks. Especially the employment of touch sensitive devices to achieve more direct control over the workflow found increasing adoption in the past years and yields many possibilities for further interface improvements.

Modern smart phones already make excessive use of touch input by the user for intuitively navigating through menu structures. A good representative for this type of devices is the *Apple iPhone* which allows the user to control applications completely with one finger by drawing gestures directly on the display. For smart phones the main advantage of this interface type is the bigger display that can be built over the whole phone extent due to omitting the keyboard.

But also for computer science this type of user interface can be advantageous. Realized with pen and tablet devices (see Figure 1.1), tasks where drawing or sketching of the user is required are massively improved by this new input techniques. The user can directly work on the screen instead of using the mouse for drawing in an unnatural manner.

Once a hardware setup is achieved, there are diverse ways of using the touch sensitive input. Therefore the area of sketch-based interfaces has been the topic of research for many years and ranges from gesture-based methods to control applications to modelling and construction approaches with direct

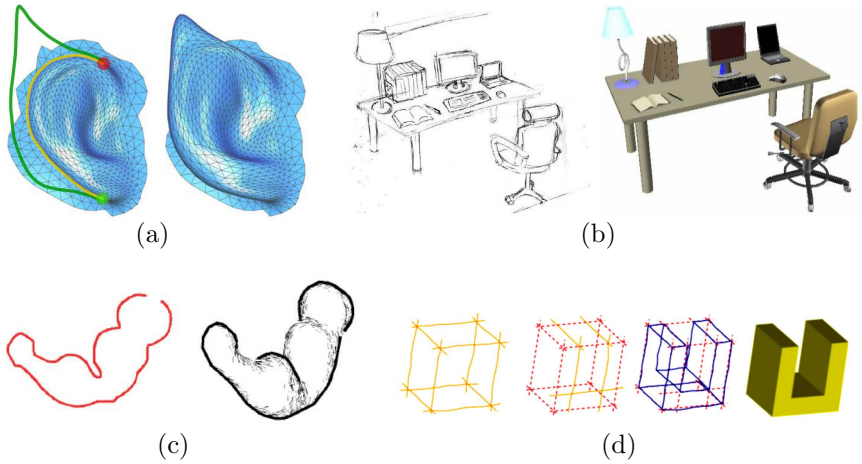


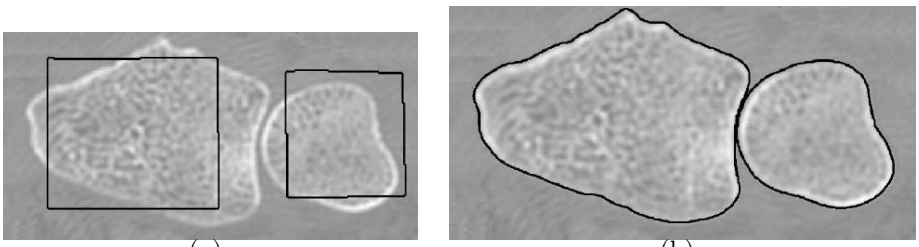
Figure 1.2: Some examples of sketch-based modelling approaches for: (a) mesh editing [34], (b) scene construction [51], (c) model generation [26], (d) CAD systems [10].

drawing input by the user. Also the area of gesture recognition is closely related to these interfaces as many use a gesture user input to trigger different application states.

In this thesis, both gesture-based interfaces to control application workflow as well as sketch-based modelling will be main topics. Sketch-based modelling employs new methods for generating three-dimensional models in an easy and intuitive manner.

Traditional modelling approaches are very time consuming and need expertise by the user. These limitations are overcome with sketch-based modelling by enabling the user to generate a 3D model out of a 2D sketch input: the user draws an input sketch representing the silhouette of a model which is then generated by inflating the contour to the third dimension. This provides a very fast way to generate models without the need for expert knowledge in traditional modelling techniques. Many approaches were proposed in the last years to employ different modelling methods with the help of sketch-based interfaces (see Figure 1.2 for some examples). Olsen *et al.* [37] provide an overview over the state-of-the-art.

Diverse applications for sketch-based interfaces with touch sensitive devices are already in use and are becoming increasingly popular. Still the



(a) (b)

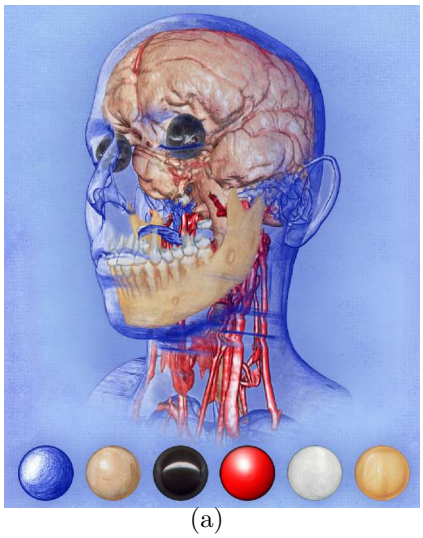
Figure 1.3: A typical application of deformable models/snakes. Input contours (a) are using image information to fit specific regions (b) [20].

real upswing of these technologies is yet to come and research in this area will push these devices to even further possibilities for improving the user experience.

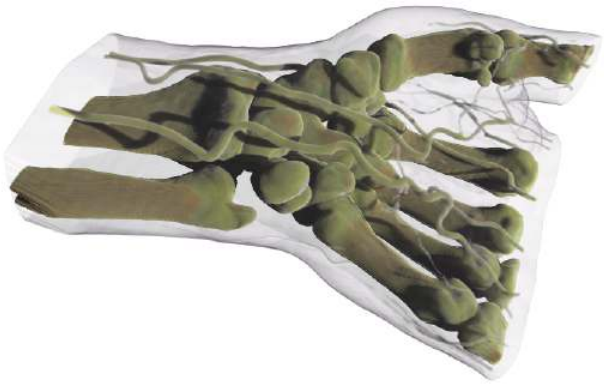
1.2 Deformable Models

Deformable models have their origin in a two-dimensional structure fitting approach called *snakes* [28]. The main idea is to take an input contour on an image and to deform it according to parameters calculated out of the image information while preserving the main topology of the contour itself. The main advantage of the snakes algorithm is the possibility of pattern matching in noisy image environments. These algorithms are used in many different fields of computer vision, important areas of application are pattern matching algorithms and image segmentation. Figure 1.3 illustrates an example of such a snake algorithm.

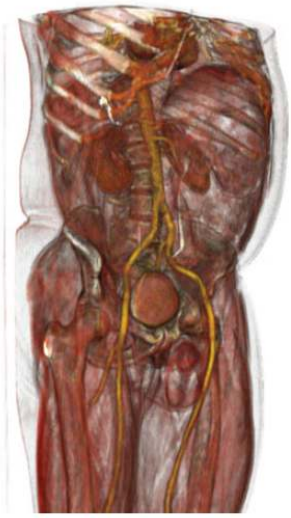
Bringing this idea to the three-dimensional space, a snake algorithm is able to fit 3D input shapes to structures in 3D images/datasets while preserving the main input shape. In this thesis a simple deformable model algorithm is used to let a roughly approximated input mesh model snap to nearby regions of interest within a volumetric dataset in order to achieve a surface approximation of the region.



(a)



(b)



(c)

(d)

Figure 1.4: Different methods of volume visualization: (a) style transfer functions [7], (b) self-shadowing [17], (c) obscurance-based [46], (d) advanced illumination [18].

1.3 Volume Visualization

The term volume visualization summarizes all technologies and methods that deal with the illustration of volumetric datasets. These datasets are usually acquired by *Magnetic Resonance Imaging (MRI)* or *Computed Tomography (CT)* scans of objects and are digitally represented by three-dimensional sample point grids of intensity values. With the well-known ray casting algorithm the intensity values are composited to generate an image representation using a transfer function to assign a color and opacity to each intensity value.

Due to modern graphic processors, high quality interactive visualizations of volumetric datasets are possible even on consumer hardware. Data can be explored and altered in real-time to achieve deeper insight into presented structures. Many different types of visualization such as illustrative or stylized methods were proposed to further enhance perception of different structures. Figure 1.4 illustrates some exemplary methods of different volume visualization approaches.

An important topic for interactively working with volume datasets is selecting specific areas of interest in the volume structures. Here, many tasks can be accomplished using a suitable transfer function setup to illustrate regions of certain intensity ranges while damping or completely hiding unwanted intensity values. The selection of areas where structures of similar intensities values lie co-located is more problematic. Methods like cropping boxes are limited in their possibilities of adapting to certain structures in the volume. This concern is assessed in this thesis by incorporating a combination of sketch-based modelling approaches with traditional volume visualization.

1.4 Approach and Goals

The main goal of this thesis is to investigate the combination of sketch-based interfaces and modelling methods in a volume visualization framework. Many benefits can be drawn from the employment of such a setup:

- Fast workflow
- Nearly button-free user interface usage
- Intuitive 3D model generation
- Models self-fitting to regions of interest on a volume dataset

To allow the user to quickly and easily switch between important operation modes of the volume visualization framework a gesture detection system is supplied. With predefined and easy-to-memorize patterns the user can change operations by simply drawing the corresponding gesture on the screen. The user is not disrupted in his main workflow by searching for different options in a toolbar. These usually grow to a huge quantity in scientific applications and misdirect the concentration from the working area.

The next important part of this thesis is formed by a sketch-based modelling approach proposed by Igarashi et al. [26]. To enable model generation, the user switches into the according mode by drawing a gesture on the screen. Now the next sketch input is interpreted as the basis path to the model inflation stage. Once the silhouette of the desired model is sketched, a three-dimensional triangle mesh is generated out of the input and rendered to the screen. To further account for more complex shapes different editing operations on the meshes are provided by combining or intersecting several sketches. This type of modelling approach allows quick and easy generation of basic model shapes and requires little expertise in traditional modelling techniques.

Due to a connected visualization of the volume dataset and the mesh model the user can generate models in such a way that desired regions of the volume are roughly selected. Once generated, these models serve as an initial

vertex position setup for the deformation stage where the model is deformed according to its surface normals and the volume gradient information. The whole deformation process is interactively controllable with different continuous gestures and the iteration continues until either the user stops the deformation or the vertices of the mesh snap to the volume region of interest defined by different criteria (covered in Section 3.4.2).

The resulting triangle mesh is a surface representation of a desired structure of interest within the volume dataset. As a surface representation of a volume is a frequently required task in volume visualization this yields many different possibilities for further usage. At first the model can serve as a cropping or masking shape to cut away unwanted data in the volume. Especially in raw datasets many undesirable structures like metal artifacts or auxiliary materials for the CT or MRT scan interfere with the visualization of the main structures of interest. Also segmentation of volume areas with similar intensity values which can not be separated by transfer function setup can be achieved by this approach in an intuitive and easy manner.

Further possibilities like texturing approaches where the main problem is to find a surface representation for the volume could possibly be based on this approximation of volume regions. Another option is to use the mesh deformation to achieve a surface model out of volume data (e.g., a human head) which can subsequently be imported to traditional modelling environments for further editing.

Altogether this thesis proposes an intuitive and easy-to-use approach for semi-automatically selecting regions of interest in a volume dataset. The user is able to achieve results within minutes due to an undisrupted fast workflow and little expertise in modelling is required.

1.5 Structure

The chapters of this thesis are structured as follows. Chapter 2 covers an overview over related approaches and techniques in the area of volume visualization, gesture/sketch-based interfaces and deformable models.

The main approach of the thesis is proposed and discussed in Chapter 3. Here the different parts of the system forming this work are presented. Chapter 4 deals with the technical details and the framework setup used to implement the system discussed in the previous chapter.

The results of the presented algorithm and implementation are topic of Chapter 5. Images illustrate outcomes of different scenarios, further possibilities and limitations of the algorithm are discussed. Finally the thesis is summarized and concluded in Chapter 6.

Chapter 2

State of the Art

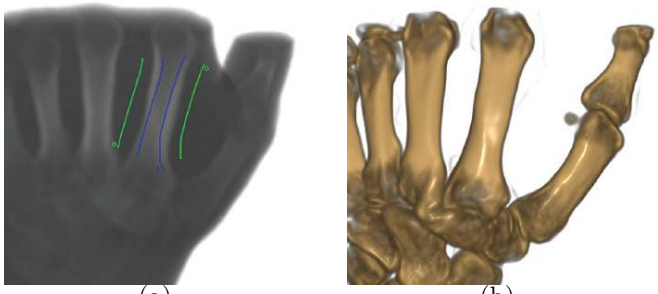
The great tragedy of Science - the slaying of a beautiful hypothesis by an ugly fact.

Thomas H. Huxley

This chapter gives an overview of the research work related to the algorithms and methods used in this thesis. Well-known approaches as well as recent techniques are discussed. As diverse methods from different areas of research are related to the algorithms used in this work, only a short overview on each is given. The interested reader is therefore pointed to latest state-of-the-art reports on the different topics for deeper investigation on the presented areas of research.

2.1 Volume Visualization

The majority of direct volume rendering methods is based on the well-known ray casting algorithm for generating an image of a volume dataset using a transfer function to assign each intensity value a color and opacity. In order to improve the performance of ray casting systems methods like early ray termination, empty space skipping or an octree representation of the volume



(a) (b)

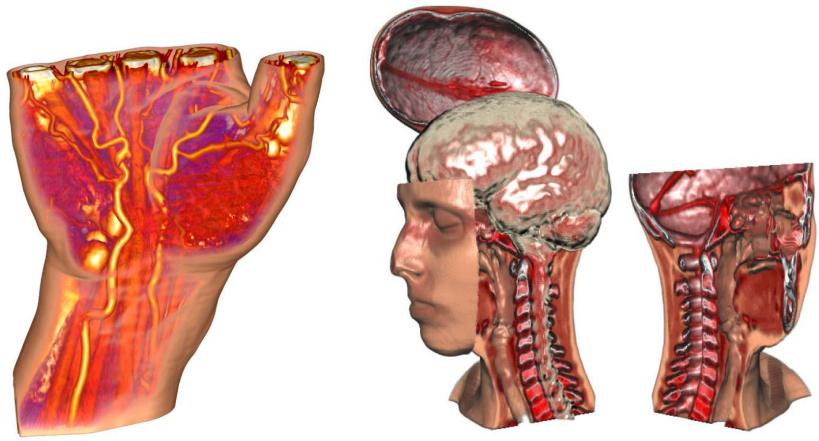
Figure 2.1: Setting up a transfer function with help of strokes on a volume [45].

are used in modern volume visualization systems [36, 30] to make interactive implementations achievable on consumer hardware.

For most algorithms the transfer function setup is achieved with traditional user interface tools allowing the user to build a correlation between intensities and colors. Recently, another approach to directly visualize desired structures has been proposed by Ropinski *et al.* [45] introducing a stroke-based transfer function setup by interpreting user defined strokes on an initial volume. This achieves the generation of an intensity-color mapping overcoming the often cumbersome and time-consuming task of manually setting up the transfer function (see Figure 2.1).

Other approaches employ techniques that completely work without a transfer function setup in order to supply instantaneous visualizations. Representatives of this category are the traditional *Maximum Intensity Projection (MIP)* first proposed by Wallis [57] or more recently *Maximum Intensity Difference Accumulation (MIDA)* [8].

