DIPLOMARBEIT

Weighting in Laplacian Mesh Editing

Bauhaus-Universität Weimar
Fakultät Medien

UWE HAHNE
Matrikelnummer: 992268
geboren: 8. Mai 1980 in Kirchheim unter Teck

Verantwortlicher Betreuer: Prof. Dr. Charles A. Wüthrich

Abgabedatum: 10. Mai 2006

Abstract

This diploma thesis has been created in cooperation with DaimlerChrysler. A software has been developed that supports the assembly simulations performed by design engineers. An assembly simulation examines the collision of units. These virtual units are presented as triangular meshes and they are currently solid and undeformable in contrast to real world objects. This software simulates the natural solidness of the units by allowing to deform the virtual objects in a natural manner. To do this, a Laplacian mesh editing scheme has been implemented. Laplacian mesh editing allows to deform 3D objects while their surface details are preserved. Detail preservation is an important feature because it allows the simulation of realistic deformations.

The thesis focuses on examining the effects of different weighting schemes for a Laplacian mesh editing method. This method supports the assembly simulation which is performed during the phase of construction in automotive design. Thus, it has been adapted to the special requirements of such an application by the use of different weights. The deformations are computed by the help of a minimisation method. The present work describes the usage of alternative weights within these minimisation and its effects. In addition, an automatic weighting method has been developed that allows rapid editing tasks.

Zusammenfassung

Diese Diplomarbeit ist in Kooperation mit DaimlerChrysler entstanden. Es wurde eine Software entwickelt, die Montagesimulationen, welche von Konstrukteuren durchgeführt werden, unterstützt. Die Montagesimulation untersucht die Kollision von Bauteilen. Diese virtuellen Bauteile bestehen aus Dreiecksnetzen und sind im Gegensatz zu den echten Bauteilen fest und undeformierbar. Die Software simuliert eine natürliche Festigkeit der Bauteile, indem sie es dem Nutzer ermöglicht die Bauteile in natürlicher Art und Weise zu verformen. Um dies zu verwirklichen, wurde ein auf dem Laplace-Operator basierendes Dreiecksnetzeditierungsprogramm implementiert. Dieses ermöglicht die Deformation von Dreicksnetzen, wobei die Oberflächendetails der Objekte erhalten bleiben. Diese Detailerhaltung ist ein wichtige Eigenschaft, denn sie erlaubt die Simulation realistischer Verformungen.
Diese Arbeit beinhaltet eine Untersuchung der Auswirkungen von Gewichten innerhalb der Methode zur Deformation von Dreicksnetzen, die auf der Verwendung des diskreten Laplace-Operators basiert. Diese Methode soll die Montagesimulationen in der Konstruktionsphase der Fahrzeugentwicklung unterstützen. Die Methode wurde an die gegebenen Anforderungen angepasst, indem verschiedene Gewichtungen eingesetzt wurden. Die Deformationen werden mit Hilfe eines Minimierungsverfahrens ermittelt. Die vorliegende Arbeit beschreibt den Gebrauch unterschiedlicher Gewichtungen innerhalb dieser Minimierung und deren Auswirkungen. Desweiteren wurde eine automatische Gewichtungsmethode entwickelt, die eine besonders schnelle Bearbeitung von Objekten ermöglicht.

Ehrenwörtliche Erklärung

Hiermit versichere ich, Uwe Hahne, geboren am 08.05.1980 in Kirchheim unter Teck, dass ich meine Diplomarbeit mit dem Titel:

Weighting in Laplacian Mesh Editing

selbstständig angefertigt habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weimar, 10. Mai 2006

Uwe Hahne

Acknowledgements

Firstly, I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor Prof. Dr. Charles A. Wüthrich from the Bauhaus University Weimar whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.
I want to thank the DaimlerChrysler Department of Research and Technology in Ulm for giving me permission to commence this thesis in the first instance, to do the necessary research work and to use departmental data. I want to thank Dr. Matthias Buck for supervising and supporting myself during my time in Ulm. My former colleagues from the Virtual Reality Competence Center supported me in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially I am obliged to Ralf Specht, Immo Gürntke, Birgit Meisert and Michael Schulz for helpful discussions and advise.
My godmother Elke Gundel, my father as well as my colleagues and friends Sven Banisch and Alexander Speed looked closely at the final version of the thesis for English style and grammar, correcting both and offering suggestions for improvement.
Additionally, I want to thank Chrystoph Toll, Sebastian Derkau and Benjamin Schmidt for the mental support in the last days.

Contents

List of Figures

List of Tables

1 Introduction

This work has been written in the virtual reality competence center of the DaimlerChrysler research and technology department in Ulm, Germany. It starts with an introduction of the topic and a description of the assignment of this diploma thesis. Followed by a description why this thesis has been written, the programming task according to it is clarified. Then, an explanation is given about the methods that have been chosen. The chapter closes with a transition to the next section that takes a look at the related work of other international scientists.
The assignment of this diploma thesis is to describe the usage and effects of weighting functions or scalar weights in Laplacian mesh editing. The following chapters explain what Laplacian mesh editing means and why and where it is used. In addition the impact of weighting is described in detail. But first, it is necessary to explain the basic idea of this thesis. There are two aspects that have to be considered. On the one hand, there was the need for a software that supports the manufacturing process at DaimlerChrysler. On the other hand this thesis contains a scientific draft about weighting in Laplacian mesh editing. First, the conditions for industrial application are described and then, the scientific aspects are to follow.

1.1 Industrial Application

The research and technology department in Ulm is committed to develop applications which support the manufacturing process of DaimlerChrysler. Many steps in the manufacturing process are assisted by VR[1] applications. One of these steps is the assembly of the vehicles. Looking under the bonnet of a modern car and comparing it with pictures of old ones shows that these days all units are arranged in a highly compact manner. There is nearly no free space between the single units and every part fits perfectly into the engine compartment. By constructing all units with the help of computers it is easy to arrange them compactly. All units exist as virtual models, that can easily be translated and rotated until they are arranged as intended. Reservoirs and containers can be formed in order to fill spaces between solid units. The problem arises by leaving the virtual environment. It is necessary to check whether it is possible for a mechanic to mount and unmount the units in the developed arrangement. This mounting task has to be simulated as well. The design engineers have to examine the feasibility of the arrangement they created. They are supported in their work by a VR software framework called “veo”. With the help of this software, it is possible to check the mounting and unmounting of units in a virtual environment. Based on OpenInventor (see [Wer93] as programming guide), “veo” provides a basic 3D scene viewer and different possibilities to change the appearance of the 3D objects like setting lights and materials. As mentioned, the position and orientation of the objects can be changed. The objects’ data are represented as triangulated meshes. There is an extensive multi user support, so that the software can be used in different working environments like desktop, CAVE and holobench, even at the same time. It builds the so called VR framework, because it provides all necessary basic features for working in a virtual environment. To adapt the software to special tasks, it is extended by a module system. Every additional feature

[1] virtual reality

is developed as a module. This makes the software very flexible and easily extendable.
One of these modules simulates the assembly in car manufacturing. There are two parts, a static collision detection and a physical contact simulation. The collision detection checks whether two units penetrate each other by using a bounding volume hierarchy method. Bounding volume hierarchy method means, that the units are approximated by simple bounding volumes. These bounding volumes can be spheres or boxes. They should have a simple shape in order to simplify the penetration examination. The first bounding volume encases the whole unit. Such a box approximates the shape of the units very fundamentally. Further, the two bounding boxes are examined, whether they penetrate each other. If they do not penetrate, it is obvious that the units do not collide. But an overlapping of these volumes does not stringently result in a collision because the volumes only approximate the true shape. Therefore, the boxes are subdivided in order to approximate the shape more precisely. This subdivision is predefined by a hierarchy. During the algorithmic processing, all objects are examined in pairs, to determine whether their bounding boxes penetrate each other. In case of a penetration, the test is continued in smaller bounding volumes until the true shape level is reached or there is no more penetration. This method is called oriented bounding box approach, if the bounding boxes are oriented in order to keep their volume as small as possible. More details about collision detection and the bounding volume hierarchy method can be found in [KZ03, Loc04].
The contact simulation controls the movement of two colliding objects. Two colliding objects are not just blocked, the system tries to continue the translation given by the user and rotates or shifts the object so that the two objects can pass each other. This simulates a realistic contact behaviour and can be explained with a simple example. If an oblong object like a screwdriver is dropped on a table and the tip of it touches the table first, then the screwdriver is rotated into horizontal position and the remainder of it touches the table. The contact simulation computes the physically correct movement of the screwdriver by comparing the contact points with the centres of gravity of the objects. See the thesis of Matthias Buck [Buc99] for details.
This software is used to check the assembly before the real cars are actually constructed. But the simulation has one negative aspect. All virtual objects are rigid, in contrast to real world objects. The prime example is the water tank for the wipers. This is a large plastic unit that can be pressed by human hands without great effort. A mechanic can easily deform it on mounting, if the gap where it has to be pushed through is too small. But in the simulation, such a task would fail because of the rigidity of the virtual objects. There are several ideas to avoid this problem. The first and most straightforward idea is to work with deformable objects in the VR environment and adopt the collision detection to these deformable objects. Collision detection of deformable objects is currently mainly used for animations [JP04, HTK+04], that neither require real-time response nor high precision. The state of the art in collision detection of deformable objects can be found in [TKH+05]. Teschner et al. present topical applications from cloth and surgery simulation and show the differences to collision detection algorithms for solid objects. Cloth and surgery simulation are beside animation the two main applications for collision detection of deformable objects. All applications are examined according to collisions and self collisions, pre-processing, collision information and performance. In cloth simulation, it is absolutely necessary to consider self collisions, because clothing wrinkles and is stitched together. In order to simulate the behaviour of cloth in real environments, physics have to be considered. Due to this fact, the costs of such a task rise extremely, because even parts of the same object have to be examined whether they collide. In surgery simulation,