Once a transfer function setup is achieved, diverse approaches to visualize the volume data for different purposes are used. While some systems use advanced illumination effects [18, 46] to obtain a realistic and natural visual impression of the data, others deliberately abstract the visualization to achieve an illustrative emphasis of different volume structures. Latter methods are grouped by the term *Non-Photorealistic Rendering (NPR)* or more general *illustrative techniques*. Here, style-transfer-functions [7] to replace



(a) (b)

Figure 2.2: Focus and context volume rendering: (a) ghosting [4], (b) exploded views [6].

the traditional transfer function with sphere maps or contour emphasizing methods [29] are proposed. A deeper report on different illustrative methods is covered in [3].

In volume visualization the problem of occlusion often occurs. Structures of interest are covered by other data and can not be perceived by the user. Different deformation approaches like cut-away [55] or exploded views [6] but also transparency altering methods like ghosting [4] have been introduced to employ a detailed illustration of desired structures while maintaining the surrounding context of the volume. Figure 2.2 shows examples of these focus and context approaches.

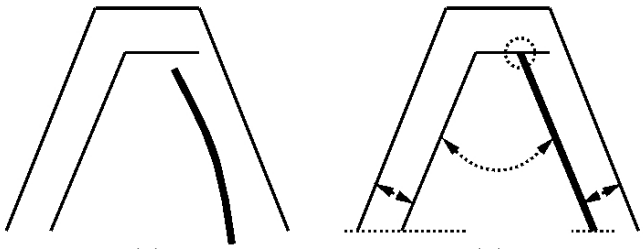


Figure 2.3: Beautification of an input (a) to the constrained stroke (b) [25].

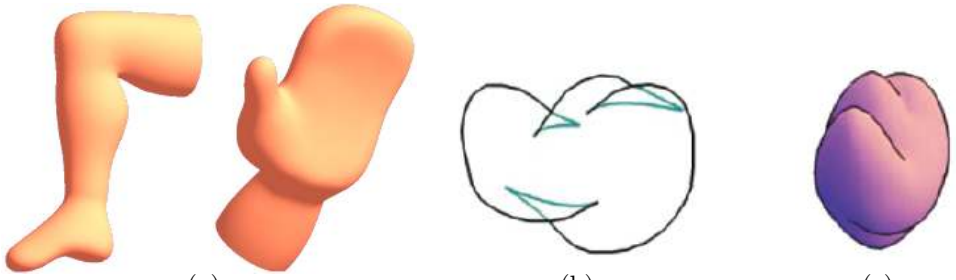
2.2 Sketch-based Modelling

Though first approaches were proposed in 1963 by Sutherland [52], substantial progress on sketch-based modelling has been achieved in the past ten years due to the upswing of touch-sensitive input devices. Many different methods have been proposed for diverse applications, as surveyed in a recent state-of-the-art report by Olsen *et al.* [37, 39]. Common to all presented algorithms is the input of the user by sketching on some kind of working canvas.

The first step is to acquire a raw sketch input from an input device ranging from tablets [49] over virtual reality [14] to haptic devices [21]. On this raw input several preprocessing and interpretation steps are proposed by the authors in order to make the sketch usable for further interpretation. While a simple preprocessing uses resampling of the raw sketch [11], other methods like curve fitting [9] or polyline approximation [26] are employed in sketch-based systems to eliminate erroneously and loosely sampled sketch path points. A rather complex method introduced by Igarashi *et al.* in [25] is called *Beautification*. Here geometric characteristics like parallelism or symmetry are used to resample the input sketch (see Figure 2.3 for an illustration).

After acquiring and filtering the raw input the sketch is ready for interpretation by the different systems. Here a categorization into three different interpretation types can be made [37]:

- Model creation



(a) (b) (c)

Figure 2.4: (a) Results of the smoothed model approach on the teddy system [24]. An input contour (b) is completed to achieve a model creation (c) [27].

- Augmentation of an existing model
- Deformation of an existing model

The majority of the proposals are situated in the area of model creation. Several approaches use the sketch input as a contour of a 3D model for free-form model creation. The contour is automatically inflated to the third dimension and a model is created. One of the first systems in this area and also the best known is called the *Teddy system* by Igarashi *et al.* [26]. This first approach generated quite bumpy and uneven triangulated models out of the input sketches. To overcome this issue research on different model smoothing methods [24] was done based on previous work. Additionally painting onto the generated models was introduced by Igarashi and Cosgrove [23]. Integrating this method with the smooth model approach resulted in the *SmoothTeddy system* [22].

Several other systems propose similar algorithms to generate 3D models out of a sketch with some alteration. One drawback to the teddy system is the restriction to simple input contours (e.g., ones that contain no self-intersections). This is handled by Karpenko and Hughes [27] by applying contour completion on the input sketch to account for hidden structures (see Figure 2.4). Other approaches [54, 11, 49] fit a surface to the supplied contour by solving an optimization problem.

Another form of generating a model out of the sketch is proposed by Zeleznik *et al.* in the *SKETCH* system [60]. Here, instead of completely

(a) (b)

Figure 2.5: The set of input strokes (a) is replaced by models to achieve a scene construction (b) [51].

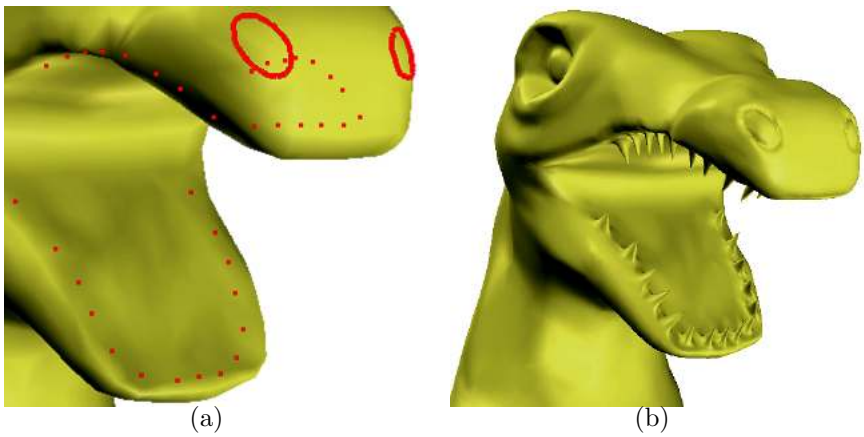
generating a model out of the sketch, the strokes are recognized and replaced by predefined shapes. While this system is based on extrapolation of basic shapes, others use more complex models for the input strokes. Figure 2.5 illustrates such a method called *Magic Canvas* [51] which uses a model database to replace input strokes.

As already stated, sketch-based input is also employed for augmentation or deformation of existing models instead of creating a new object. Olsen *et al.* [38] introduce a system to augment models with a set of different operational strokes. This allows the user to enhance existing models intuitively with detail structures (see Figure 2.6). Other approaches use the strokes as new geometric constraints in surface optimization [33].

For deformation of existing models several operations like cutting [59], bending [26] or over-sketching [34] have been proposed. These systems incorporate easy-to-use methods to edit existing meshes with strokes. Often such deformation approaches are connected to model creation systems as a further toolset to edit a generated object.

Closely related to the topic of sketch-based modelling is the research area of gesture recognition to control applications. The main task is to match an input to an existing template to trigger some kind of operation.

Several approaches to achieve a fast matching have been published, one proposed by Wobbrock *et al.* is called *$1 recognizer* [58]. They present



(a) (b)

Figure 2.6: Sketch-based augmentation of an existing model (a) to achieve modelling of detail structures (b) [38].

an easy-to-implement algorithm using point wise distance calculation from an input to a template after scaling and rotating both to a common value. Hammond and Davis [19] introduce a whole sketching language that is based on topology features between structures (e.g., type of connection). Another more general approach is to employ a neural network algorithm [1, 44] to match predefined input patterns to recorded gestures. This allows a more general way of matching an user input to a previously trained pattern.

Figure 2.7: A surgery simulation approach with deformable models [48].

2.3 Deformable Models

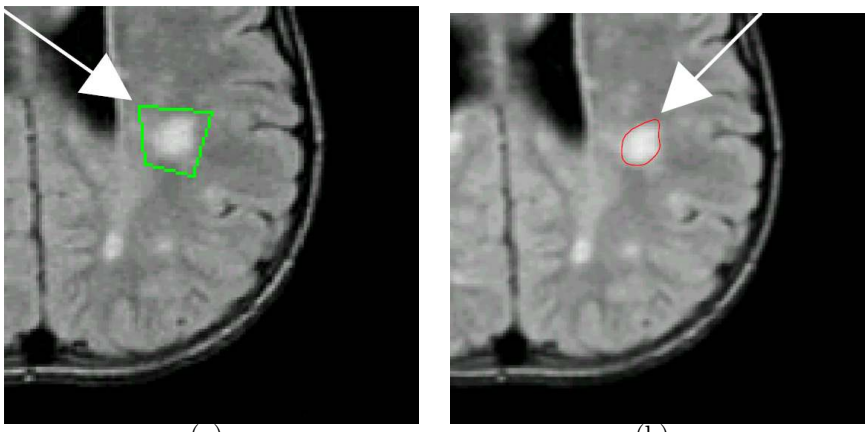
Deformable models have been a topic of research for several years in different areas of computer vision. The terminology used for this field of methods is not consistent within the literature and ranges from *active contours* over *snakes* to *deformable models*, all naming analogous techniques.

Common is the general approach: an input shape (contour or model) is exposed to some kind of force field deforming it while the original input defines restrictions limiting the amount of deformation.

While some approaches use this technology for simulation of elastic objects [53, 15] or even for surgery simulation [48] (see Figure 2.7), many are situated in the area of pattern recognition and image segmentation. Surveys on the different areas of application of deformable models are presented by Gibson and Mirtich [16] and more recently by Moore and Malloy [32].

Originating from the two-dimensional approach of *snakes* [28] many proposals for medical applications have been published. Richens *et al.* [43] and Deraz *et al.* [13] present methods for fitting input contours to specific regions of interest in MRI datasets to achieve medical image segmentation. Figure 2.8 illustrates an example of an active contour fitting to a tumor in a MRI image of a human brain.

Extensions of these methods have been proposed (e.g., Han *et al.* [20]) to fit 3D shapes to 3D datasets analogous to the 2D method. This thesis uses



(a) (b)

Figure 2.8: An initial contour (a) fitting to a tumor in a MRI image of the brain (b) [13].

a simple proof-of-concept deformable model approach following the idea of the snake algorithm to fit 3D models to volume regions. The force field is thereby formed by the gradient information of the volume dataset.

Chapter 3

Sketch-based Modelling for Volume Visualization

Everything should be made as simple as possible, but not one bit simpler.

Albert Einstein

Intuitively selecting specific regions of interest in volumetric datasets is an elaborate task. Standard tools like cropping boxes are often not sufficient to achieve the desired results. Moreover selections based on transfer functions are not only time-consuming but also limited due to the unpredictable nature of the intensity-to-opacity mapping. This chapter details the approach to interactively select volume structures using sketch-based modelling and deformation of models onto volume features of interest.

3.1 Overview

Gesture interfaces allow a quick and easy-to-use workflow. Combined with sketch-based modelling techniques the use of traditional user interface elements is reduced to a minimum, which streamlines the workflow especially for touch-sensitive devices.

The user can interactively generate three-dimensional models out of two-dimensional input sketches in order to roughly select a specific area of interest in a volumetric dataset visualized on the screen. Switching between different operations like model generation or mesh operations is controlled by a gesture recognition system. This way the user can easily switch between operation modes without being disrupted in the main workflow.

Once a rough model enclosing an area in the volumetric dataset is set up, a deformation process approximates the model to the desired volume features. The main factors of influence are the model surface normals and the volume gradient information. Further, different restrictions support the deformation and allow the model to snap to the region of interest. The result is a surface approximation of a volumetric region which can not only be used for selecting, masking or cropping operations on the volume but potentially enable other approaches where a surface approximation is necessary (e.g., a volume texturing approach).

3.1.1 Application Pipeline

The approach described in this thesis is based on a pipeline architecture. The employed techniques can be grouped into three main stages, which are formed by:

1. Gesture recognition
2. Model generation
3. Model deformation

These stages represent the three main techniques used in this thesis, but also describe the typical workflow. While the gesture recognition stage controls the different operations of the application, the model generation stage generates the meshes out of the input sketches. These are used in the model deformation stage where the automatic fitting on volume regions is done. Figure 3.1 illustrates the stages of the application pipeline and the corresponding user inputs.

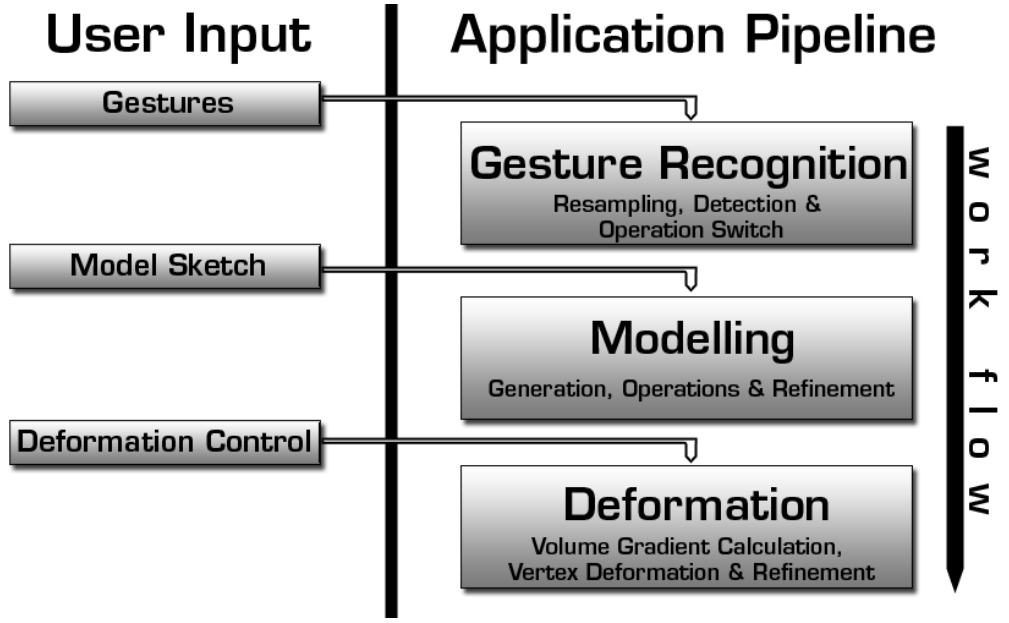


Figure 3.1: The three main stages of the application pipeline with the correlating user input. The black arrow indicates the main workflow direction.

3.1.2 Workflow

The main workflow is as follows: the user generates a 3D model out of a sketch in order to enclose a specific area in the volume dataset. Control of the different operation modes is supplied by the gesture recognition system. When the model fits the desired volume region well enough, the deformation process is started interactively. The model is then deformed until the region of interest is approximated to the needs of the user. The resulting triangle mesh is now ready to be used for selecting, cropping, masking or other operations on the volume.

The further sections of this chapter are structured according to this pipeline and cover the different algorithms and techniques used in each stage.

3.2 Gesture Recognition

Conventional user interface approaches provide the user with control over application states using common interface tools like buttons or sliders. Es-

Figure 3.2: Different input variations match to the same pattern, the correlating operation mode is activated.

pecially in scientific applications, the number of different control elements quickly becomes unmanageable. The act of searching for the desired user interface component disrupts the main workflow and misdirects the user's attention from his goals. Especially when working on a tablet device this interrupted workflow quickly gets distracting and restrains the user from efficiently using the benefits of such an input device.

In an effort to provide fast and convenient means of switching between important application states, gesture interfaces have become increasingly popular in the last years. Particularly for touch-sensitive devices like modern smart phones or tablets this type of interface greatly simplifies the workflow. The main idea of gesture interfaces is to detect simple gestures drawn by the user and to change application states based on a predefined gesture-to-operation mapping. Common examples include the *Opera* browser using the mouse to trigger browser state events or the *Apple iPhone* which is controlled with finger gestures drawn on a touch-sensitive screen.

An important factor in the efficiency of a gesture-based user interface are easy memorable patterns that clearly correlate with the associated operation modes. Figure 3.2 illustrates the switching of the application modes based on a predefined gesture.

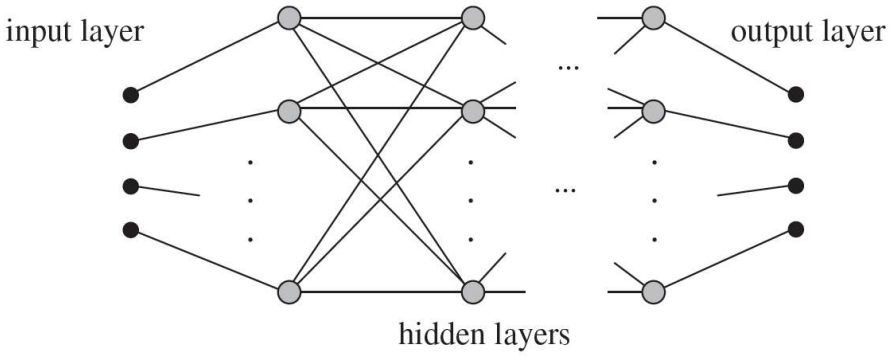


Figure 3.3: Generic setup of a multi-layer neural network [44].

3.2.1 Neural Networks and Pre-trained Patterns

Independent of the input device the main challenge in gesture recognition is accurately matching an input to a predefined pattern. A specific pattern may exhibit considerable variation when drawn by different users or even by the same user. The gesture recognition system needs to compensate these variations (Figure 3.2) and robustly match them to the underlying pattern. This is achieved by a feed-forward neural network in a multi-layer arrangement. The network is trained with the well-known back-propagation algorithm for supervised learning introduced by Rumelhart *et al.* [47].

Neural networks - originally inspired by the examination of the human central nervous system - are used for computational problems of high complexity which are difficult to solve with traditional methods. One major benefit of neural networks is that a given problem does not have to be solved analytically but can be approached with an exemplary learning process. In this learning process different inputs and the desired output are provided to the network which learns to mimic the correlation function.

The basic setup of a feed-forward neural network has neurons arranged in multiple layers formed by one input and one output layer, with multiple hidden layers in between (see Figure 3.3 for an illustration). A signal is propagated through the layers with weighted connections between the neurons. These weighted signals are summed up and handed over to the next layer of neurons through a transfer function. While different transfer functions are

possible, we - as many approaches - use a sigmoid function to generate a degree of nonlinearity between the input and output signals.

To approximate the desired input-output function the weights of the neuron connections must be set up by training the network. Therefore, predefined patterns are submitted to the network in different variations and the network calibrates itself with the back-propagation algorithm which works as follows [44]: an input pattern is propagated through the network layers and the output is compared to a predefined result pattern. The difference between the results is assumed as a network error. Now the error is back-propagated through the network and the connection weights are adjusted in dependence of their influence on the error. This is repeated until the overall error falls below a certain threshold. The threshold thereby controls how well the network will be trained and therefore also has an impact on the duration of the training process. Once the network is trained well enough, the user input can be matched to this previously learned pattern by the network.

Another important factor in generating a robust neural network is the representation of the patterns. To make the patterns independent of orientation the sketch paths are remapped to a sequence of sine and cosine values representing the local direction changes [2]. As the pre-trained patterns are provided in the same format the neural network achieves pattern matching independent of sketch input orientation.

Once the user draws a gesture on the canvas, the matching of the input to a pattern is achieved as follows: while the user is drawing, the system captures the input data and builds a structure holding the raw sketch path information. As the user finishes the gesture, the sketch path is resampled to uniform sample distances in order to achieve good conditions for the gesture recognition. This removes jittering in the sketch path - a typical result of varying drawing speed and unsteadiness while sketching - and thus supports a robust pattern matching. In the next step we calculate the sine/cosine descriptors out of the regulated sketch path which are then propagated through the network and compared to the pre-trained patterns. Thereby, the outcome is not a single match between the sketch path and a single pattern, but a list indicating the probability of a match for each pre-trained pattern. Based

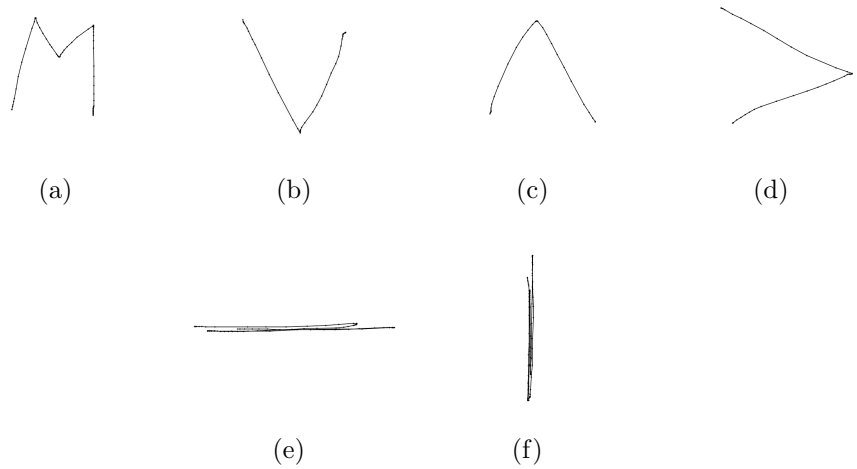


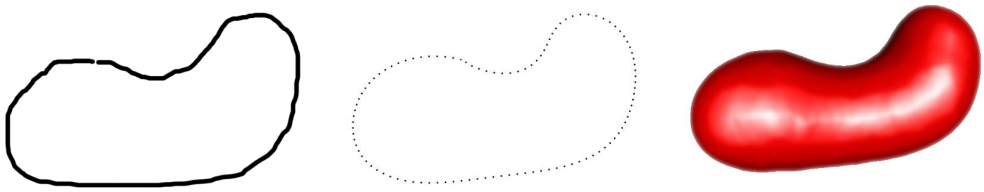
Figure 3.4: Controlling operations with gestures: (a) activating mesh generation mode, (b) activating union operation mode, (c) activating intersection operation mode, (d) activating mesh deformation mode, (e) interactively controlling *Shrink-Wrap* deformation and (f) interactively controlling *Blow-Up* deformation.

on this information, the system can now decide which pattern matches best and trigger the associated application state change. Further the probability values can be used as indicators on the quality of the pattern matches. In order to get robust matching results this probability value is thresholded before match selection. This eliminates unintentional triggers by the user and further inhibits operation activation by erroneously matched patterns.

3.2.2 User Interface Control

Neural networks provide a method to robustly match user gestures to pretrained patterns and switch the application state dependent on the match found. The presented system uses these gestures for two different kinds of user interface operations:

- Switching between modelling operation states
- Interactively controlling the deformation process



(a) (b) (c)

Figure 3.5: Generating a 3D model out of a 2D sketch: (a) basic sketch, (b) pre-filtered sketch, (c) resulting 3D model.

Figure 3.4 covers an overview of the provided operation modes. With the gestures illustrated in Figure 3.4a - d the user controls the application state. This provides quick switching from the mesh generation mode to boolean operations like mesh union or mesh intersection (covered in Section 3.3.4).

Opposed to changing the application state, the gestures from Figure 3.4e and f interactively trigger the model deformation. These two patterns define which deformation type (detailed in Section 3.4) is used and provide the user control on the iterations of the deformation process. The deformation proceeds as long as the user is continuously sketching the deformation gestures. Stopping the sketch-flow therefore also halts the deformation. This allows the user to supervise the progress of the model deformation at any time and - if necessary - to adjust the mesh triangulation or geometry with the different operations.

3.3 Model Generation

Traditional modelling approaches are based on mesh creation with precise manual vertex setup. While the modelling process benefits from expertise in efficient usage of various model generation tools, setting up a complex model is a cumbersome and time-consuming task.

In contrast to traditional modelling techniques, sketch-based modelling allows quick and easy model generation without expertise required. The main goal is to let the user intuitively generate a model out of a 2D sketch. The basic idea is to simulate the classic pen-and-paper sketching process used

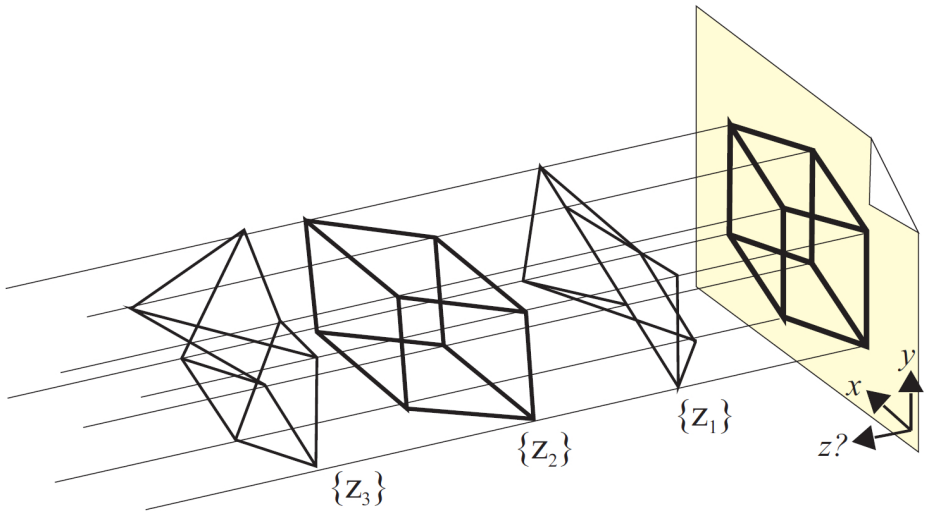


Figure 3.6: An infinite number of objects result in the same 2D projection [37].

by designers and architects to quickly communicate objects and to further generate 3D models out of these sketches. Thus, the sketch is interpreted as a 2D projection of the desired 3D object. Figure 3.5 illustrates this process.

A problem inherent the 2D nature of sketches is their ambiguity. A 2D sketch may correspond to several 3D shapes (Figure 3.6). Hence the process of obtaining a 3D model from the sketch must be bound to assumptions in order to eliminate this ambiguity of the 2D representation.

The main assumption made in the area of sketch-based modelling strongly correlates with the human perception of silhouettes. A silhouette is interpreted as the representation of the simplest possible 3D object, thus a circle is interpreted as the 2D representation of a sphere. Sketch-based modelling approaches also use this simple characteristic when inflating the 2D sketch to a three-dimensional object.

Obviously, modelling with a contour inflating system is limited compared to traditional approaches. Still basic objects are generated accurately with the strong benefit of a simple and fast workflow. In order to further increase the possibilities to generate more complex shapes, we facilitate operations on the created meshes (see Section 3.3.4). These enable the user to achieve

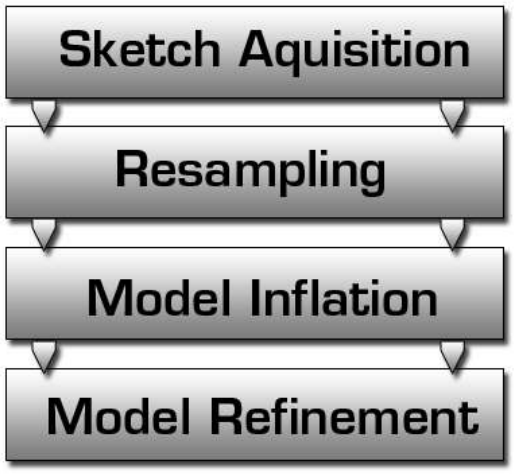


Figure 3.7: The different stages of the sketch-to-model pipeline.

more detailed models by combining basic shapes with boolean operations. In the following sections the different stages from the user input sketch to the 3D model will be discussed (see Figure 3.7 for an overview over the sketch-to-model pipeline).

3.3.1 Sketch Handling

The first step of sketch-based modelling is to obtain an input sketch by the user. In the most basic situation this is achieved by capturing the mouse activity while giving mouse position feedback on the screen. In this case the mouse path simply represents 2D position information stored in screen space coordinates. More sophisticated systems use tablets in order to let the user directly sketch on the screen. This facilitates a more straightforward user experience and provides further control on the sketching workflow itself. Another benefit of such a setup is additional pressure information generated by many tablet systems. The pressure component can be used to control operations or for identification of important parts of the sketch.

An important element in the sketching process is to identify the sketch completion. One way is to let the user control the finalization of a sketch by some input (e.g., a mouse button). Another possibility takes sketch features

into account to decide when the model inflation is started. This is used to restrict the user input to certain constrains such as non-self-intersecting sketches. In the presented approach, a self-intersection in the sketch-path is therefore interpreted as the end of the sketching process which triggers model inflation. This eliminates another necessary user input, further increases the classic pen-and-paper user experience and inhibits self-intersections in the sketching input which leads to problems in model generation.

The raw input sketch frequently contains irregularly placed and misarranged samples due to poor drawing skills or digitization errors of the input device. These main sources of error where first identified by Sezgin and Davis [50]. To compensate for these errors, a filter is applied to the user input in a preprocessing step in order to supply a regular smooth input path to the model inflation stage.

The preprocessing step used in this thesis resamples the captured user input to uniform distances between the sample points. Irregularly spaced points are eliminated and misarranged samples are smoothed out due to interpolation. Furthermore, the resampling stage enables the application to control the complexity of the input path and therefore the complexity and performance of the model inflation process. This is achieved with a reduction of the sample point number which has influence on the triangulation and therefore on the vertex count of the resulting model. Figure 3.5 shows the result of the resampling process on a raw input sketch.

Due to stopping the sketching process at self-intersections and resampling the raw sketch, the final sketch path forms a non-self-intersecting closed contour with regular spaced sample points when handed over to the model inflation stage.

3.3.2 Model Inflation

After acquiring and resampling the user input sketch is used as input for the inflation stage where the transition from the 2D sketch to the 3D model is achieved. For this purpose we use a method proposed by Igarashi *et al.* [26] to generate 3D models out of a non-self-intersecting closed sketch path. This

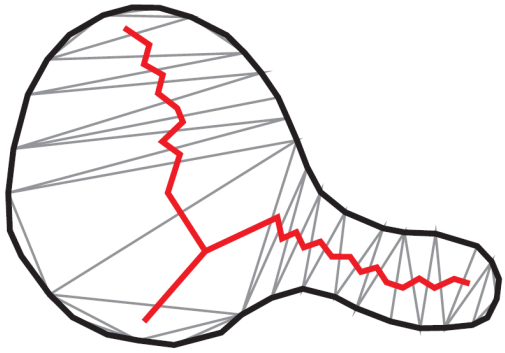


Figure 3.8: Polygonal approximated sketch path with exterior edges in black, gray interior edges and the red chordal axis [37].

well-known approach achieves plausible results by inflating a 2D contour to the third dimension. The algorithm consists of 3 main steps:

1. Triangulation
2. Chordal axis calculation
3. Vertex elevation

In the first step, the model inflation stage calculates a 2D polygonal mesh out of the input sketch which serves as an initial closed polygon with uniform edges, called the *exterior edges* [26] of the mesh. Further, a constrained *Delauney triangulation* [12] adds the *interior edges* building the final 2D polygonal mesh.