the contact of surgery tools with deformable tissue and to that, simulating tasks like cutting have to be developed. Additional information about the stiffness and the maximal penetration depth is necessary in order to create realistic deformations. This information has to be defined and additionally be stored for each object. In a pre-processing step, the data can be arranged in order to speed up the system. Here, bounding volume hierarchies can be constructed. However, it is difficult to construct bounding volumes for deformable objects, because the shape changes dynamically. Therefore, it is necessary to construct hierarchies which can be updated very fast. Alternatively to the bounding volume hierarchies, there are several other methods like stochastical ones which only approximate the deformation instead of calculating it exactly. In addition, there are approaches using distance fields. A distance field $D : \mathbb{R}^3 \rightarrow \mathbb{R}$ defines surfaces as a zero level set $S = \{p \| D(p) = 0\}$. The surface is defined by a number of points p that have the distance $D(p) = 0$. This definition requires a huge amount of data points, since the areas surrounding the surface have to be stored, too. In order to decrease the number of discrete points only a small band around the shape is considered and the fields are stored in data structures like octrees or binary space partition trees. Using distance fields makes it easy to recognize collisions, but there is great effort necessary to update the distance fields after deformations. For every application, there are many parameters that can be changed in order to enhance the system. There is no nostrum for all applications, and the number of parameters grows with every new feature. This rapid increase of complexity arrests the development of an useful system for assembly simulations.

Regarding the assembly simulation, a change of the bounding volume hierarchy algorithm in the existing method would be necessary and therefore, the whole contact simulation module has to be redone. This would exceed the extend of a diploma thesis. There is a doctoral thesis in progress at DaimlerChrysler about the assembly of flexible wires and cables. It was determined that such a task would be too extensive, because every object would in addition need properties about its stiffness that do not exist yet. This is why there was no way to develop a system which is able to decide, whether a mounting task can definitely be done.

But the user knows intuitively the rigidness of the units he reviews for mounting. Out if it, an approach has come into existence that gives the user the possibility to deform these objects like a mechanic can press the water tank. The user can easily estimate the flexibility of a unit and keep full control over the assembly process, without the need to change the objects in a CAD environment, he can deform them within the VR system. Such an enhancement would save a lot of time, as the deformation functions are developed in order to serve the purpose of simple and rapid mesh editing. In order to enable a rapid prototyping, a STL export has been realised. The STL format is an ASCII file mainly used in manufacturing. It contains a simple list of all triangles in Cartesian coordinates. Most rapid prototyping machines use it as an input, and all CAD programmes can read it, too. This way, the deformed object can be exported and be sent back to the CAD environment, or transferred directly to a machine that creates a prototype which can be checked by the mechanic for mounting. According to this task, the focus was directed on a simple modelling approach that fulfils the conditions from Table 1.1. The user should be able to change the objects on vertex level, but at the same time should have the possibility to deform large surface areas. In addition, the user should be able to modify the surface without a long period of vocational adjustment. This is the most challenging part, because

Modelling conditions
High precision
Intuitive user interaction
Fine and smooth deformations
Flexibility
Full control
Measurement feedback
Fair surface design

Table 1.1: Modelling conditions

> "a complex mathematical framework has to be hidden behind an intuitive modelling metaphor." From [BK04a, p.1].

Here, the thesis' scientific work truly begins.

1.2 Scientific Application

The first task was to find out which existing modelling approaches fulfil the listed conditions. Looking into history of deformation tools leads directly to the free form deformation idea. Based on this idea, which is described in detail in Section 2.2, there are several current developments that may comply with the requirements. The basic free form deformation is a volume-based deformation approach. This means that an approximating simple volume is generated, which can be deformed by the user. The deformation of this volume is interpolated and adapted to the detailed shape. This approach does not fulfil the condition of direct manipulation. Due to this fact a direct vertex manipulation method has been developed. In general, surfaces are represented as a set of points or vertices which can be connected to a mesh. This method allows to transform single vertices by drag and drop, while the connections between the vertices are retained. However, this approach lacks the possibility of global deformations. It should be possible to easily deform large regions of the surface. Therefore, a region selection mechanism has been developed. The performance of editing the selected region depends on the number of surface points in the region. Hence, it is necessary to approach the need of transforming each point discretely. Combining this need for regional deformations and the aspect of exact vertex editing, there are several alternative approaches available. The multiresolutional shape editing approach bases on the subdivision of the surface. In this approach the surface mesh is subdivided into a coarser one, which contains less points. Through this reducing of the amount of vertices, the shape can be edited interactively. However, this approach does not consider the connection of the vertices during the subdivision and hence, artefacts can appear if strong details like sharp features are deformed. The models of the units for the assembly simulation contain such sharp features. Therefore, a method called Laplacian mesh editing was chosen. It is a *detail preserving* modelling approach that complies with the intended properties. It is based on the use of differential coordinates which are an implicit description of the surface. This implicit description contains

information about the details of the surface. The detail of a surface can be defined as the difference of the surface and a smoothed version of the surface. After transforming a part of the shape, the complete deformed shape is reconstructed using the detail information from the differential coordinates. This reconstruction is realised as a minimization that can be controlled by weighting functions.
This approach was implemented and refined focusing on the weighting possibilities. The idea was to adopt and extend the Laplacian mesh editing scheme, in such a way that both fine and large-scale deformations are possible. Therefore, different weighting schemes are examined and constructed. Laplacian mesh editing is an approach for *fair* surface design. This means, the deformation should always keep the mesh as fair as possible. Fair surfaces are surfaces that do not contain holes, they are pleasing to the eye or mind especially because of fresh, charming, or flawless quality. To achieve this, several constraints and boundaries have to be added to the implementation. This approach is also called boundary constraint modelling. The boundaries are presented as an energy function of the shape. The energy is zero if the shape is unchanged. Deforming the shape leads to an increase of the energy, that is comparable with a tension. Minimizing the energy function avoids such tensions and keeps the surface fair.
Another aspect to be kept in mind is that there was no need for ad-initio modelling. Ad-initio modelling is the creation of a model from the beginning. In general, the user can build a simple shape and manipulate this shape until it is a model. This is comparable to a sculptor, who starts with a block of stone or wood. In this case, there is always a complex 3D shape available that needs to be deformed. These shapes are created by laser scanning devices or, like in the industrial application case, they derive from CAD data. With the use of Laplacian mesh editing it is not possible to deform a basic shape like a sphere in order to receive a complex structure because the tessellation structure is retained. There are several approaches that enable ad-initio modelling, but for this reason they are not considered in this thesis. In contrast, the Laplacian mesh editing scheme bases on the preservation of details through an implicit description of the connection of the vertices. Not their absolute positions are decisive, but their differential coordinates. With the help of the discrete Laplacian operator of a mesh, which is used for the differential representation of the mesh, an energy function is built. Minimizing this function results in a fair surface editing tool. A precise definition of differential coordinates can be found in Section 3.3, and a detailed explanation of how the Laplacian mesh editing scheme is build up follows in Section 3.6. The Laplacian mesh editing can be controlled by weights. The possibilities and effects of weighting are examined in this thesis. Firstly, an intensive weighting scheme is presented, that gives the user full control over the transitions between deformed and undeformed regions of the surface. Further on, it can also relieve the user from this controlling task by automated weighting. The automatic weighting enables a fast and simple editing of large areas.

1.3 Overview

The remainder of this thesis is built up as follows. At first an overview about the related work is given and the state of the art in Laplacian mesh editing is presented. In this chapter, all different approaches in shape modelling are considered, but the focus is laid on systems that are similar as the one that has been developed according to this thesis. After that, a chapter explaining the mathematical background follows.

It is suggestive to define some mathematical expressions and repeat some basics in order to support the understanding of the whole system. In Chapter 4, the implemented methods are described. Reasons are given, why solutions have been chosen and the purposes of each method is described in detail. Then a chapter follows containing explanations of effects and meanings of weighting in Laplacian mesh editing. This chapter depicts the impact of several weights and presents their intention. The thesis closes with a conclusion and an outlook on further work.

2 Related Work

2.1 Introduction

In the following section, a review of existing systems and approaches is given. The implemented system presented in this thesis unites several different methods for mesh deformation and so do other approaches as well. Therefore, this chapter is arranged according to these different methods and for this reason, some works will be mentioned multiple times. As stated in Chapter 1, there is a need for an intuitive shape editing tool in order to increase the flexibility of the interaction, so that the design engineers can optimise the assembly. At first, the free form deformation (FFD) technology is regarded. Being one of the oldest methods in 3D mesh manipulation, it has been further developed during the last two decades and all recent frameworks base on the ideas of this approach. The next part discusses several drafts using the multiresolutional approach in shape editing. Further, other existing Laplacian mesh editing frameworks are presented. After dealing with the drawbacks of Laplacian mesh editing and with some other approaches that solve or elude these problems, an overview on similar frameworks for shape editing in general will follow. Within all the upcoming paragraphs, the different approaches are ordered chronologically according to their historic emersions, so that the reader is introduced to the development from one development stage to another.