The next step of the algorithm extracts the *chordal axis* [42] of the mesh by connecting the midpoints of the interior edges. A setup of such a triangulation is illustrated in Figure 3.8.

Finally, the system inserts and elevates vertices away from the chordal axis according to their distance from the contour. This inflates the 2D polygonal mesh to 3D. The tessellation is thereby chosen according to the needs of mesh complexity. The result is a 3D triangle mesh based on the resampled input sketch of the user.

This method is only practicable when rounded objects are desired and lacks the ability to generate a rectangular model. Since the main usage of

the models in this thesis is to enclose a specific part of a volumetric dataset and the models only act as an initial setup for mesh deformation, this limitation is neglected for the benefit of fast and robust model generation. Also rounded meshes fit most needs in selecting specific volume parts in volume visualization where the majority of datasets are based on biological objects.

3.3.3 Refinement

Depending on the sketch input, the polygonal approximation and the model inflation, the mesh may contain irregularly triangulated parts or rough surface features. To smooth the mesh and to achieve a regular triangulation three different refinement steps on the mesh are executed. Smoothing of the mesh not only improves mesh geometry itself but also surface normal calculation and therefore the visual impression of the whole model. Further, a proper triangulation is beneficial for the mesh deformation stage where a predictable deformation is dependent on uniformly distributed vertex positions.

Vertex Position Smoothing

This step iterates over all vertices and repositions them in dependence of the local neighborhood. First, the center of mass of the neighboring vertices is determined with the following formula where P_i is a neighbor vertex connected by an edge and n is the total count of neighbors at the current vertex.

$$P_{CenterOfMass} = \frac{\sum_{i=1}^{n} (P_{ix}, P_{iy}, P_{iz})}{n}$$

Next, the normalized vector from the current vertex position to the center of mass is calculated and acts as direction vector to reposition the current vertex by a certain factor. This factor is experimentally determined to achieve a smooth model appearance while preserving the overall mesh size. A repositioning by a too large amount would result in shrinking of the whole model as each moved vertex further has influence on its neighbors. Using the factor $\frac{1}{5} \times |P_{CenterOfMass} - P_i|$ to move the vertex along the direction vector achieves

an elimination of irregularities in vertex positioning and preserves the overall mesh size.

Triangulation Coarsening

The first step to generate a regularly triangulated mesh is to eliminate over-tessellated regions of the mesh. Therefore all edges of the mesh are traversed and sorted according to their edge length. In a second step all edges with an edge length smaller than the average are collapsed to a vertex [31]. This compensates errors from the inflation stage where the mesh is - depending on the input sketch - complexly triangulated which results in unintentional surface effects and normal calculation.

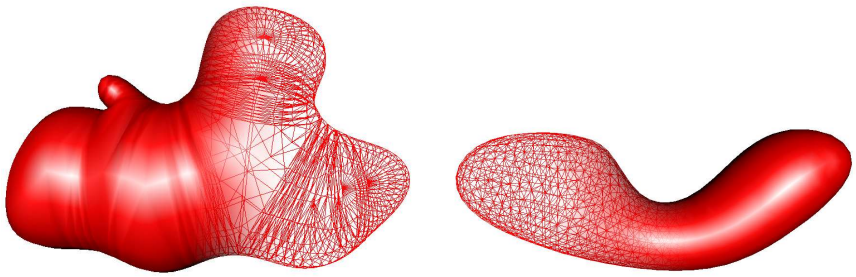
Triangulation Refinement

After repositioning the vertices depending on their local neighborhood and eliminating complexly triangulated regions of the mesh, a triangulation refinement algorithm is applied. This further smooths the mesh surface and regulates regions where triangle edges are too long. To achieve this, the algorithm from the coarsening step is reversed: again all edges are visited and sorted by their length (this time the longest first). Then all edges longer than the average edge length are split with a mid-vertex insertion technique [40].

After applying this last step all inhomogeneous regions of the model are compensated and the mesh is uniformly triangulated. The model now is optimized for the deformation stage where the initial vertex positions and the relationship between vertex neighbors are important factors for predictable results. Figure 3.9 shows two example meshes to illustrate the effect of the refinement stage in this sketch-based modelling system.

3.3.4 Operations with Models

The basic abilities of sketch-based modelling with a single input sketch are limited due to the two-dimensional nature of the input path. To provide the generation of more complex and detailed models further operations are



(a) (b)

Figure 3.9: Purpose of the refining algorithm: (a) a basic mesh output from inflation stage without refinement, (b) another mesh with smoothed positions and regulated triangulation.

required. Many toolsets from traditional modelling techniques could possibly be used for sketch-based systems, yet two basic boolean operations - mesh intersection and mesh union - are employed in this thesis to investigate the usage of such operation tools.

Union and Intersection

With the union operation two meshes are united together at their overlapping regions to form one resulting mesh. In practice, the user selects the union operation mode and draws a sketch connected to an existing model. This generates a new mesh with the already discussed model inflation steps, but hidden to the user. Instead of rendering it to the screen, the new mesh forms the input of the boolean operator *union* on the existing model. The outcome of this operation builds a new mesh and is shown to the user. In the case of the input mesh not overlapping the existing model, the union operation leaves the existing model unchanged.

In contrast to the union operator, the intersection operation only affects regions of the existing model that lie within the new intersection mesh. The intersection operator on meshes thus provides the user the ability to cut away parts of existing models. Figure 3.10 shows the results of the discussed operations.

The flexible combination of these operations allows the generation of com-

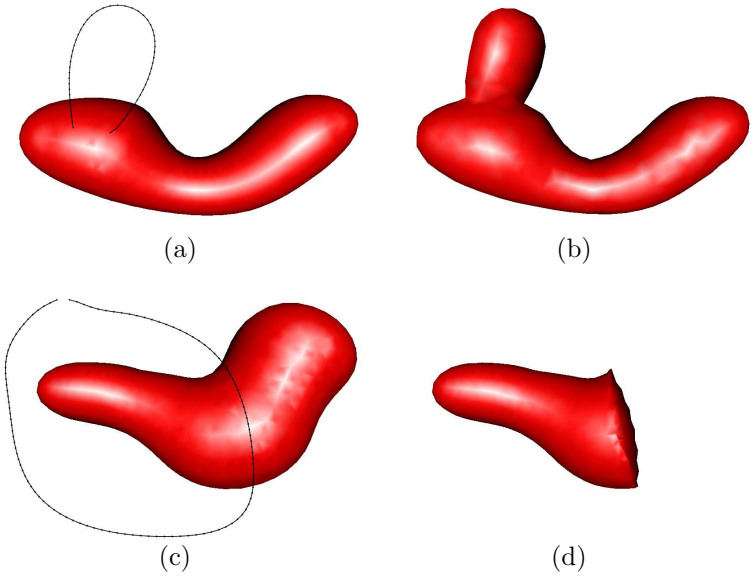


Figure 3.10: Basic operations on meshes: (a) stroke path for union operation with resulting extrusion (b). (c) Intersection stroke path with resulting cut in (d).

plex objects using a simple interface. Thus, a user can approximate complex volume structures with the model and subsequently apply the model deformation algorithm.

3.4 Model Deformation

In this thesis the term *Deformable Model* is used for the 3D derivate of the 2D active contour method called *snakes*. Snakes, first proposed by Kass *et al.* [28], are contour finding algorithms that snap to image contours while preserving a basic predefined shape. These algorithms are widely used in computer vision and provide solutions to diverse problems like image segmentation.

We use a three-dimensional variant of the general snake method to let a predefined mesh model snap to an area of interest in a volumetric dataset. Here, the mesh normals and the volume gradient information are the main factors of influence in the deformation process in a way that the model surface

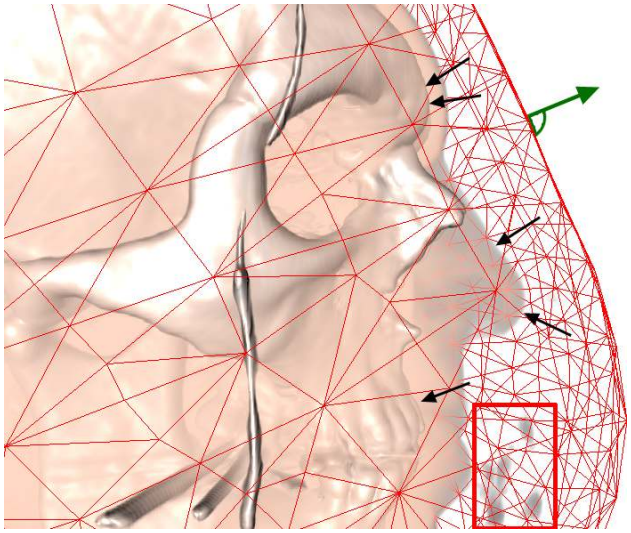


Figure 3.11: Typical initial deformation setup: black arrows indicate the volume gradient pointing in the direction of maximum slope, green arrow illustrates the surface normal of the red triangle mesh wire-frame. The red framed area is formed by erroneous digitization data which shall be skipped in the deformation process.

snaps to volume structures in the local neighborhood.

A common problem in volume visualization are undesired structures in the volume dataset like difference in air intensity or auxiliary tools to stabilize the scanned object. In practice, these structures shall not influence the deformation process in order to allow the user to select the desired region of interest within the volume. To achieve this, we apply geometric restrictions to the deformation process to maintain the basic geometric shape and to smooth out erroneously snapped vertices of the model. As the model keeps its basic shape during the deformation, the algorithm is able to bypass undesirable structures in the dataset.

The basic algorithm for the deformation stage works as follows: for each vertex of the mesh a lookup in the volume is done and the local gradient is calculated. If no gradient information is available (typically in the initial state where the mesh is located in regions where no volume structures are visible), the deformation starts along the surface normal of the mesh model. As soon as the vertex deformation hits areas where gradient is present, the

deformation continues along the gradient direction until a stopping criterion is met.

3.4.1 Starting the Deformation

In the initial state, the deformation setup contains two main elements: the visualized volume dataset and a mesh model generated by an user sketch input. The mesh thereby is generated in such a way that it roughly approximates the region of interest of the volume dataset. Now two deformation modes are available to the user to let the mesh model snap to the volume:

- *Shrink-Wrap* mode
- *Blow-Up* mode

If the mesh encloses the desired area, the user can select the Shrink-Wrap mode to shrink the model onto the volume. On the other hand, if the mesh is generated in such a way that it lies within the volume region, the model can be grown onto the volume with the Blow-Up mode. This distinction between deformation modes is needed as an initial input to the system as most parts of the mesh will be located in a certain distance to the volume and therefore no gradient information is yet available. To inform the system about the mode of deformation, two gesture patterns are included in the gesture recognition interface to control the process (see Figure 3.4). The deformation mode may also be altered during the process itself in order to further approximate the mesh to certain volume regions.

A main factor of influence for the deformation process is the generation of the mesh. A good approximation of the region of interest while sketching the model assists the deformation stage to snap to the desired area. This is correlated with the fact that in the initial state the majority of vertex positions are located in regions where no gradient information is available, hence the surface normal is used to start the deformation. Thus a correctly oriented surface normal - as a result of good volume approximation in the model generation stage - also has an impact on the deformation result.

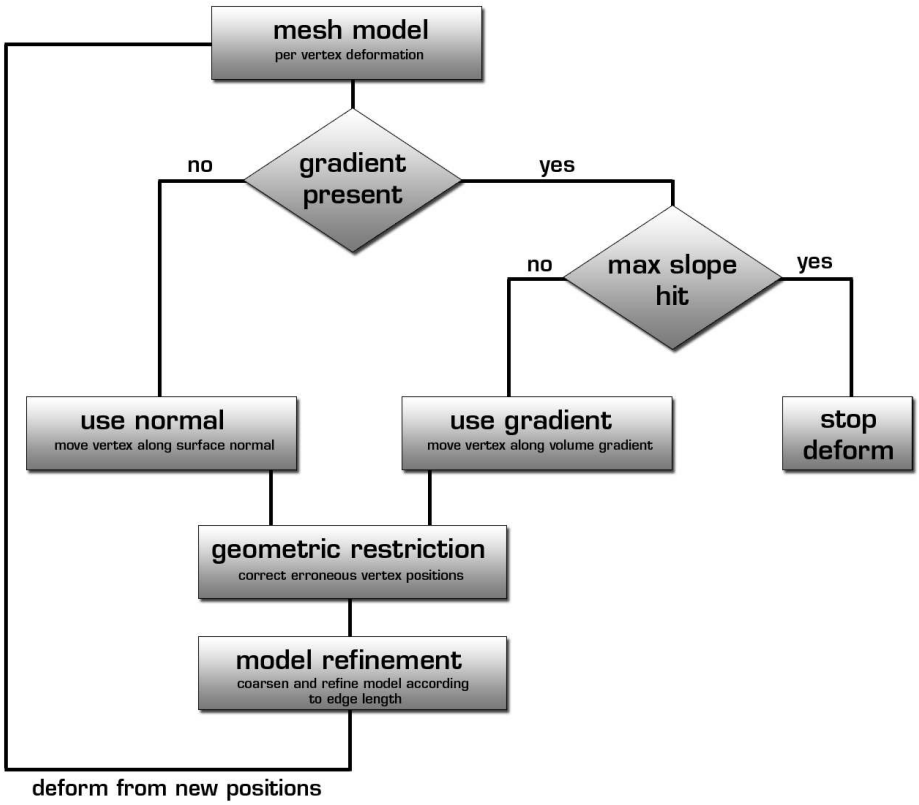


Figure 3.12: The deformation algorithm: this process is executed as long as the user triggers the deformation process by the correlating gesture.

Another important criterion to robust deformation is the regular triangulation of the mesh model. This uniform triangulation is ensured by the model generation stage which regulates the triangulation in the refinement and coarsening steps of model refinement (see Section 3.3.3).

3.4.2 Vertex Deformation

The main deformation step is applied to each vertex of the mesh model. This is repeated as long as the user interactively triggers the deformation by the according gestures or all vertices have snapped to the volume. An overview of the deformation process iterations is illustrated in Figure 3.12.

In a first step, we calculate the correlating volume position for each vertex. This is achieved by transforming the modelview space coordinates of

the mesh model into the volume space defined by the dataset dimensions. The transformations for the mesh model and the volume visualization are linked together. As a next step, the volume gradient information at the current vertex position is calculated, which is achieved by the common central difference method widely used for reconstructing the gradient out of the volume intensity values. The intensities of the volume are weighted with their opacity values defined in the transfer function adjusted by the user. This is necessary to amplify the effect of visible volume information while damping the influence of invisible intensity values. Omitting this weighting would lead to unintuitive model deformation as vertices would snap to invisible volume regions.

After gradient calculation there are two possible outcomes:

1. The vertex lies in range of the visible volume and gradient information is available

2. The vertex is located in *empty* space, thus lying in invisible volume space or outside the volume dataset dimensions.

In the first case, if a gradient is available at the current vertex position, its information is used as a direction vector for the position translation.

In the other case, the vertex position lies outside the volume dataset or no opacity weighted gradient information is present at the current position, therefore the vertex position is translated along the surface normal. At this point the decision on the deformation mode, whether to blow up or shrink the model, decides if the deformation uses the normal directly or in inverted manner. In other words: the translation along the surface normal is basically used to move the vertex position into regions where opacity weighted gradient information is available.

Whereas the initial surface normals may be precomputed, they later need to be recalculated on-the-fly as the model shape may alter from one deformation iteration to another. Thereby, the normal construction works as follows: first we calculate the normal vector for each triangle by taking the cross product of the two direction vectors (defined by the three triangle ver-

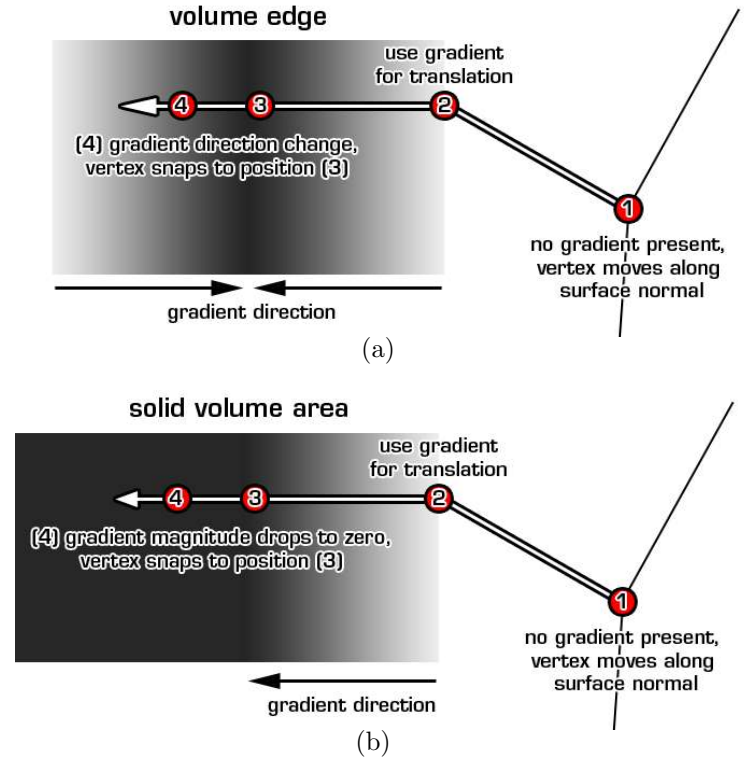


Figure 3.13: Deformation stopping criteria: the gray colored areas represents volume data. The white arrow indicates the iterations of a mesh vertex (red dot) in the deformation process. Criterion 1: vertices snap to volume edges when the gradient direction changes (a). Criterion 2: deformation is stopped for solid volume structures when the magnitude drops off (b).

tices). Secondly, an iteration over all normals smooths them according to their neighbor triangles.

Typically, the initial vertex positions are all located in space with no gradient information present and the normal is used for starting the deformation. Once the model hits on areas where gradient information can be taken into account, the decision whether to move further or to stop the deformation for the current vertex is determined by a stopping criterion.

Stopping Criteria

In this thesis the regions where to stop the deformation are defined by the slope and magnitude of the volume gradient. The model shall snap to edges of changing gradient direction, which typically form the areas of interest in volume visualization. To identify these important edges we calculate two parameters while translating the vertex position along the volume gradient. In each step of the deformation iteration the gradients before and after moving the vertex position are computed. Calculating the dot product of these two gradients results in the angle between them. A change in angle by a certain amount (the presented system uses a value of 90°) forms the first criterion to stop the deformation. This is based on the observation that at highly weighted edges of the volume - defined by the transfer function - the gradient switches direction when stepping through. Figure 3.13a illustrates the described stopping criterion.

This first stopping criterion fails in homogeneous areas when the transfer function is set up in such a way that the structure of interest consists of a solid object with consistent or minimally varying intensity values. In this case the direction of the gradient will not switch direction when stepping over the edge of interest or in fact even can be virtually zero when calculated in homogeneous areas of the volume.

To account for homogeneous volume regions, the deformation stage uses another stopping criterion which is built on changes in gradient magnitude. We calculate the gradient magnitude both before and after the vertex translation in each deformation iteration and compare their values. If the magnitude drops off by a large amount the deformation process is again stopped and the vertex is locked in position. Thereby, the threshold factor triggering the stopping criterion is not absolutely set but compares the ratio of the two magnitudes of one deformation iteration. This is based on the fact that a large drop in magnitude occurs when a vertex is translated from a volume area with high gradient magnitude values - e.g., an edge of a section of interest - to a homogeneous structure (see Figure 3.13b).

Given these two criteria to stop the deformation process, the algorithm

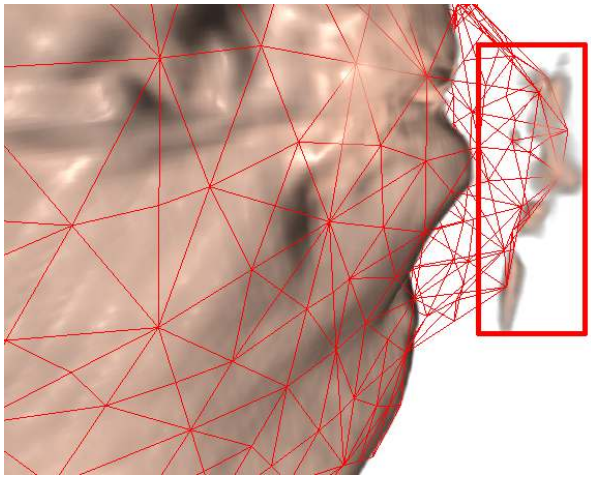


Figure 3.14: Without restricting the geometric setup of neighbors the vertices erroneously snap to regions of no interest (indicated by the red frame).

translates the vertex positions either to an edge in the volume or to the surface of a solid structure. The deformation stops and the vertex snaps to an area of interest in the volume defined by the transfer function setup.

Further important parts of the deformation algorithm are the geometric restriction and model refinement steps. Simply applying the two described stopping criteria would lead to undesirable artifacts in the result mesh after deformation. The vertices would snap to erroneous areas in the volume visualization (e.g., air intensity variances) which are hard to eliminate through transfer function setup (see Figure 3.14 for an example of erroneously snapped vertices). Also, as the model grows or shrinks, the triangulation of the mesh has to be adapted accordingly while deforming the model.

Geometric Restrictions

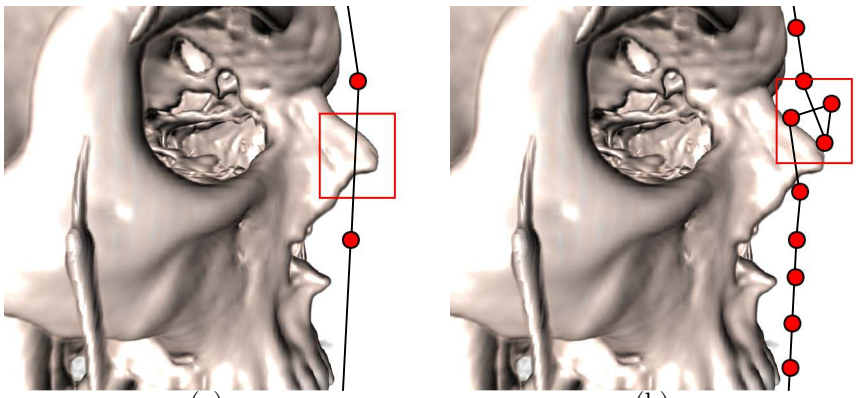
To account for erroneously snapping vertices and to roughly preserve the local shape of the mesh, geometric restrictions are applied in each step of the deformation process. These take the basic shape of the model into account and regulate the vertex positions according to the relative positions of the local vertex neighborhood. Erroneously snapped vertices are detected, repositioned relative to their neighborhood and then ready to be further deformed

by the algorithm until locked in the correct position.

The algorithm for the geometric restriction is based on the vertex position smoothing method used in Section 3.3.3 with some alterations. Analogous to the method described previously, the center of mass defined by the edge-connected vertices is calculated for each current vertex. Next, the distance from the center of mass to the current vertex position is calculated and compared to a threshold. If the distance is beyond the given threshold (two times the average edge length of the mesh), the direction vector from the current position to the center of mass is used to translate the vertex. Distances below the threshold are ignored as the model shall not be smoothed completely, which would lead to shrinking of the whole model when applied consecutively in every deformation iteration. By this restriction, vertices that are too far away from their neighborhood are snapped back to the center of mass. This allows the algorithm to recapture single vertices or small mesh regions from erroneously locked positions. After being repositioned the algorithm further iterates over the given positions until another area of interest is hit without breaking the geometric restrictions.

Using this method still erroneously snapping vertices can occur when larger regions of the mesh are affected. To account for these cases a better approximated initial mesh model or an altered triangulation are possible solutions.

Further geometric restrictions to handle larger erroneously snapping areas of the mesh would be possible, still the used method sufficiently handles most cases where typically only single or few vertices are affected.



(a) (b)

Figure 3.15: Irregular triangulation: (a) vertices (red dots) miss structures of interest (red frame) because of triangulation becoming too coarse. (b) Self-folding effects (red frame) due to edges becoming too short and intersecting each other.

3.4.3 Model Refinement

While the model is modified in the deformation stage the triangulation of the mesh model may become irregular. Two possible problems may arise:

1. In Blow-Up mode: the triangulation may become too coarse to hit structures of interest

2. In Shrink-Wrap mode: the edges may become too short and self-intersect

For deformation, the vertices are initially moved along the model surface normals. In the first case, if the system needs to translate the vertices by a large amount before hitting structures of interest, the whole model grows in extent and the triangulation becomes coarse. As a result of this, vertices, which are the anchor points for the deformation algorithm, may miss features of the structure of interest due to edges becoming too long (see Figure 3.15a for an illustration).

In the second case, when the model is shrunk by a large amount, the edges may become very short. A further movement of the vertices can then result in self-folding effects, because of one edge intersecting another. This

degrades the visual result of the model and also causes problems for the whole deformation process (e.g., surface normal calculation). Figure 3.15b illustrates such a self-folding effect.

To account for the described problems, the mesh triangulation is adjusted while deformation. The mesh refinement and coarsening algorithms already described in Section 3.3.3 iteratively adjust the model and account for a regular triangulation. This backs up the deformation algorithm and inhibits many self-folding effects.

When the user finalizes the deformation process, the model is again refined in a final step. This smooths out last irregularities in vertex positioning and further improves the visual result of the deformed model. Figure 3.16 illustrates different stages of the mesh deformation process.

The presented deformation method achieves a good approximation of volume regions of interest. While further geometric restrictions or stopping criteria would be possible extensions, the current algorithm robustly handles typical volume setups.

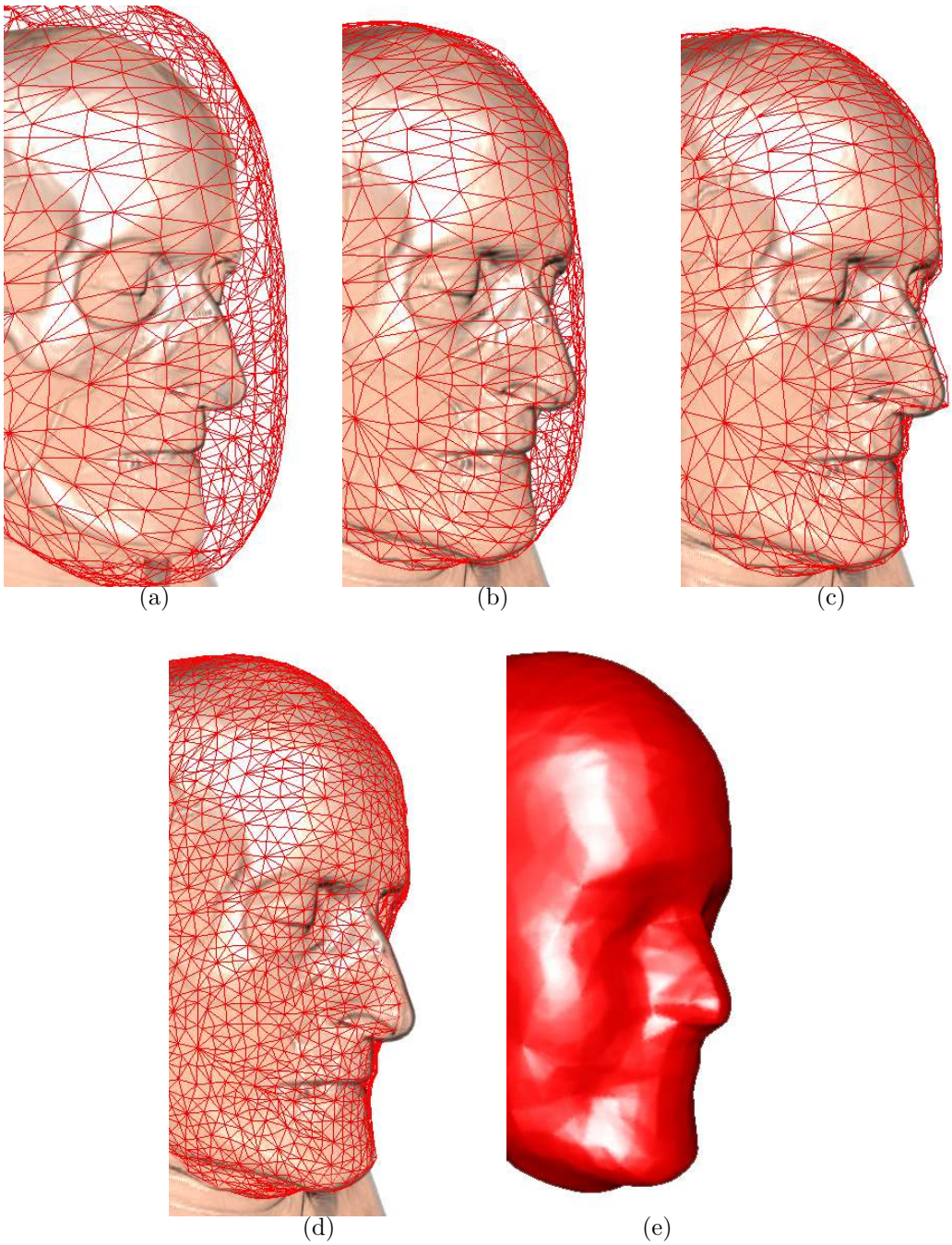


Figure 3.16: Different stages of mesh deformation: (a) initial setup with mesh and volume region, (b) mesh shrinks along inverse surface normals, (c) vertices snap to volume by gradient information, (d) final refinement to smooth model and (e) final deformed model.

Chapter 4

Implementation

Don't reinvent the wheel, just realign it.

Anthony J. D'Angelo

The previous chapter covered the algorithm of the presented approach. In the following sections the main topic will be the implementation details on how this algorithm was realized.

An important aspect for the implementation was to make extensive use of existing libraries and implemented algorithms in order to put the main focus on realizing the interaction of the different techniques and newly introduced methods.

4.1 Main Framework Setup

The main framework we use in this thesis is the *VolumeShop*, an interactive volume visualization system introduced by Bruckner and Gröller [5]. The framework employs a plugin structure to achieve a wide range of possible visualization tools and their combinations.