2.2 Free Form Deformation

In this section the oldest approach of shape manipulation is introduced. It all starts with "Global and local deformations of solid primitives", presented by Alan Barr [Bar84]. When regarding global transformations, such as scaling and rotating, the existing transformation matrix does not change. Barr introduces the application of global transformations as local deformations of simple objects. Therefore, he modified the transformation matrix for each vertex. The following explains this in detail. Every deformation is expressed as a transformation

$$(X, Y, Z) = F(x, y, z),$$

where x, y and z are the vertex coordinates of the undeformed object and X, Y, Z are the coordinates of the deformed object. For example, a scaling operation is expressed by

$$(X, Y, Z) = (s_x x, s_y y, s_z z),$$

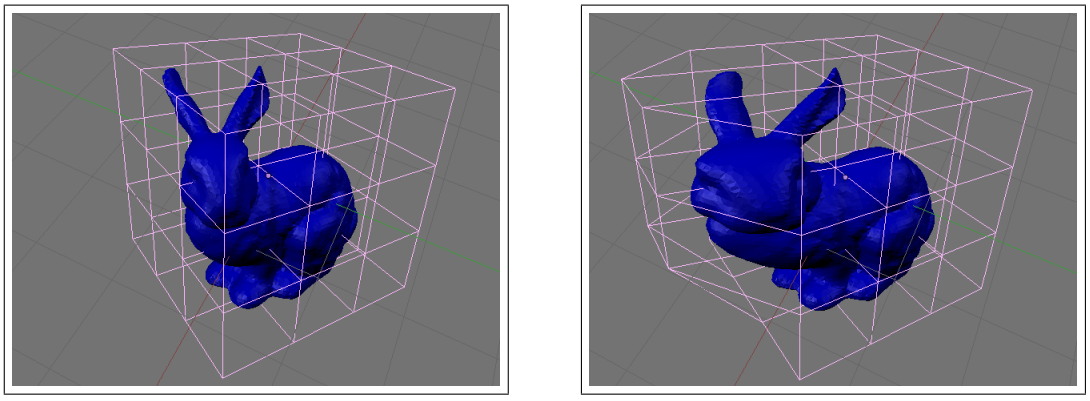
with s_x, s_y and s_z as the scaling coefficients for the three dimensions. In order to realise a tapering along the z-axis deformation the scaling along the x- and y-axes is made dependent of the z value. Thus, we can write

$$(X, Y, Z) = (rx, ry, z),$$

where $r = f(z)$ is a linear or non-linear function. In the same way, Barr introduced deformations such as twisting, bending and the shown tapering. In addition, a Jacobian matrix exists for each of these deformations containing the first derivatives of the deformation function. Barr found out that there is a coherence

of local and global deformation through the Jacobian matrix and thus, the normals can be calculated easily. Barr found formulas for the mentioned simple deformations. All these deformations can be combined hierarchically to complex operations. By multiplying the transformation matrices of each vertex, complex shapes can be constructed in an intuitive manner, at least by mathematicians, because Barr's implementation lacks a graphical user interaction. The user must define the deformations mathematically. In order to enhance the user interaction, this approach was refined by Karen Singh and Eugene Fiume in 1998 [SF98]. Here, the user is able to draw the curves directly onto the complex surface, like winding a wire around the object. These wires can now be transformed and the underlying surface is deformed respectively. In addition, the user controls the range of the wire deformation, in order to get smooth or sharp features depending on his intention. The advantage of this system is that interaction is highly intuitive. It is a natural working behaviour to pull and push parts of the object like in clay modelling. The immediate visual feedback of the deformation makes the modelling task even more realistic.

In 1986, the free form deformation (FFD) approach was introduced firstly by Sederberg and Parry [SP86].



(a) Bunny with lattice (b) Deformed lattice and bunny

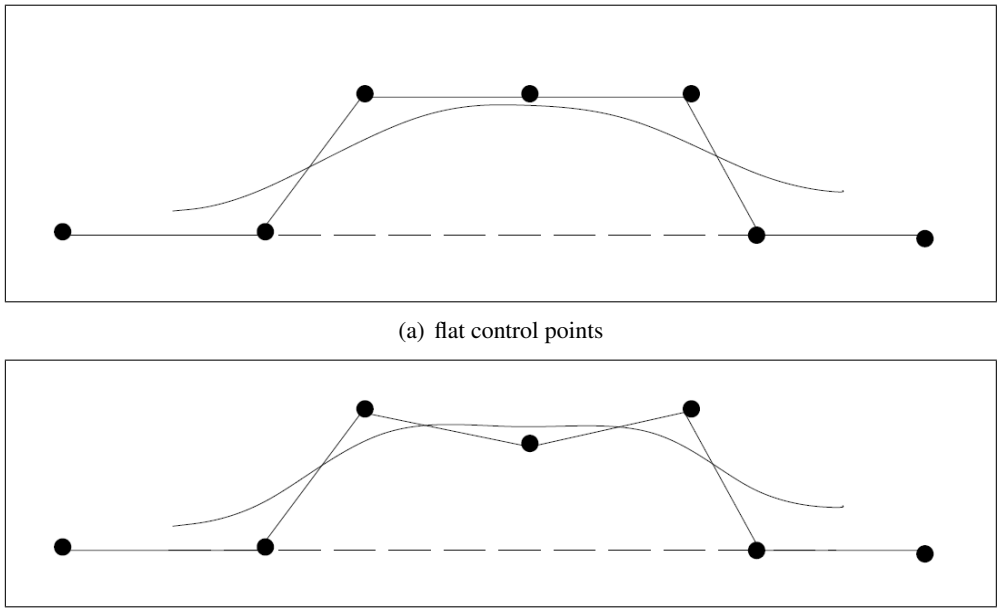
Figure 2.1: FFD on the bunny model with a regular lattice

This approach resembles the one of Barr, but Sederberg and Parry build up a lattice around the object with a small amount of lattice points. The user can translate the lattice points and the object is deformed according to the deformation of the lattice. Hence, Barrs curves are extended to a lattice or a grid. This grid encloses the whole object, and hence, the deformation is detached from the object's complexity. The user can deform arbitrary objects using the enclosing lattice. Sabine Coquillart extended this approach by removing the constraint that the lattice had to be rectangular [Coq90]. She introduces cylindrical grids, that approximate spherical objects much more precisely. She also introduces the tool metaphor. The user can define own grids and use them like a sculpting or pulling tool. The tool metaphor will be explicitly discussed later in this chapter. MacCracken and Joy extended the approach even more, so that the lattices can have an arbitrary structure [MJ96]. In order to obtain this, they introduced the use of the Catmull-Clark subdivision scheme to adjust the lattices to the objects. The FFD modelling technique is available in recent commercial modelling tools. See Figure 2.1[2] as an example for FFD. The left image shows the undeformed bunny model and a lattice surrounding it. In the right image the lattice is deformed and the bunny is deformed accordingly.

[2]Coloured images can be found in Appendix: Colour plates

All these lattice-based approaches suffer from missing precision. It is not possible to map specific points directly to a new position, only if they are lying on the lattice. But, laying all points on the lattice contradicts with the FFD approach making the deformation independent of the 3D shape. Another problem with FFD is creating flat surfaces. To do this, the user must change the lattice points in a way that is not intuitive or needs to control an overwhelming amount of parameters. This is clearly shown in Figure 2.2: the dots show the control points. The dashed line shows the original surface and the solid line shows the deformed surface. The user has to move the control points to an apparently illogical position. Hsu et al. extended the FFD approach and modified the user interaction [HHK92]. In their system the user moves some object points directly and the system finds the corresponding lattice point transformations leading to the desired deformation. This point-based approach is called direct manipulation FFD.

Another aspect that yet has not been considered are physical parameters. The models of the units in the



(a) flat control points

(b) flat surface

Figure 2.2: Problem of flat surface with FFD, courtesy:[HHK92]

engine compartment are constructed accurately, and therefore, the deformation that is applied in order to examine the assembly should retain the volume of these objects. There are several applications where the models should only be deformed as if they were elastic as for example vulcanised rubber. Rubber can be bent and stretched but its volume stays constant. Borrel and Rappoport et al. inserted the physically-based modelling aspect to FFD [BR94, RSB95]. They extended the FFD with an energy function in order to achieve a constant volume and surface energy. The surface energy is a parameter for the shape of the object, and for this reason, edges and wrinkles within a smooth shape caused by the deformation operation are avoided. This idea makes the modelling more realistic and more practical for an industrial usage. Thus, the surface editing is considered *fair*.