A major benefit of this system is the already established pool of plug-ins and renderers usable for this thesis. Tasks, such as the rendering of the

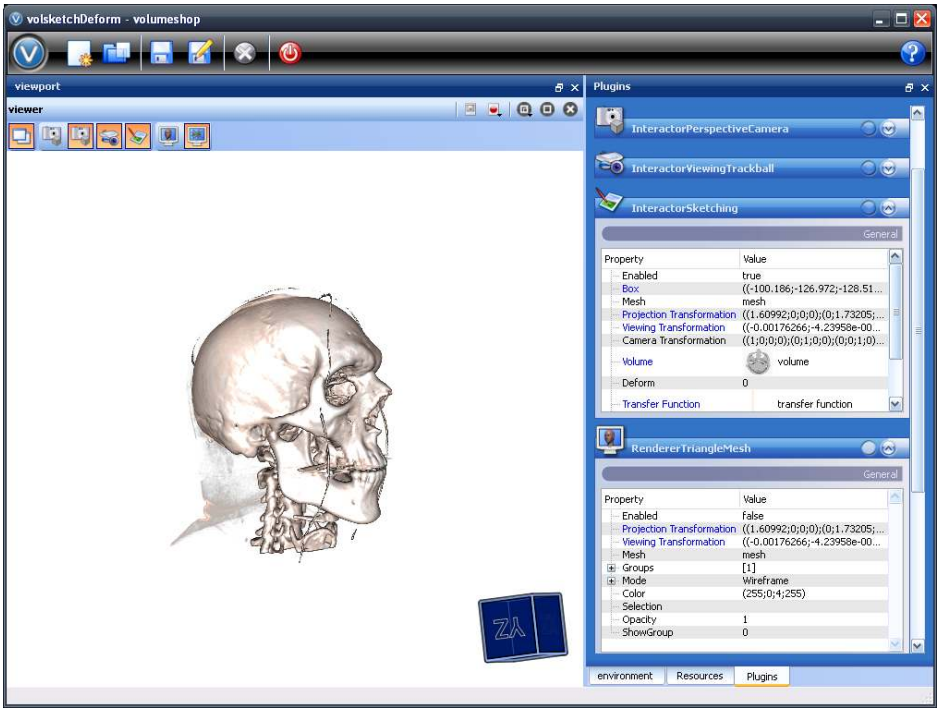


Figure 4.1: The VolumeShop framework with different plugins connected to achieve a volume visualization.

triangle meshes, can directly be achieved using the existing *RendererTriangleMesh* plugin. Also a wide range of volume renderers is available in order to visualize the data not only with a traditional transfer function but in diverse ways like using style transfer functions [7] or realizing maximum intensity projection [57] techniques.

The system is based on *OpenGL/C++* and is also able to handle shader files implemented in *OpenGL Shader Language (GLSL)*. A variety of data structures like vectors and matrices is available and ready-to-use. Also complex geometric structures like a triangle mesh with all its attributes is on-hand and used for this thesis.

To integrate the introduced functionality of the presented system a new plugin named *InteractorSketching* is set up handling the different stages of the discussed algorithm. This new plugin can easily be connected with other existing tools to realize a visualization chain. Sharing of data like the handling the sketch-generated model to the mesh renderer is done via resource

handlers to achieve availability over different plugins. Attributes (e.g., the model view matrix) are linked together to realize a connected transformation of the volume visualization, the sketch paths and the mesh models.

4.2 Gesture Recognition

The implementation of the gesture recognition interface is based on the *ML-Net* implementation of a neural network by Boukreev [2]. Here, a feed-forward neural network with 32 input synapses, 32 hidden neurons and 29 output axons is implemented and trained with the well-known back-propagation method. A pre-trained pattern file providing 29 basic gestures is used to trigger the different user interface operations.

Adaptions in sketch path representation had to be made to make an inclusion of the existing neural network implementation possible. The sketch path, originally captured in x, y value range from 0.0 to 1.0 by VolumeShop is - after simplification - transformed into screen space coordinates ranging from $-\frac{screenwidth}{2}$ to $+\frac{screenwidth}{2}$ and analogous height values. Further, the system converts the path in a series of sine and cosine values required by the gesture recognition stage to match the patterns (see Section 3.2.1).

The different operation states are triggered by the pattern matching system when the mouse button is released. Thereby, a continuous path simplification regulates the input sketch while the user draws on the screen. This eliminates self-intersections and erroneously placed sample points where the drawing speed is very slow and regulates the path where the distance between sample points becomes too large. Figure 4.2 illustrates the uniformly distributed sample points of a sketch path while drawing.

Also a consistent self-intersection test was implemented to check the path while in model generation mode. Once the test encounters a self-intersection, the system starts the model inflation. To correctly trigger the model inflation the location of the self-intersection relative to the whole path is evaluated. This way only an intersection including more than half of the whole path triggers the inflation after eliminating the open ends of the path to achieve a closed regular path for a smooth model generation.

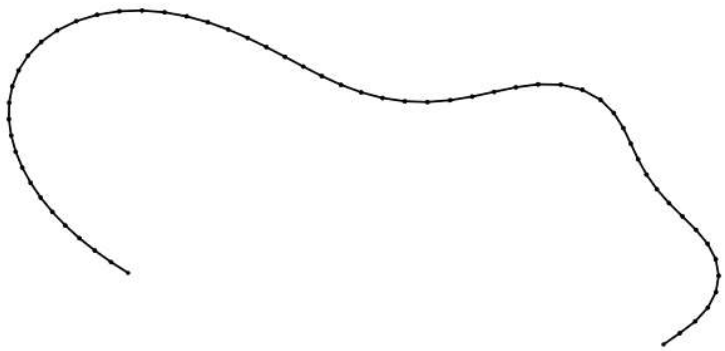


Figure 4.2: The sample point path is continuously regulated and simplified during sketching.

4.3 Model Generation

The model generation stage uses two different libraries to achieve the sketch path inflation. The first is called *OpenTeddy 3D Sketch Modeller* [35] and provides a basic implementation of the teddy algorithm introduced by Igarashi *et al.* [26]. Provided with an input path the OpenTeddy system covers all steps from triangulation to sketch path inflation to generate a three-dimensional model out of the two-dimensional input sketch. As described in the previous section the sketch path is continuously simplified and regulated while being drawn by the user and can therefore be directly used as input for the OpenTeddy system. Another option provided by the system is to alter the tessellation of the resulting mesh model by parameter input.

4.3.1 Mesh Handling

The OpenTeddy system itself is based on the *GNU Triangulated Surface Library (GTS library)* handling the triangle mesh data structure and forming the second external library used in this stage. The author introduces it [41]:

> GTS stands for the GNU Triangulated Surface Library. It is an Open Source Free Software Library intended to provide a set of useful functions to deal with 3D surfaces meshed with interconnected triangles. [...] Careful attention is paid to performance

> related issues as the initial goal of GTS is to provide a simple and efficient library to scientists dealing with 3D computational surface meshes.

The advantage of the GTS library is the variety of algorithms on triangle meshes already implemented and ready-to-use. These are implemented for different operations in the presented algorithm:

Operations with Models

The GTS library provides basic interfaces to apply boolean operations on triangle mesh models. These are initially limited to simple in-or-out operations (e.g., cutting all vertices of one mesh that lie within another mesh) but can be combined in order to achieve the union and intersection operations described in Section 3.3.4. Further operation modes for mesh generation could be implemented using these different combinations.

Vertex Position Smoothing

The vertex position smoothing covered in Section 3.3.3 is implemented using the per-vertex interface provided by GTS. This interface easily allows a loop over all vertices of a triangle mesh model with access to edge-connected neighbors of a current vertex. A major benefit of this GTS function are the integrated data structures that allow fast mesh geometry traversal and therefore account for high performance at this stage.

Triangulation Coarsening and Refinement

For adjusting a mesh model's triangulation the GTS library offers two interfaces for coarsening and refining. Each consists of three functions which can be implemented according to the needs of the application and are passed over to the library as function pointers together with the mesh to be adjusted.

The triangulation adjustment functions consist of:

1. Cost function
2. Stop function
3. Main refine or coarsening function

These functions were implemented as follows: the cost function is called for every edge of a surface and simply calculates the edge length of a current edge. The stop function compares this cost value to a given threshold (see Section 3.3.3) and decides whether to continue or stop the triangulation adjustment process. The main function implements how to split an edge into two edges when refining a model and how to collapse an edge when coarsening a model. For both cases the standard midvertex technique is used: in case of refinement the midvertex of an edge splits it in two new edges while in case of coarsening the edge collapses and is replaced by its midvertex.

Again the GTS interface offered is high-performance as the cost values are calculated in a first step and the edges are ordered accordingly in order to avoid unnecessary iterations over the mesh model.

Other refinement functions like approximation by a spline function would be possible in order to further improve the smooth appearance of the model [24].

4.3.2 Conversion of Mesh Model Data

Two different tool sets are working with the mesh data: while the GTS library is responsible for all operations and adjustments on mesh model data, the VolumeShop triangle mesh renderer is used to render the mesh content to the screen. This renderer also offers interfaces for interaction with the mesh data like easy combination with the viewing transformation or switching the render mode to a wireframe representation.

One problem resulting from the usage of these two different libraries and plugins, is the inconsistent representation of the triangle mesh data. In order to make both technologies usable, conversion functions from a GTS surface

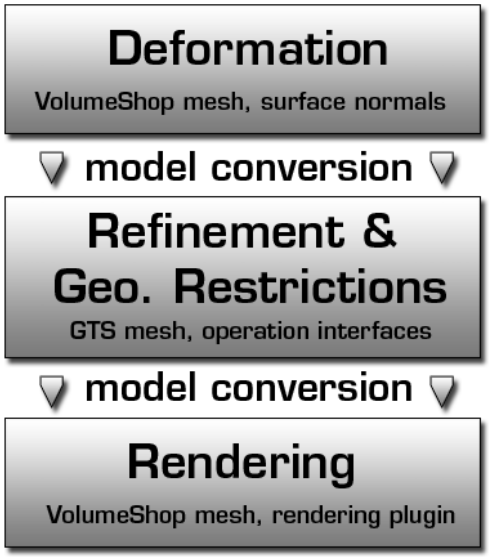


Figure 4.3: The model conversion pipeline in the deformation implementation. Each mesh representation offers special interfaces beneficial to the different tasks.

mesh to a VolumeShop triangle mesh and backwards had to be supplied. These are used in every operation switch where a transition from rendering the content to applying operations on the meshes is needed.

4.4 Model Deformation

The deformation of the mesh models is - in contrast to all other operations on models - built on the VolumeShop triangle mesh representation. This is due to the fact that this representation already includes the surface normal handling which is important for the deformation stage as covered in Section 3.4.2.

In a loop each vertex is handed over to the deformation function. Here the transition from the mesh space to the volume space is done and the volumetric data is accessed with the new coordinates. The resulting intensity values are weighted by the transfer function defined by the user and used to compute the volume gradient using the common central difference method to

reconstruct gradient information. Depending on the availability of gradient information either the normal of the current vertex or the computed gradient is used for vertex deformation. The transfer function in this application setup is also supplied as a plugin by VolumeShop and can easily be accessed as an one dimensional image where the x-axis represents the intensity values and the four color channels indicate the corresponding color and opacity value.

After each deformation iteration the geometric restriction and triangulation adjustment algorithms are executed. They are implemented analogous to the methods for model generation described in the previous section. As we use GTS interfaces for these operations, the mesh data needs to be converted to the GTS representation at this point. After applying the adjustment algorithms, the model is again back-converted to the VolumeShop data structures and rendered by the RendererTriangleMesh plugin. Figure 4.3 illustrates an overview of the model conversions between the different stages.

The system repeats this whole process while the deformation is continuously triggered by the user or until the vertices have snapped to the volume region of interest (defined by gradient direction change or magnitude drop off as described in Section 3.4.2).

Chapter 5

Results

However beautiful the strategy,
you should occasionally look
at the results.

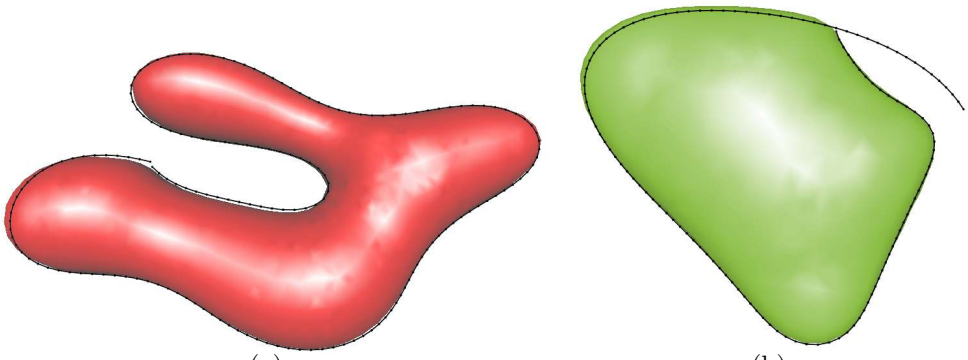
Sir Winston Churchill

In this chapter results of the introduced system will be presented and discussed. First, two cases of the model generation stage are reviewed. The second part of this chapter is dedicated to the deformation of models. The chapter is concluded by a short discussion of performance issues.

5.1 Model Generation

Two main quality criteria for model generation can be observed: first, it is important that the sketch path is well approximated by the model as lacking accuracy degrades the users ability to select a region of interest in the volume dataset. Secondly, it is crucial that the triangulation of the meshes is uniform and regular in order to achieve a robust initial state for the deformation.

Figure 5.1 shows two examples of the correlation between input sketch path and the resulting mesh model. The first example (Figure 5.1a) is triggered by releasing the mouse button while sketching the path, the open ends are connected and the model is inflated.



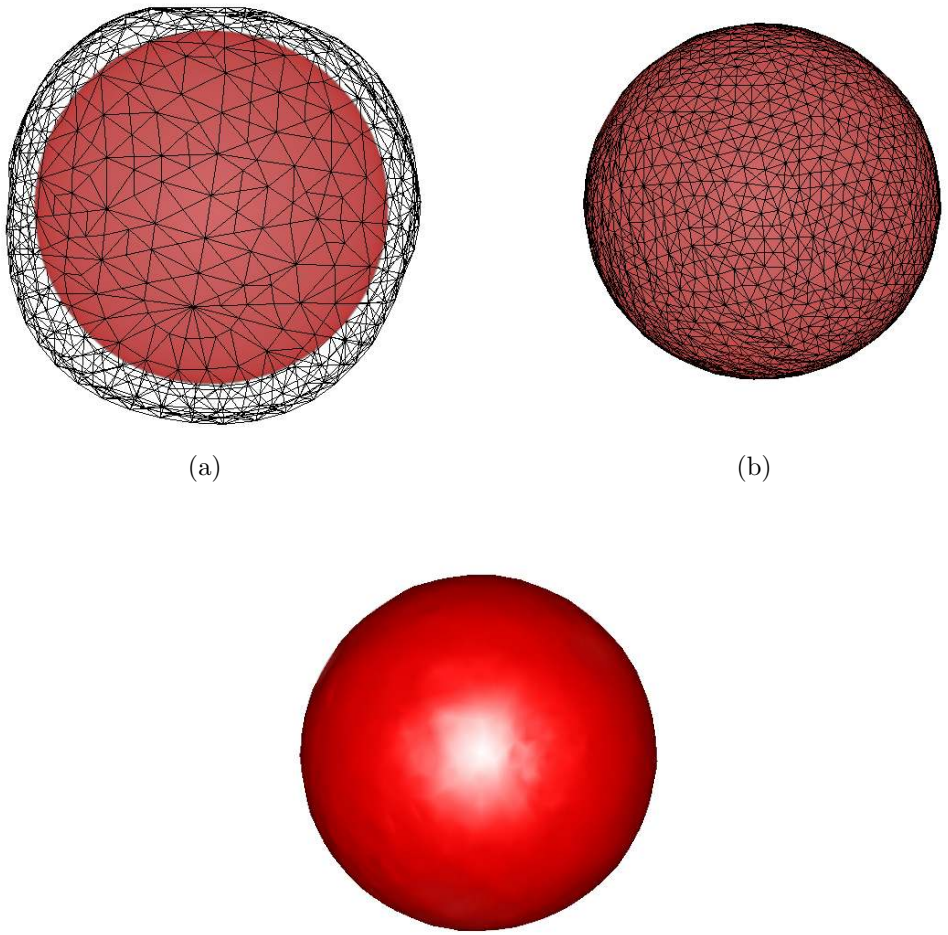
(a) (b)

Figure 5.1: Sketch paths (indicated in black) with the correlating generated mesh model. (a) Model inflation triggered by mouse button release. (b) The model inflation is triggered by self-intersection, the open ends of the sketch path are cut away.

In the second case (Figure 5.1b) the model inflation is triggered by a self-intersection in the path. Here the open ends of the sketch path are cut off previously to model generation in order to achieve a smooth mesh. Due to the triangulation regulation and vertex position smoothing algorithms applied on the mesh the three-dimensional representation may differ by a small amount from the original input sketch, but has smooth visual appearance and regular triangulation.

5.2 Model Deformation

The deformation of a model onto a volume region is the main task of the presented system. This is done in order to create a selection on volume areas that are not achievable via transfer function setup. In an artificial setup the deformation algorithm easily fits the model onto the desired structure because of no disturbing volume features and good gradient information (illustrated in Figure 5.2).



(a) (b)

(c)

Figure 5.2: Test with an artificial sphere dataset. (a) Initial selection, (b) wireframe representation fit to sphere and (c) surface representation of deformed model.

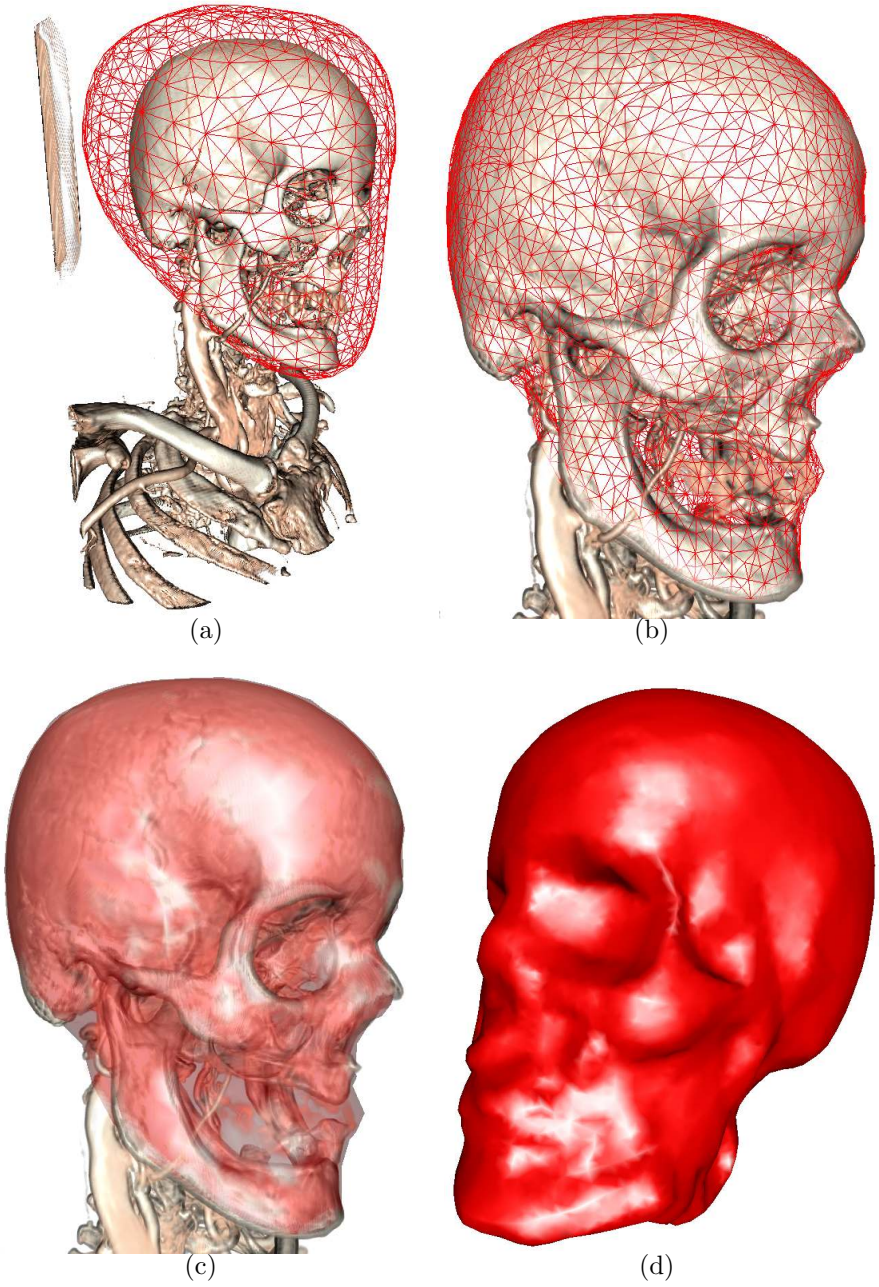


Figure 5.3: Model deformation onto an exposed volume region. (a) Initial model to select area, (b) wireframe model after deformation (opacity of volume altered after deformation to achieve better perception of mesh model), (c) mesh surface representation of volume region, (d) final mesh surface.

Regarding realistic datasets two main cases are observed during the task of selection:

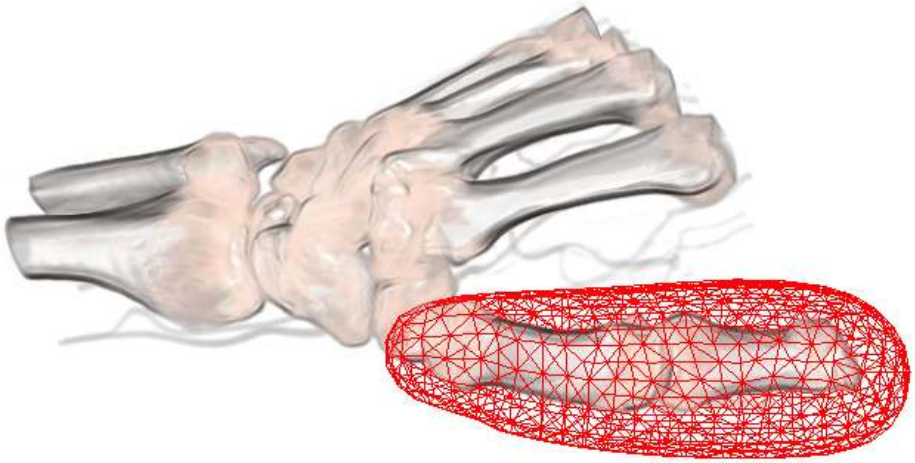
- The region of interest is exposed
- The region of interest lies co-located to other volumetric structures

In the first case, the deformation on the volume is trivial as the initial model sketch can be easily approximated onto the volume region. Also the matching of the region is straight-forward as no surrounding structures interfere with the deformation. Small erroneous volume structures are bypassed due to the geometric restriction algorithm and the mesh vertices snap to the edges of interest. Figures 5.3 and 5.4 illustrate such setups and their results on exposed volume areas. The transfer function in the result images 5.3b and 5.4b were altered after deformation in order to achieve a better visualization of the resulting mesh.

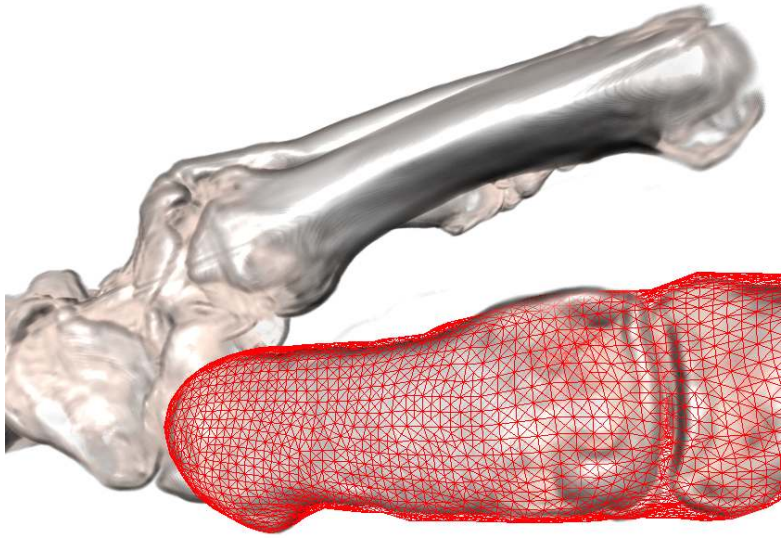
Another example is illustrated in Figure 5.4. Here only one finger shall be selected after deformation. In this example, it is crucial that the initial selection does not overlay with any other solid structures of the volume (e.g., other fingers or the hand bones). The small structures formed by vessels in this example can be bypassed by the geometric repositioning and are therefore to no relevance for creating the initial sketch model.

The second case, where the volume region of interest lies co-located to other volume structures, is more demanding in terms of correctly snapping to the desired area. Especially the initial sketch is crucial to achieve satisfying results. Ideally, the initial model surface should lie in an area within the volume where no structures perturbing the deformation are located. Then the deformation algorithm is able to start in the correct direction and hit the gradient information of the desired region.

An example of the geometric restrictions applied during deformation is illustrated in Figure 5.5. Here, the deformation process is able to bypass the leg structures and let the vertices snap to the body of the stag beetle. Without the geometric repositioning, the mesh vertices located at the beetle's legs would instantly snap to the leg structures.

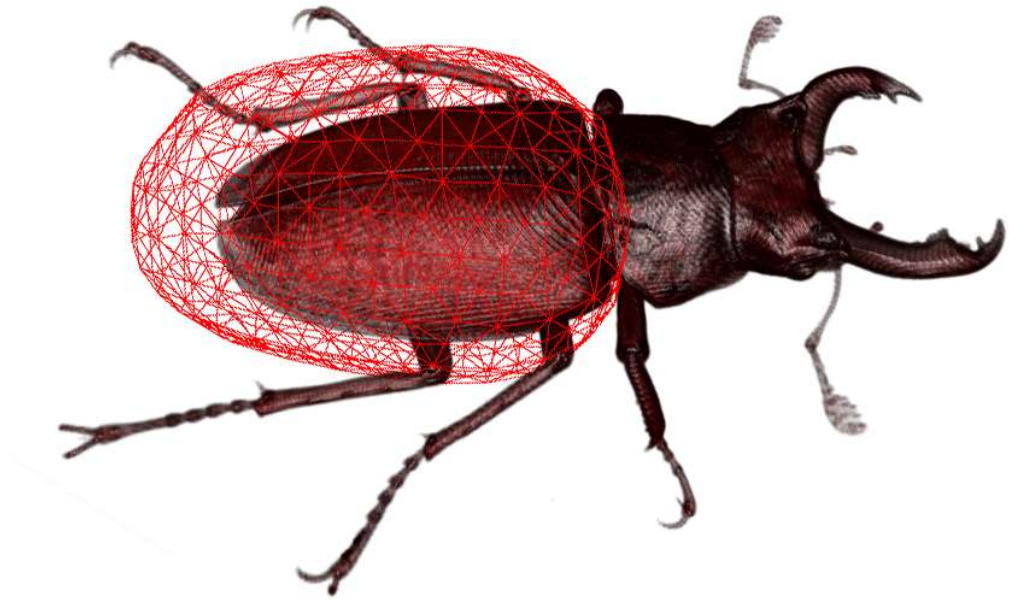


(a)

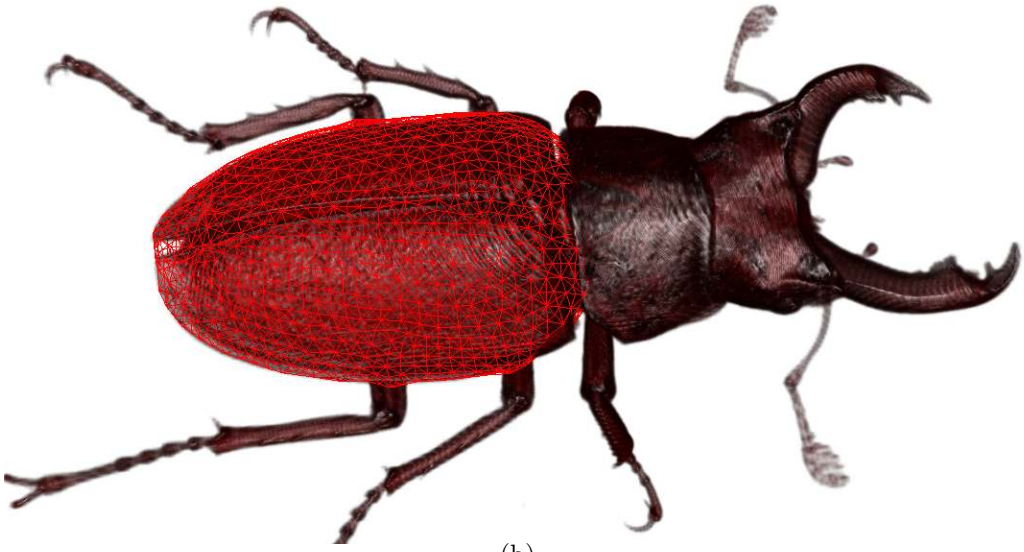


(b)

Figure 5.4: Deformed model on hand bones data set. The initial selection (a) is shrunk and fit to the desired region of the volume (b).



(a)



(b)

Figure 5.5: Surface approximation on the stag beetle dataset. (a) Initial selection, (b) deformed model approximating the volume region.