The physical aspect has been examined by other authors as well. However, Welch and Witkin only worked on triangulated surfaces [WW94, WW92]. They modified the mesh connectivity in order to minimise the mesh energy. This enables changing the topology of the manipulated objects, in contrast to Hirota et al.

who maintained the triangulation using a multilevel representation [HML99]. In recent work, Bendels et al. give the user the opportunity to adjust the smoothing behaviour [BKS03]. They utilise the same direct handle transformation scheme as used in the implementation presented here. They give the user the possibility to choose the smoothing function for the neighbourhood, however in this application, the smoothing function is fixed to the Gaussian. They extend the system with a so called *occluder* [BK03a] and they describe a complete editing environment in [BKK04], where they construct a workbench for virtual modelling. The user manipulates the objects with a PHANTOM [Pha] device, while holding the virtual object with the hand. The PHANTOM is a kind of pen that gives haptic feedback. The effect of holding is realised by projecting the object onto a semi-transparent glass plate. The user, who wears shutter glasses for stereoscopic viewing, can see his own hand holding a virtual object. With the use of the PHANTOM device, the user can transform single vertices on the surface while the neighbourhood of the vertices is interpolated by a smoothing function as explained. The direct handle transformation with neighbourhood smoothing is used in a similar way in the work of Pauly et al.[PKKG03]. Here, a special hybrid point-sampled geometry is used, built

> "by combining unstructured point clouds with the implicit surface definition of the moving least squares approximation." From [PKKG03, p.1].

This enables the utilisation of implicit and parametric surfaces, and thus, the application is no more limited to connected meshes.
These extensions of the FFD method show that the use of energy functions is essential for fair surface editing. There is a need for global parameters that avoid the destruction of the basic shape. This idea of global control is used in nearly all up to date mesh editing systems. Many of them use subdivision surfaces and multiresolution methods. We will take a closer look at multiresolution methods in Section 2.3. However, there is one recent work using classic FFD methods made by Tao Ju et al. [JSW05]. A classical control mesh divides the shape in coarse areas. These are deformed and the deformation is propagated to the original shape by using mean value coordinates. The mean value coordinates define a coordinate as the mean of its direct neighbours [Flo03]. These neighbours build the so called one-ring around the vertex. This method can also be used to interpolate 3D textures if a part of the mesh is cut off.
FFD is also used in commercial systems. Milliron et al. describe a software framework using FFD [MJBF02]. The authors work for Pixar and focus on commercial applications with results enhancing animation creation.

2.3 Multiresolution Editing

While the described energy minimisation and FFD methods work well for smooth surfaces, they have drawbacks for manipulating objects with a high detail level such as those acquired from scanning devices because of the missing detail preservation [GSS99, KCVS98, ZSS97]. While the interpolation of deformations is computed according to the manipulation of a coarse lattice, the details of the shape can not be stored in an efficient way and hence, they may get lost during mesh editing. In order to overcome this drawback, multiresolutional shape editing is introduced. Matthias Eck et al. [EDD+95] describe

a method for building multiresolution representations of arbitrary meshes. Based on this work Denis Zorin, Peter Schröder and Wim Sweldens developed a first multiresolution mesh editing system using subdivision surfaces in 1997 [ZSS97]. The idea of multiresolutional mesh editing is to split the complex object into a coarse base mesh and its detail levels. For each coarseness level a description of the details exist. The multiresolution technique is known from digital signal processing and uses wavelets. Gabriel Taubin was the first author to use wavelets on 3D meshes [Tau95]. He noticed that the discrete surface are signals and applied discrete Fourier analysis on them. The mesh is interpreted as a signal which can be decomposed into its frequencies. Taubin reduces the fairing problem of scanned 3D mesh data into low-pass filtering. In addition, he published an survey on signal processing of meshes [Tau00]. Zorin et al. combined this technique with FFD to achieve a detail preserving mesh editing tool. The manipulation is done in the classic FFD manner, but the user is allowed to choose the detail level. If only the base mesh is deformed, all the details, the higher frequent parts of the mesh, are retained. The advantage with respect to the classic FFD is the detail preserving low level editing that allows the manipulation of complex shapes with a large number of vertices. Deformations of highly detailed surfaces is not possible with FFD at interactive rates.

This basic approach of multiresolutional shape editing has been extended by several authors during the last decade. Leif Kobbelt et al. [KCVS98] developed a modelling system for arbitrary meshes by combining mesh hierarchies, local frame coding and multi-level smoothing. Guskov, Schröder and Sweldens [GSS99] proposed another system for arbitrary meshes. They generalised basic signal processing tools to irregular triangle meshes. Through the combination with mesh simplification tools in order to build subdivisions, they developed different applications including a mesh editing tool. Another approach was presented by Seugyong Lee in 1999 [Lee99]. He mapped the editing area onto a 2D rectangle and interpolated the editing information over this rectangle. For embedding the editing region, harmonic maps are used and the interpolation is done with multilevel B-splines.

Multiresolution editing is a very powerful tool in combination with subdivision surfaces. Much work has been done in the past ten years. [Zor05] gives a very good review about the state of the art in modelling with multiresolution subdivision surfaces. It has been presented by Denis Zorin at the tutorials session at the Eurographics conference in Dublin 2005. Subdivision surfaces are very popular for animation tasks. The famous animated motion pictures Geri's Game, A Bug's Life, and Toy Story 2 have been created using this technique.

The drawback of multiresolution editing is pointed out by Kun Zhou et al. [ZHS+05]. He argues that through the independent manipulation of the displacement vectors, indicating the deformation at each vertex, artefacts can appear. This may happen, especially in strongly deformed regions, because the details are not connected and hence, they are not preserved uniformly over the whole surface. If the deformation of the base mesh is too strong, the details will not be reconstructed at their correct position. In order to avoid this problem, mesh connection is considered. To do this, Laplacian mesh editing has been developed.

2.4 Laplacian Mesh Editing

The history of Laplacian mesh editing starts with the paper of Marc Alexa about the use of differential coordinates for mesh morphing and deformation [Ale03]. He introduces the differential (also called Laplacian) coordinates δ of a mesh and describes how to apply them especially for morphing. The differential coordinates of a mesh can be interpreted as the difference of the original mesh and a smoothed version of this mesh. As mentioned in 1.1, these coordinates describe the detail of the surface. Like in the multiresolutional approach, the details are stored separately, and only the smooth base mesh is processed. In addition, Alexa also introduces the potentials of differential coordinates in mesh editing. This potential was first used for interactive editing in the work of Lipman et al. [LSCO$^+$04]. They developed a system that allows the user to deform 3D shapes at interactive rates while the detailed structure of the object is maintained. Hence, the differential coordinates δ are defined as the detail and constructed as a transformation of the absolute coordinates $\vec{p}$; they establish a linear system that describes the transformation of the absolute coordinates of the mesh. They are defined as the difference of a vertex and the mean coordinate of its one-ring neighbours. A multiplication of the Laplacian matrix with these coordinates leads to absolute coordinates of the mesh. The Laplacian matrix is build up from the adjacency matrix and the valences of each vertex and therefore, it is quite sparse for large meshes. Then, it is reduced to the size of the ROI[3]. The ROI is determined by the user and defines the region on the surface where the deformation takes place. This localises the deformation. Once ROIs are introduced, the matrix is extended with constraints that depend on the fixed vertices that stay at their original position and on the handle vertices that are transformed by the user. This allows the reconstruction of the edited surface by solving the linear system

$$L\vec{p} = \delta \tag{2.1}$$

for each dimension x, y and z. This equation system reconstructs the local details, if it is solved by a least squares minimisation. To reach interactive rates, the solution of the linear system is precomputed with a Cholesky factorisation that splits the matrix in an upper and a lower triangular matrix. This decomposition is done once, while the new coordinates are computed very fast by a simple forward and back substitution for each dimension.

Olga Sorkine et al. [SLCO$^+$04] describe several alternative applications with Laplacian coordinates such as coating transfer and transplanting surfaces. All applications base on the solution of a sparse linear system. There are two more publications [Ale05, Sor05] that describe the same technology as a tutorial and a state of the art report.

As mentioned in the last section, the problem of multiresolution editing is the lack of connectivity of the detail structures. In Laplacian mesh editing this drawback is overcome by the definition of the differential coordinates. Due to the fact that the differential coordinates approximate the second derivative of the surface, the minimisation equation presented by Sorkine et al. is a discretisation of the Poisson equation [SLCO$^+$04]. The solutions of the Poisson equation, which is a generalisation of Laplace equation, are harmonic functions. Harmonic functions are smooth in their boundaries, and are therefore well suitable for fair surface interpolations. Yu et al. developed a similar method as Sorkine et al., but they

[3] region of interest

started from the opposite side [YZX+04]. Their equation system is different. They do not modify the Laplacian matrix, but make some changes on the right side of the equation. Instead of attaching the constraints to the Laplacian matrix, constraints are formulated as boundary conditions on the right side of the equation. These boundary conditions are defined by the euclidean distances in the editing ROI, which is determined by the user. This approach has been extended with the so called *harmonic guidance* by Zayer et al. [ZRKS05]. Harmonic guidance means that the propagation of the boundary constraints is controlled or guided by a harmonic map. Hence, the influence is not any more only dependent on the euclidean distance, but the shape structure is considered as well. This has very positive effects on branched shapes, like a cactus or an open hand. The harmonic map makes it possible to bend only one branch instead of all neighboured ones. In this thesis, this problem is avoided by the neighbourhood selection scheme. Here, the ROI is defined by a combination of euclidean distance and mesh connectivity. Every vertex whose position will be reinterpolated has to be near enough to the directly selected ones and there must be a connection to all other vertices inside the ROI. This selection scheme avoids the deformation of neighboured but not connected regions. This raises the realism of the deformations. If there is a unit with several *arms*, it is natural that the deformation of one arm does not influence the other ones.
Laplacian mesh editing acts on objects in case of rotations like if someone wrings out a sponge, involving a loss of volume on rotations. Hence, Zhou et al. [ZHS+05] introduce volumetric details in order to improve volume preserving. This idea is based on the work of Mario Botsch and Leif Kobbelt who used volumetric information in multiresolutional mesh editing in order to avoid self-intersections and make bending more natural [BK03b]. The surfaces are extended with an tetrahedral submesh which is additionally considered in the interpolations. However, this extension increases the amount of considered vertices strongly and so the computation time rises dramatically.
In addition, there are three further approaches of Mario Botsch and Leif Kobbelt that introduce alternative applications using the Laplacian. In the first one [BK04b], they describe a framework for real-time free-form modelling. Here, also an energy function is used in order to keep the shape deformations fair. They write that the order of the energy function controls the stiffness of the deformation. Therefore, they use the k-th power of the Laplacian operator in order to realise different smoothness of the boundary conditions. This approach needs to solve

$$L^k \vec{p} = \delta, \tag{2.2}$$

in a least squares sense. There are three cases for the energy functional examined:

> "...for $k = 1$ this equation characterizes membrane surfaces which minimize surface area, for $k = 2$ it characterizes thin plate surfaces which minimize surface bending and for $k = 3$ we obtain surfaces that minimize the variation of linearised curvature." From [BK04b, p.3]

These higher order Laplacians are defined recursively, as described in [KCVS98]. The discretisation of the Laplacian was implemented according to Meyer and Desbrun et al. [DMSB99, MDSB02], using the cotangent weights. Through this higher order definition, it would be necessary to extend the one-ring around a vertex defining the differential coordinate to a k-ring and hence, the computation is more time-consuming.
Their second approach [BK04a] combines the Laplacian mesh editing scheme with multiresolution. The

editing operations are executed on a smooth base mesh and details are added afterwards. To achieve a symmetric matrix even with higher order Laplacian operators, the smooth base surface is remeshed. To do this, they use a simple remeshing approach that uses the precomputed cotangent weights. This approach is not accurate, but because of its applying on the smooth base, there is no need for a high precision. For solving the system they use a direct solver with a Cholesky decomposition as well.
In their third approach, Botsch and Kobbelt [BK05] introduced the use of radial basis functions in the field of real-time shape editing. Radial basis functions are a tool for interpolation of data streams. They defined a space deformation function by a vector valued radial basis function

$$d(\vec{p}) = \sum_j w_j \varphi_j(\vec{p}) + \psi(\vec{p}),$$

where $\varphi(r) = r^3$, ψ is a polynomial of low degree and w_j are weights, that are computed to achieve a smooth interpolation. This function is the replacement of the discrete Laplacian operator in the implementation according to this thesis. Through the use of a global radial basis function instead of the Laplacian, a dense linear system is built. Botsch and Kobbelt overcome this drawback, through previous calculation steps and a transfer of computations onto the GPU[4]. Their approach also combines the aspects of exact control and high quality. Thus, it is possible to change the position of every point exactly on the one hand, and on the other one the transformation of large parts of the objects keeps the surface structure. This is the so called simplest shape principle which is a base of fair surface design. Botsch and Kobbelt see the drawback of Laplacian mesh editing in the fact that,

> "...their computational effort and numerical robustness are strongly related to the quality and complexity of the tessellation". From [BK05, p.1]

The advantage of radial basis functions is the independence from the data structure. The data can be point sampled or a mesh. It works fine for scattered data, because holes or degenerated triangles are smoothed. The same authors wrote a paper on removing degenerated triangles [BK01] and this confirms that their method is prone to anomalies. Since we work with CAD data, there was no need for radial basis functions because the tessellation of the meshes is controlled by the user and depends on the export from the CAD software. That is the main reason why the Laplacian scheme has been used here instead of a radial basis function, which would be the best alternative for this application. In the implementation presented in this thesis, the higher order boundary constraints are compensated by the weighting of the fixed and handle vertices. See Chapter 5 for details. This makes the computation faster and because the industrial application aims on doing fine deformations, the first order Laplacian has been chosen.

2.4.1 Drawbacks

The drawback of Laplacian mesh deformation is the rotation invariance of the differential coordinates. In Section 4.5 we examine this problem in detail. There are two approaches that avoid the problem by using alternative coordinates. Alla Sheffer and Vladislav Kraevoy introduced *pyramid* coordinates [SK04].

[4] graphics processing unit

These coordinates are rotation invariant, so the details of a surface are maintained also during strong rotational deformations of the shape. This method leads to good looking results, but the authors make no clear statement about the efficiency and the runtime of their algorithm. The reconstruction of the mesh from their coordinates has a quadratic complexity. Lipman et al. present rotation invariant coordinates that are linear [LSLCO05]. They split the differential description of the mesh coordinates in two parts, the first one defines the tangential part and the second one the normal part. Their reconstruction scheme is based on the solution of two sparse linear systems. Even though their coordinates are linear and the rotations lead to very good results, they also make no clear statement about the computation times of the reconstruction. They are still able to handle tens of thousands of vertices at interactive rates, but they do not compare with their previous work. This shows that it is necessary to have a balance between performance and deformation quality, in relation to the intended application. In the context of this thesis, the application aims at fine deformations and so, there is no reason to handle large rotations properly with saving thus computational time.

2.5 Shape Editing

Several shape editing frameworks have been examined according to other aspects like the user interaction and the data structure of the models. At first, the user interaction scheme is regarded which can be divided in two methods. There are systems using the tool metaphor [Coq90, GM05, Ang05], and there are frameworks using a ROI [BKS03, BK04b]. In the application in this thesis, a ROI is used in order to give the user exact control where the deformation is to take place. The ROI is divided in three parts. A handle region, an interpolation area and a fixed region. This subdivision enables flexible and precise deformation. When working with a tool, the range reachable by the tool needs to be defined. In other words, the tool has to be defined properly. The developer has two possibilities. Either he determines a number of fixed tools that are precisely defined according to their range of influence, or he gives the user the possibility to create his own tools. Hence, there is either a restriction because of a limited variety of tools or an increased difficulty to manage it because tools have to be designed by the user. Therefore, a simpler and more intuitive method such as ROI selection has been developed in order to enable rapid working. Another reason for the selection mechanisms is the experience of the users with CAD applications such as CATIA, which provide such selections, too. For working with tools in VR, it is sensible to support the user with haptic feedback. Especially tasks such as pressing or pulling the shape are hardly manageable without haptic feedback. By now, this feedback can only be realised by special input devices such as the already mentioned PHANTOM. Enabling haptic feedback has another drawback. The human tactile sense is much more sensitive than the visual system. While a human can recognise only about 30 images per second, a haptic feedback device needs to work with about 1000Hz in order to achieve a realistic simulation for the tactile sense. Due to this fact, computing the deformation of the shape for haptics needs to be much faster than if only visual feedback is necessary. The tool metaphor aims to copy the traditional sculpting methods. This makes shape editing more intuitive, because it is well-known what happens if a shape is processed with, for example, hammer and chisel.

Several practical examples can be found in the literature [BKK04, WBD02, HQ02]. At DaimlerChrysler, the assembly simulation is done by design engineers that are trained in CAD programmes. Hence, they are the main users of the editing software, thus we stick to selection mechanisms instead of virtual tools.

A further modelling approach is sketch-based modelling [ACOL00, HQ03]. Here, the user draws his

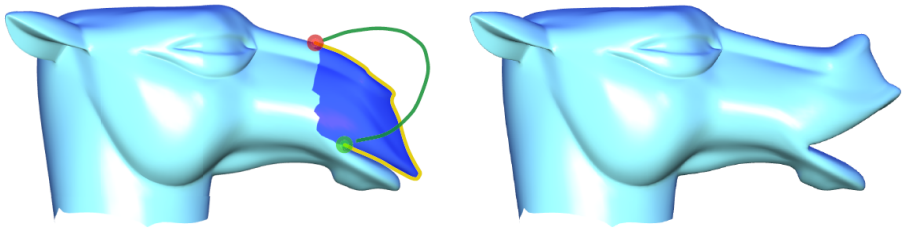


Figure 2.3: Sketch-based editing, courtesy: [NSACO05]

deformation intention with symbolic lines and strokes. This input can be realised by a sketch-pad or by usual mouse interaction. These line and stroke inputs are transformed in 3D displacements by methods from pattern recognition. Therefore, the user has to learn the meaning of the strokes in order to use the system properly. This technique can also be used for ad-initio modelling, as in the “Teddy” system by Igarashi [IMT99]. Olga Sorkine presents in her state of the art report [Sor05] another work she supported, created by Andrew Nealen et al. [NSACO05]; a sketching application in combination with Laplacian mesh editing. Here, the sketches are mapped onto the surface and another sketch draws the intended deformation. See Figure 2.3 for an example application. The user can control the measure of precision of the sketch by weighting. Depending on this, the drawn line will be roughly approximated or exactly interpolated. As you can see in the figure, the main application for such sketching methods is comic animation. In the context of an assembly simulation, the sketching does not provide the required precision. In addition, the possibilities of visual feedback for the measure of the drawn sketches are limited.