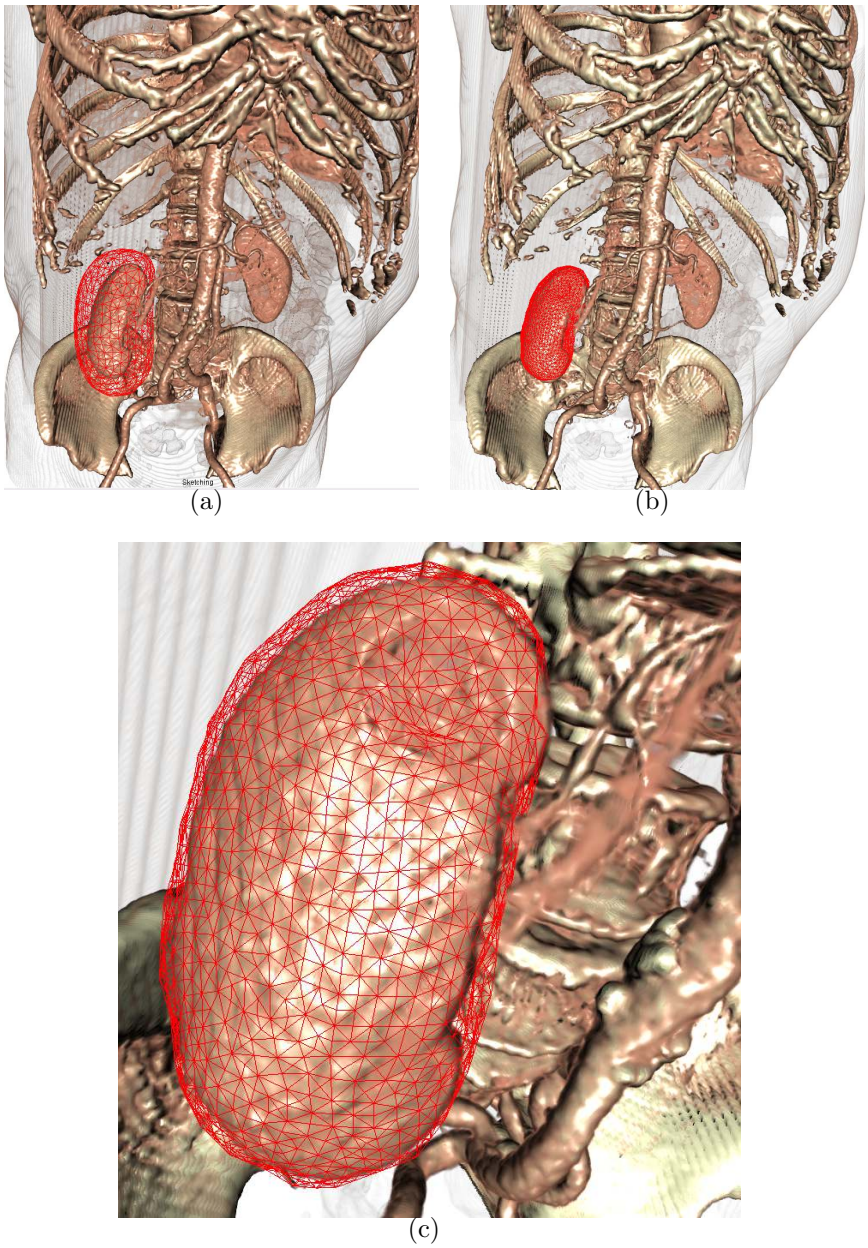


Figure 5.6: Selection of an interior area in a volume dataset. (a) Initial selection, (b) deformed model shrunk onto volume region, (c) close-up view of selection.

Another example is illustrated in Figure 5.6 where the kidney is the structure of interest. This structure is especially hard to initially select with the sketch model as it is very closely located to other bone and vessel structures. In this case, the main effort for the user is to generate a good approximation of the area without hitting too many surrounding structures. If a good approximation is supplied, the algorithm is able to shrink the model to the desired area as shown in Figure 5.6b and c.

Generally it can be stated that the quality of the result models is heavily dependent on the initial user input model, especially when selecting volume regions that are co-located with other structures. Once the user supplies a good initial selection, the deformation algorithm is able to robustly fit the mesh to the desired volume feature.

Structures that are completely surrounded by other solid volume structures and not differentiable using the transfer function are not yet possible to be approximated with the presented method. To account for these cases, further research is necessary to employ other deformation stopping criteria in order to hit completely inlaid volume structures.

5.3 Performance

This thesis does not take performance measures into account, as the frame rates never dropped under interactive values. The system used to achieve the presented results was:

- Processor: Intel Core 2 Duo 3Ghz
- RAM: 2GB DDR2
- Graphics Processor: GeForce 8800 GTS 320MB

Model inflation and operations on models are achieved with an instantly perceived response.

The most performance critical part of the algorithm is the deformation stage due to the different operations that are iteratively applied on each vertex. Therefore, the main performance controlling parameter at this stage

is the vertex count of the mesh models which is around a number of 1000 for a typical setup like shown in Figure 5.3. Currently, a software-only implementation is used. However, a more sophisticated deformation approach may benefit from the additional performance of a graphics-hardware-based implementation.

Chapter 6

Summary

Results! Why, man, I have gotten a lot of results. I know several thousand things that won't work.

Thomas A. Edison

In the following chapter a summary of the thesis is given. Benefits and drawbacks of the presented system are discussed and finally further areas of research on this topic are proposed.

6.1 Sketch-based Modelling for Volume Visualization

A typical problem in volume visualization is the selection of different structures within the dataset that are either co-located or not separable by transfer function setup. Methods like cropping boxes are often unable to fit desired regions and simple masking brushes fail due to the three-dimensional nature of the visualization.

This thesis presents an approach to integrate a gesture-based user interface, sketch-based modelling and semi-automatic model deformation into a

volume visualization framework in order to achieve fast selections of volume regions of interest. A quick and undisrupted workflow is achieved by providing easy-to-memorize gestures for switching between important application operations directly on the working canvas. The sketch-based modelling approach allows the user to generate three-dimensional mesh models within seconds by drawing the correlating 2D silhouette sketch. The model generation works automatically and requires no learning process or modelling expertise by the user. Using different boolean operations, the system allows the generation of complex shapes and provides enough flexibility to roughly approximate most volume structures. Due to the connected view of both volume visualization and sketching/modelling, the user can directly enclose regions of interest in one process while not being distracted from the main working canvas.

Once a basic approximation of the region of interest is achieved, the interactive model deformation fits the mesh onto important structures of the volume. This process can either be stopped by the user to achieve rough selections or automatically lets vertices snap to edges in the volume to fit the desired feature. Using model refinement and geometric restriction algorithms, the result mesh is smooth and directly usable for further applications like masking, segmentation or cutting of the volume dataset.

6.2 Conclusion and Further Work

Overall the combination of sketch-based interface technologies with traditional volume visualization had shown promising results.. The gesture-based operation switching accounts for a fast and natural workflow, especially since it reduces the large amount of controlling options to a few simple gestures. Once a transfer function is set up, the user can devote his attention on the main working canvas during the complete workflow from sketching and editing a model to fitting it to volume structures.

Although a full user study is beyond the scope of this thesis, initial experiments show that the provided interface is perceived fluid and natural by users testing the application. Given a basic knowledge on working with a volume

visualization framework, the provided interface enables an user to achieve results within minutes of work. Further abstraction from traditional user interface tools would be possible and especially useful for transfer function setup. An integration of a stroke-based transfer function setup [45] could be of interest to further replace tools like buttons and sliders which are typically hard to operate with touch-sensitive devices.

For this thesis, tablet devices as well as classic mouse input were used to test the presented interface. Pressure information provided by modern input devices is currently discarded but offers many possibilities for different operation variations like controlling the modelling stage or the deformation process.

The modelling process performs very well and robust but is yet limited to rounded shapes inherent to the used contour inflation approach [26]. Further options like modelling of sharp edges and direct editing of a mesh with gestures would be possible. Nevertheless, the provided operations are yet sufficient to fulfill the task to approximate a desired region within the volume. Incorporating more complex modelling operations would on the one hand alleviate the volume selection but on the other hand also result in more complexity.

Once a good approximation of an area of interest is established, the deformation algorithm is able to fit the model on the volume structure. The user can visually track the whole process using the interactive controlling gestures provided. The presented stopping criteria are designed to suit the needs of typical volume visualization but could fail for other specific tasks. Especially if an initial selection of a structure is not possible because of densely located data, the deformation algorithm is not able to snap to the wanted region due to being misdirected by undesired gradient information. An incorporation of different deformation and stopping criteria could possibly support better results, although best improvements could be achieved by providing diverse modelling and mesh positioning operations. Still, the approach is able to fit well to structures were a good initial model is possible and achieves significant results for many different setups as shown in Chapter 5. Foremost, the system provides the possibility to achieve a selection and surface representation

of a volume structure with an effort of some minutes.

Although - as already mentioned in the introduction - the presented algorithm is most applicable for masking, cropping or segmentation of volumetric datasets, it also yields many possibilities for further application such as texturing or modelling approaches.

Finally, it can be said that the presented system allows a very intuitive and fast workflow in terms of user interaction/model generation and further accomplishes volume structure selections for many typical volume visualization setups.

Chapter 7

Acknowledgements

From now on, ending a sentence with a preposition is something up with which I will not put.

Sir Winston Churchill

I would like to thank my family and my girl friend Sandra, always supporting me during the process of creating this thesis.

Special thanks go to Stefan Bruckner for his excellent supervision, ideas and motivation not only on this thesis but on other projects during my studies. Many thanks to my father and to Gilbert for proofreading the final versions of this thesis.

Further I thank Eduard Gröller, the *Meister*, for making this thesis possible and finally the TU Vienna for providing excellent studying conditions throughout the whole master program.

Bibliography

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

[2] K. Boukreev. Mouse gestures recognition. http://www.codeguru.com, 2001.

[3] S. Bruckner. *Interactive Illustrative Volume Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2008.

[4] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EuroVis 2005*, pages 69–76, 2005.

[5] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *VIS '05: Proceedings of the IEEE Visualization*, pages 671–678, 2005.

[6] S. Bruckner and M. E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 2006.

[7] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[8] S. Bruckner and M. E. Gröller. Instant volume visualization using maximum intensity difference accumulation. *to appear in Computer Graphics Forum (Proceedings of EuroVis 2009)*, 28(3), 2009.

[9] J. J. Cherlin, F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st Spring Conference on Computer Graphics*, pages 137–145, 2005.

[10] M. Contero, F. Naya, J. A. Jorge, and J. Conesa. CIGRO: A minimal instruction set calligraphic interface for sketch-based modeling. In *ICCSA '03: Proceedings of the International Conference on Computational Science and Its Applications*, volume LNCS 2669, pages 549–558, 2003.

[11] B. De Araujo and J. A. Jorge. Blobmaker: Free-form modeling with variational implicit surfaces. In *Comunicaçao ao 12 o Encontro Portugues de Computaçao Grafica*, pages 17–26, October 2003.

[12] B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 7(6):793–800, 1934.

[13] F. Derraz, M. Beladgham, and M. Khelif. Application of active contour models in medical image segmentation. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing*, volume 2, page 675, 2004.

[14] T. Fleisch, G. Brunetti, P. Santos, and A. Stork. Stroke-input methods for immersive styling environments. In *SMI '04: Proceedings of the Shape Modeling International*, pages 275–283, 2004.

[15] N. Galoppo, M. A. Otaduy, P. Mecklenburg, M. Gross, and M. C. Lin. Fast simulation of deformable models in contact using dynamic deformation textures. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 73–82, 2006.

[16] S. F. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories, 1997.

[17] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. GPU-accelerated deep shadow maps for direct volume rendering. In *GH '06: Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 49–52, 2006.

[18] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski. Advanced illumination techniques for GPU volume raycasting. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses*, pages 1–166, 2008.

[19] T. Hammond and R. Davis. LADDER, a sketching language for user interface developers. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 35, 2007.

[20] X. Han, C. Xu, and J. L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):755–768, 2003.

[21] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre. Haptic interfaces and devices. *Sensor Review*, 24:16–29, 2004.

[22] T. Igarashi. SmoothTeddy: Quick 3D modeling and painting. http://www.freshmeat.net/projects/smoothteddy/, 2003.

[23] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 209–216, 2001.

[24] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *I3D '03: Proceedings of the 2003 Symposium on Interactive 3D Graphics*, pages 139–142, 2003.

[25] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: a technique for rapid geometric design. In *UIST '97: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, pages 105–114, 1997.

[26] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 409–416, 1999.

[27] O. A. Karpenko and J. F. Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics*, 25(3):589–598, 2006.

[28] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[29] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *VIS '03: Proceedings of the IEEE Visualization*, page 67, 2003.

[30] J. Kruger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *VIS '03: Proceedings of the IEEE Visualization*, page 38, 2003.

[31] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *VIS '98: Proceedings of the IEEE Visualization*, pages 279–286, 1998.

[32] P. Moore and D. Molloy. A survey of computer-based deformable models. In *IMVIP '07: International Conference on Machine Vision and Image Processing*, pages 55–66, 2007.

[33] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. FiberMesh: designing freeform surfaces with 3D curves. *ACM Transactions on Graphics*, 26(3):41, 2007.

[34] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics*, 24(3):1142–1147, 2005.

[35] A. Ni. OpenTeddy 3D sketch modeler. http://openteddy.sourceforge.net, 2006.

[36] M. Ogata, T. Ohkami, H. C. Lauer, and H. Pfister. A real-time volume rendering architecture using an adaptive resampling scheme for parallel and perspective projections. In *VVS '98: Proceedings of the 1998 IEEE Symposium on Volume Visualization*, pages 31–38, 1998.

[37] L. Olsen, F. Samavati, M. Sousa, and J. Jorge. A taxonomy of modeling techniques using sketch-based interfaces. In *EG'08 STAR: Eurographics 2008 State-of-the-Art Report*, 2008.

[38] L. Olsen, F. Samavati, M. Sousa, and J. A. Jorge. Sketch-based mesh augmentation. In *SBIM '05: Proceedings of 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2005.

[39] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[40] J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics*, 16(4):420–431, 1997.

[41] S. Popinet. GNU triangulated surface library. http://gts.sourceforge.net, 2006.

[42] L. Prasad. Morphological analysis of shapes. *CNLS Newsletter*, 139:1–18, 1997.

[43] D. Richens, N. Rougon, I. Bloch, and E. Mousseaux. Segmentation by deformable contours of MRI sequences of the left ventricle for quantitative analysis. In *IPA '92: International Conference on Image Processing and its Applications*, pages 393–396, 1992.

[44] R. Rojas. *Neural Networks - A Systematic Introduction*. Springer, 1996.

[45] T. Ropinski, J.-S. Praßni, F. Steinicke, and K. H. Hinrichs. Stroke-based transfer function design. In *Proceedings of IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 41–48, 2008.

[46] M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *Proceedings of IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 113–120, 2008.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Neurocomputing: Foundations of Research*, pages 696–699, 1988.

[48] S. Schein and G. Elber. Discontinuous free form deformations. In *PG '04: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pages 227–236, 2004.

[49] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. ShapeShop: sketch-based solid modeling with blobtrees. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 14, 2006.

[50] T. M. Sezgin and R. Davis. Scale-space based feature point detection for digital ink. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 29, 2006.

[51] H. Shin and T. Igarashi. Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. In *GI '07: Proceedings of Graphics Interface 2007*, pages 63–70, 2007.

[52] I. E. Sutherland. Sketch pad a man-machine graphical communication system. In *DAC '64: Proceedings of the SHARE Design Automation Workshop*, pages 6.329–6.346, 1964.

[53] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 205–214, 1987.

[54] G. Turk and J. F. O'Brien. Variational implicit surfaces. Technical report, Georgia Institute of Technology, 1999.

[55] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *VIS '04: Proceedings of the IEEE Visualization*, pages 139–146, 2004.

[56] Wacom. Official wacom europe website. www.wacom-europe.com, 2009.

[57] J. Wallis, T. Miller, C. Lerner, and E. Kleerup. Three-dimensional display in nuclear medicine. *IEEE Transactions on Medical Imaging*, 8(4):297–230, 1989.

[58] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST '07: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, pages 159–168, 2007.

[59] B. Wyvill, K. Foster, P. Jepp, R. Schmidt, M. C. Sousa, and J. A. Jorge. Sketch based construction and rendering of implicit models. In *CAe '05: Proceedings of the First Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 67–74, 2005.

[60] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. SKETCH: an interface for sketching 3D scenes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 19, 2007.

List of Figures