3 Foundations

3.1 Introduction

What does Laplacian mesh editing mean? What is a Laplacian? What is the coherence of meshes and graphs? How can an over-determined linear system be solved? This chapter introduces the mathematical basics of this thesis in order to answer these questions. So as to understand the idea behind Laplacian mesh editing, the reader is introduced to differential geometry and it is shown why the mesh editing is named after Pierre-Simon Laplace. The differential geometry describes continuous surfaces. As differential coordinates describe discrete surfaces, this discretisation will be explained in detail. After that, the different weighting functions for these differential coordinates are examined carefully. In order to describe the whole reconstruction method for mesh editing, some basic aspects about matrices are explained. An outlook to eigenvalues and eigenvectors shows the potentials and limits of Laplacian mesh processing. The chapter continues with a description of Laplacian mesh editing that explains where all the mentioned foundations interlace, and it ends with some numerical methods for solving sparse linear systems through factorisation and conditioning.

3.2 Laplacian

In order to explain what a *Laplacian* is, we have to introduce Laplace's equation. The solutions of this partial differential equation play a decisive role in many different fields of science, especially electromagnetism, astronomy and fluid dynamics. They describe the behaviour of electric, gravitational and fluid potentials. The connection between these topics and meshes lies in the fact that both are represented as fields. In differential geometry, surfaces are described as fields. Meshes are discrete surfaces and thereby, they can be seen as discrete fields with a finite amount of vertices. Every point or vertex has an impact on its neighbours. Vice-versa a continuous field can be seen as a mesh or graph with an infinite number of vertices. At first, the Laplace equation is described for continuous surfaces and then the discretisation is explained in detail. Laplace's equation

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0$$

sets the second derivative of a real three dimensional function $\varphi(x, y, z)$ equal to zero. This is comparable with finding inflection points in a curve. The second derivative of a surface describes the curvature. Setting this derivative equal to zero, the solutions will be points with no curvature. This equation is often shortened to

$$\nabla^2 \varphi = \Delta \varphi = 0 \tag{3.1}$$

or

$$\text{div grad } \varphi = 0,$$

where Δ is the Laplace operator, which is defined as the divergence of the gradient of a scalar field or potential. The Laplace operator is also called *Laplacian*. All functions φ that fulfil this equation, that

are continuous and whose first and second partial derivatives are continuous too, are called harmonic functions or Laplace differential functions. Harmonic functions are analytic and hence, they can be locally expressed by Taylor series.
We regard the Laplacian of a surface instead of a function φ and therefore, the Laplace-Beltrami operator

$$K(\vec{p_i}) = \lim_{r(A)\to 0} \frac{\operatorname{grad} A}{A}, \tag{3.2}$$

with A as an infinitesimal area around a point $\vec{p_i}$ and r(A) as its diameter, is introduced. As the divergence is defined as a volumetric differentiation, the limit from (3.2) can be interpreted as the divergence of $\operatorname{grad} A$. Due to this reason, the Laplace-Beltrami operator is simply the Laplace operator for manifolds and in Euclidean spaces they are identical. According to Meyer et al. [MDSB02], the Laplace-Beltrami operator can also define the product of the mean curvature and the surface normal at a given point on the surface. This product is the so called mean curvature normal $\kappa_H \vec{n}$. It is constructed by

$$K(\vec{p}) = 2\kappa_H(\vec{p_i})\vec{n_i} = \lim_{r(A)\to 0} \frac{\operatorname{grad} A}{A}. \tag{3.3}$$

The gradient of the area A directs in the normal direction and can be interpreted as the inclination of the surface, and hence as the first derivative. Computing the limit of this division is comparable with a differentiation. Thereby, the mean curvature normal $\kappa_H \vec{n}$ can be interpreted as the second derivative or the Laplacian of the surface. A detailed description of this derivation can be found in [DMSB99].
The mean curvature can also be expressed as the limit of a curvilinear integral

$$\kappa_H(\vec{p_i})\vec{n_i} = \lim_{|\gamma|\to 0} \frac{1}{|\gamma|} \int_{p\in\gamma} (\vec{p_i} - \vec{p}) dl(\vec{p}), \tag{3.4}$$

where γ is a circle domain around the point $\vec{p_i}$ with $|\gamma|$ as its area. The integral sums up the differences of all points in an infinitesimal area to a point $\vec{p_i}$. Regarding the integral we see, that the connection between meshes and Laplace's equation comes by discretisation. The discretisation of this curvilinear integral leads to the following definition of the Laplacian of meshes. Let $M = (V, E)$ be a mesh (or a graph) with n vertices V and a number of edges E. Let δ_i be the differential coordinate and $\vec{p_i}$ the absolute coordinate of a vertex i. The discrete Laplacian operator Δ_d is defined as

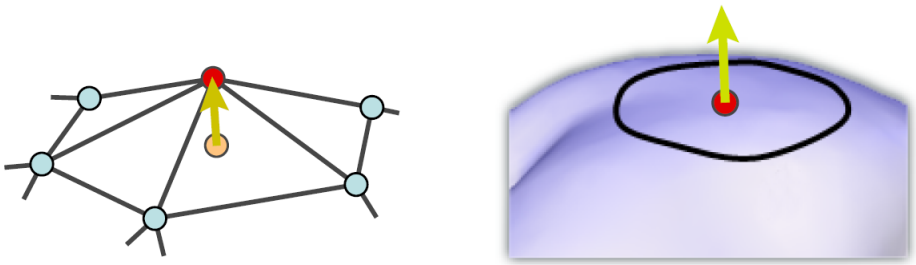
$$\Delta_d(p_i) = \delta_i = \frac{1}{d_i} \sum_{j\in N(i)} \vec{p_i} - \vec{p_j} \tag{3.5}$$

where $d_i = |N(i)|$ is the valence (number of neighbours) of the vertex i and $N(i)$ is the set of vertices j that are connected with the vertex i. These neighbour vertices j build a one-ring around the vertex i. This one-ring builds the smallest region around the vertex, and hence, it is comparable to the limit from (3.4). The differential coordinates δ can be seen as

> "a discretization of the continuous Laplace-Beltrami operator, if we assume that our mesh M is a piecewise-linear approximation of a smooth surface." From [Sor05, p.2].

Consequently, the direction of the differential coordinate approximates the surface normal, and its length is proportional to the surface curvature. This discretisation is also shown in Figure 3.1. According to

these definitions, it can be said that the Laplacian operator applied to a surface contains the first and second derivatives of the surface. The mean curvature is an expression for this second derivative. The normal is related to the first derivative, because it is always orthogonal to the tangent plane of the surface at a specified position. It can be deduced therefore, that the discretisised Laplacian approximates both the curvature and the normal of the mesh at the specified vertex. However, this approximation is only topologically correct, because we assumed that the mesh approximates the surface exactly. Hence, this is not a fact, it is necessary to enhance the differential coordinates. As a triangulated surface mesh is not stringently regular, which means that it does not only contain of unit length edges, the use of weights is necessary. As the differential coordinates are used for shape editing there is a need for geometric precision. This leads to a more detailed description of the differential coordinates.



(a) $\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} \vec{p_i} - \vec{p_j}$ (b) $\frac{1}{|\gamma|} \int_{p \in \gamma} (\vec{p_i} - \vec{p}) dl(\vec{p})$

Figure 3.1: Comparison of discrete and continuous surfaces, courtesy: [Sor05]

3.3 Differential Coordinates

Remember, the differential coordinate, in the following also called *detail vector*, of a vertex i, that belongs to the mesh $G = (V, E)$ with the set of n vertices V and the set of edges E, has been defined as

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} \vec{p_i} - \vec{p_j}, \tag{3.6}$$

where $\vec{p_i}$ is the absolute coordinate of i and d_i its valence. From this definition, it is obvious that δ_i becomes $\vec{0}$ if all neighbours j are coplanar.

The differential coordinate from (3.6) is built by the so called topological or graph Laplacian, because there is no geometrical information about the vicinity of the vertex i. Other sources call it umbrella operator because the one-ring looks like one. See Figure 3.2. All edges are assumed to be unit length, which is only true for regular meshes. The most commonly used models are not regular at all, since they are usually triangulated from CAD data. Therefore, weighting schemes are to be introduced.

3.4 Weighting

In order to add geometric weighting possibilities, one can extend the equation (3.6) by adding weights as

$$\delta_i = \frac{1}{\sum w_{ij}} \sum_{j \in N(i)} w_{ij} (\vec{p_i} - \vec{p_j}), \tag{3.7}$$

Figure 3.2: The umbrella; courtesy: [KCVS98]

where w_{ij} is the weight of the edge (i, j). There are several alternative ways to define the weights w_{ij}. Every alternative definition leads to a so called *weighting scheme*. In the following, these schemes are presented and explained. When $w_{ij} = 1 \forall i, j$, (3.7) and (3.6) become identical. Therefore, this scheme is called *uniform weighting* or *umbrella operator*. It only describes the topological properties of the mesh but not the geometrical ones. The differential coordinate is only defined by the mean of the surrounding vertices without considering their geometry. However, this does not describe the detail properly, as shown in Figure 3.3. We regard the top of the both pyramids as the vertex i whose differential coordinate is computed. The base corner points of the pyramids define the one-ring of vertices j around the vertex i. Both one-rings, left and right, have the same point as mean. This results in the same detail vector although the shape of the surface at these points differs. Hence, it is necessary to use variant values as weights.
The most common weighting scheme is the scale-dependent umbrella operator with $w_{ij} = 1/l(i, j)$, where $l(i, j)$ is the edge length. It is presented by several authors [Fuj95, Tau95]. There is a similar scheme described by Matthieu Desbrun et al. [DMSB99] which is also called the scale-dependent umbrella operator. However, Desbrun et al. use a different definition of the Laplacian. This weight has the effect that the geometric properties of the mesh are better approximated. It is a simple approach that weights only the edges like in a weighted graph. To completely approximate the geometric properties, it is necessary to add some information about the angles formed by the edges.

In the literature, two main concepts are mentioned: the *cotangent weight* and the *tangent weight*. The

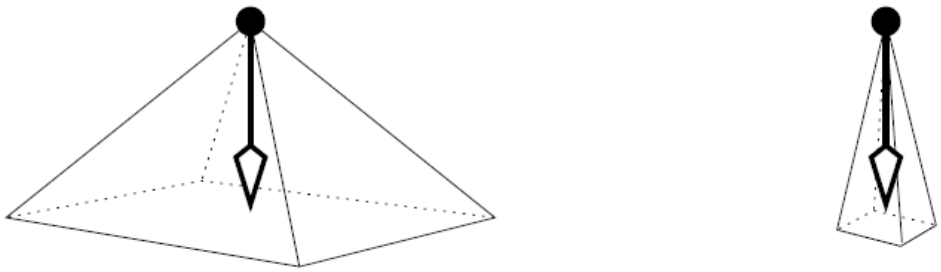


Figure 3.3: Different surface, same differential coordinate, courtesy: [DMSB99]

tangent weight, which is also called mean value weight, with

$$w_{ij} = \frac{1}{l(i,j)}(\tan\frac{\alpha_j}{2} + \tan\frac{\alpha_{j-1}}{2}),$$

where α_j is the angle between the edges (i, j) and $(i, j + 1)$ around vertex i as pictured in 3.4, has been deduced from Michael Floater et al. [FKR05, Flo03]. They introduced the mean value coordinates in 3D. This defines the detail vector δ_i as the difference between the absolute coordinate and its mean value coordinate. Floater et al. proved that the mean value coordinates are valid in 3D for arbitrary star shaped polygons [FKR05].

3.4.1 Cotangent Weighting

The definition of the cotangent weight

$$w_{ij} = \cot \gamma_{j-1} + \cot \beta_j$$

is not so easy to derive from literature. It seems that every publication has its own way to derive the cotangent formula. In the following, some of them are outlined. Cotangent weights are used for the mesh editing application in [SLCO$^+$04, Ale05, ZHS$^+$05]. The formula appears in different forms in each paper. The oldest one is the work of Pinkall and Polthier [PP93], who obtained the formula by minimizing the Dirichlet energy over a triangulation. Matthieu Desbrun comes to the formula by calculating the gradient of a surface area for the entire one-ring vicinity [DMSB99]. According to Meyer et al., the discretisation of the Laplace-Beltrami operator leads to the same formula [MDSB02]. But all the results from these papers differ in the multiplication factor. Hence, in this thesis, the work of Matthias Eck et al. [EDD$^+$95] and their spring constant has been examined in detail. They introduced a spring constant κ_{ij} for the edge (i, j). It is defined by

$$\begin{aligned}\kappa_{ij} = &\big(l(i,j{-}1)^2 + l(j,j{-}1)^2 - l(i,j)^2\big)/A(i,j,j{-}1) \\ &+ \big(l(i,j{+}1)^2 + l(j,j{+}1)^2 - l(i,j)^2\big)/A(i,j,j{+}1),\end{aligned} \tag{3.8}$$

where $A(i, j, k)$ indicates the area of the triangle stretched by the indexed vertices. By converting this

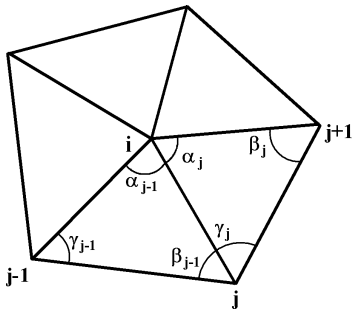


Figure 3.4: Vertex i and its one-ring neighbours

formula the edge vectors $\vec{e}(x, y)$ with $|\vec{e}(x, y)| = l(x, y)$ are introduced. So the area A can be expressed as follows:

$$A(i,j,j{-}1) = \frac{1}{2}\big(\big|\vec{e}(j,j{-}1) \times \vec{e}(i,j{-}1)\big|\big) \tag{3.9}$$

$$A(i,j,j{+}1) = \frac{1}{2}\big(\big|\vec{e}(j,j{+}1) \times \vec{e}(i,j{+}1)\big|\big). \tag{3.10}$$

In order to keep track of the system the substitutions

$$L_{j-1} := l(i,j{-}1)^2 + l(j,j{-}1)^2 - l(i,j)^2 \tag{3.11}$$

$$L_{j+1} := l(i,j{+}1)^2 + l(j,j{+}1)^2 - l(i,j)^2 \tag{3.12}$$

are made. So we can rewrite (3.8) as

$$\kappa_{ij} = \frac{1}{2}\left(\frac{L_{j-1}}{|\vec{e}(i,j{-}1)\times\vec{e}(j,j{-}1)|} + \frac{L_{j+1}}{|\vec{e}(i,j{+}1)\times\vec{e}(j,j{+}1)|}\right). \tag{3.13}$$

Remembering to the Theorem of Pythagoras for arbitrary triangles $c^2 = a^2 + b^2 - 2ab\cos\gamma$ and considering the notation from Figure 3.4, we have

$$l(i,j)^2 = l(i,j{-}1)^2 + l(j,j{-}1)^2 - 2l(i,j{-}1)l(j,j{-}1)\cos\gamma_{j-1}. \tag{3.14}$$

This leads to

$$\begin{aligned} L_{j-1} &= 2l(i,j{-}1)l(j,j{-}1)\cos\gamma_{j-1} \\ &= 2\big(\vec{e}(i,j{-}1)\cdot\vec{e}(j,j{-}1)\big) \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} L_{j+1} &= 2l(i,j{+}1)l(j,j{+}1)\cos\beta_j \\ &= 2\big(\vec{e}(i,j{+}1)\cdot\vec{e}(j,j{+}1)\big). \end{aligned} \tag{3.16}$$

Merging these equations (3.8) can be expressed as

$$\begin{aligned} \kappa_{ij} &= 2\cdot\frac{1}{2}\Big(\frac{\vec{e}(i,j{-}1)\cdot\vec{e}(j,j{-}1)}{|\vec{e}(i,j{-}1)\times\vec{e}(j,j{-}1)|} + \frac{\vec{e}(i,j{+}1)\cdot\vec{e}(j,j{+}1)}{|\vec{e}(i,j{+}1)\times\vec{e}(j,j{+}1)|}\Big) \\ &= \cot\gamma_{j-1} + \cot\beta_j. \end{aligned} \tag{3.17}$$

This means that using the spring constant from Eck et al. [EDD+95] for harmonic maps as a weight w_{ij} leads to the cotangent weighting scheme for Laplacian mesh editing. These weighting schemes enhance the detail vector in order to better approximate the geometries in the mesh. As already mentioned, the differential coordinates can be constructed by a multiplication of the Laplacian matrix of the mesh with the absolute positions of each vertex. As we want to reconstruct the absolute positions after editing some vertices and fixing some others, it is necessary to examine the properties of the Laplacian matrix.

3.5 Laplacian Matrices

The Laplacian matrix has several important properties, such as being sparse and symmetric. The mesh editing system described in this thesis builds up a linear equation system. This system is delineated as one equation $L\vec{p} = b$, where L is the Laplacian matrix. It is constructed through $L = D - A$, where D is a diagonal matrix with the valences of each vertex and A is the adjacency matrix of the current mesh. As the mesh can be interpreted as a graph, the Laplacian matrix is often called the graph Laplacian. This leads to the examination of the main properties of the matrix L, if it is constructed in this manner.

3.5.1 Sparseness

A matrix is called sparse, if it has many more zero elements than non-zero ones. Zero elements are important for computational aspects, because multiplications with zeros need no computation. In addition, it is not even necessary to store the zero entries. If at least one element in a row or column of a matrix is non-zero, the row or column is called occupied, and thus, it has to be stored because it contains information. Sparse matrices are mostly stored as double lists. In a double list, all entries contain the non-zero matrix element and an index. In this implementation, the sparse matrices are stored as follows. Because of the connectivity of the manipulated meshes the valence of each vertex is greater than zero and therefore, the diagonal is completely occupied. For each row an array is constructed. Every entry of this array contains an index and a value. The index indicates the current column. Thus, only the non-zero entries are stored and it is easy to count the number of entries of each row.
Sparse matrices provide the possibility to significantly accelerate the classic matrix algorithms. But the drawback of them is the fact that many of these classic matrix algorithms are not easy to adapt to the sparse matrix data structure with enhancing their efficiency. Recent compilers optimise the machine code in order to avoid zero multiplications and speed up the system. Furthermore, it is necessary to balance between the use of new complex algorithms and the use of fast and simple ones. In this implementation, both representations are used concurrently. Each task is processed with the usage of the appropriate data structure. For example, a matrix multiplication is much faster with the sparse matrix data structure, but not its Cholesky decomposition.

3.5.2 Symmetry and Rank

A Laplacian matrix is symmetric. That means that $L^T = L$. Symmetry is one of the most important properties of a matrix, because of the impact it has on the eigenstructure of the matrix. As certainly known, a linear equation system can have no solution, one solution or an infinite number of solutions. The number of solutions existing can be described by the rank of the matrix involved. The rank r of a matrix is the amount of linearly independent row or column vectors. r is always less or equal than the dimension n of the diagonal of a matrix. Every matrix with $r = n$ and non-zero determinant is called *regular*. Thus, it can be said, that every regular matrix has an inverse and thus its linear equation system can be solved. The Laplacian matrix L is *singular* (not regular). Hence, it is not invertible and therefore, the equation system is not analytically solvable. The singularity can be shown as follows. The Laplacian matrix is symmetric and every row sums up to zero. This results in $\det L = 0$ and its rank can be determined. L is symmetric, due to this fact, each column sums up to zero, too. In other words, all the row vectors sum up to the zero vector. As the last row vector is equal to the negated sum of the other rows, it can be constructed as a linear combination of the other rows. For this reason L has the rank $n-1$, if the mesh is connected. For the shape editing application it is quite usual to work with one connected component. In the case of several connected components, the rank becomes $n-k$, where k is the number of connected components. This makes it clear why it is necessary to add constraints when the absolute coordinates are to be reconstructed out of the detail vectors. If the Laplacian of a connected mesh and the differential coordinates of the mesh are known, it is only then possible to reconstruct the absolute

coordinates, if at least one of the absolute coordinates is determined, because such a determination raises the rank up to n and makes the matrix invertible and hence, the system solvable.

3.5.3 Norm

In an editing application, it is usual to transform more than just one vertex. As described, the user defines a ROI containing fixed and handle vertices. Due to this, there are several constraints that over-determine the equation system. Such a system is best solved numerically. For approximating the quality of numerical solutions of linear equation systems, it is necessary to introduce the following terms.
The norm $\|A\|$ of a matrix $A_{n\times m}$ can be defined in several ways. According to the literature [BSM00, Saa96], the three most important norm definitions are the following, the so called p-norms.

- Sum of column norm ($p = 1$):
$$\|A\|_1 := \max_{1\leq j\leq m} \sum_{i=1}^{n} |a_{ij}|. \tag{3.18}$$
- Spectral norm ($p = 2$):
$$\|A\|_2 := \sqrt{\lambda_{max}(A^T A)}, \tag{3.19}$$
where $\lambda_{max}(A^T A)$ denotes the largest eigenvalue of the matrix $A^T A$.
- Sum of row norm ($p = \infty$):
$$\|A\|_\infty := \max_{1\leq i\leq n} \sum_{j=1}^{m} |a_{ij}|. \tag{3.20}$$

A norm of a matrix has the correspondent properties of a vector norm. So we have

$$\|A\| \geq 0, \qquad \forall A \in \mathbb{C}^{m\times n}, \qquad \text{and } \|A\| = 0, \text{ if and only if } A = 0,$$

$$\|\alpha A\| = \alpha \|A\|, \qquad \forall A \in \mathbb{C}^{m\times n}, \quad \forall \alpha \in \mathbb{C}$$

and the triangle inequality

$$\|A + B\| \leq \|A\| + \|B\|, \qquad \forall A, B \in \mathbb{C}^{m\times n}.$$

A norm is called consistent, if

$$\|AB\| \leq \|A\| \|B\|. \tag{3.21}$$

Every p-norm ((3.18)-(3.20)) is consistent. A result of the consistency is that for every square matrix A,

$$\left\|A^k\right\|_p \leq \|A\|_p^k. \tag{3.22}$$

These norms are necessary to compute the condition number of a matrix, which is a measure for the quality of a solution. Later, the quality of the solutions in the Laplacian mesh editing application is examined and therefore, the defined norms are applied.

3.5.4 Eigenvalues and Eigenvectors

Before the Laplacian mesh editing scheme is explained in detail, the spectral properties of the graph Laplacian are described. With the help of the spectral properties the coherence of the rank and the number of connected components of the mesh or graph will be shown. This coherence fortifies the theoretic base of the Laplacian mesh editing approach. The Laplacian mesh editing bases on the fact that the rank of the Laplacian is $n-1$ because the absolute coordinates can be reconstructed if and only if one or more vertices are fixed. To show that the rank of a Laplacian is determined by the number of connected components, the eigenvalues of the Laplacian need to be considered. The following theorem has been formulated in [BSM00, p.289]:

Theorem 3.1 *The rank of a matrix depends on the multiplicities of the eigenvalues.*

In order to explain the term eigenvalues, we need to introduce the special eigenvalue problem. Remember that L is a symmetric Laplacian matrix with size n. Solving

$$L\vec{p} = \lambda\vec{p}$$

is called the special eigenvalue problem. It can be rewritten as

$$(L - \lambda I)\vec{p} = 0, \tag{3.23}$$

where I denotes the identity matrix. The complex scalar λ is called the eigenvalue and $\vec{p}$ is called the eigenvector of the matrix L. If and only if

$$\det(L - \lambda I) = 0,$$

the homogeneous equation system (3.23) has non-trivial solutions $\vec{p} \neq \vec{0}$. Through the expansion of the determinant it arises

$$\det(L - \lambda I) = \begin{vmatrix} l_{11} - \lambda & l_{12} & l_{13} & \dots & l_{1n} \\ l_{21} & l_{22} - \lambda & l_{23} & \dots & l_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} - \lambda \end{vmatrix}$$

$$= P_n(\lambda) = (-1)^n\lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0 = 0,$$

where $P_n(\lambda)$ is a polynomial of degree n. This is called the *characteristic* polynomial. This polynomial is constructed by the calculation scheme for the determinant of the system. The coefficients a_i are calculated by using rules for the calculation of the determinant. This calculation is very expensive and depends on the size of the matrix n. The roots of the characteristic polynomial are the eigenvalues of the matrix L. If Q is a regular matrix, the characteristic polynomial of $Q^{-1}LQ$ is the same as P_n. Hence, it holds that $Q^{-1}LQ$ has the same eigenvalues as L. It is obvious that the matrix L with size n has exactly n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, because every polynomial of degree n has n roots, if they are counted with their multiples. The set of all multiple eigenvalues is called the spectrum $\sigma(L)$ and the maximal value

$$\rho(L) = \max_{\lambda \in \sigma(L)} |\lambda| \tag{3.24}$$

is called the spectral radius of L.
Since $\det L = 0$, we get the eigenvalue $\lambda_1 = 0$. In Section 3.5.2 it is written that the sum of every row of L sums up to zero. This implies the non-trivial eigenvector $(1, 1, \ldots, 1)^T$ for the eigenvalue λ_1.
Hence, L is a real symmetric matrix, the spectral theorem is valid. The spectral theorem includes the following five items:

Theorem 3.2 Spectral theorem

- *L has real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and n orthonormal eigenvectors, that form a basis in $\mathbb{R}^n$.*
- *The multiplicity of the eigenvalue λ_i is the order of λ_i as root of the characteristic polynomial and so the maximal number of linearly independent eigenvectors of λ_i.*
- *It exists Q with $Q^{-1} = Q^T$ so that $Q^T L Q = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$.*
- $\det(L) = \prod_{i=1}^{n} \lambda_i$.
- *trace$(L) = \sum_{i=1}^{n} \lambda_i$, whereby the trace of a matrix L is the sum of its diagonal elements.*

In case of the Laplacian, the diagonal elements are the valences (the number of direct neighbours) of each vertex. Therefore, the sum of the valences is the sum of the eigenvalues. This makes it possible to bound the largest eigenvalue of the mesh Laplacian. This property will be recalled later. We know that the Laplacian has n real eigenvalues, but we need to examine if they are all positive. This leads to an examination of the spectrum of the Laplacian. Regarding the mesh or graph $G = (V, E)$ with the set of vertices V and its edges E, we can write $L = L(G)$ and call L the Laplacian of the graph G. Remember the following

$$L = D - A; \qquad L\vec{p} = \lambda\vec{p}$$

with D as a diagonal matrix with the valences of each vertex and A as the adjacency matrix. Let R^V be a set of functions

$$R^V = \{f : V \to \mathbb{R}\}.$$

For every vertex i there is a function $f_i \in R^V$. With the summation $(f + g)_i = f_i + g_i$, the multiplication by real numbers $(af)_i = a(f_i)$ and the inner product

$$\langle f, g \rangle = \sum_{v \in V} f_v g_v,$$

R^V becomes a real vector space of dimension n. The corresponding norm in R^V is

$$\|f\| = \langle f, f \rangle^{1/2} = (\sum_{v \in V} f_v^2)^{1/2}.$$

The Laplacian matrix L can be regarded as a linear operator on R^V. Its action is determined by the rule of matrix-vector multiplication, i.e. $g = Lf$ is the function defined by the formula

$$g_i = (Lf)_i = \sum_{v \in V} (L)_{iv} f_v, \qquad i \in V.$$

There is a natural quadratic form associated with L:

$$\langle f, Lf \rangle = \sum_{iv \in E} a_{iv}(f_i - f_v)^2, \tag{3.25}$$

where a_{iv} is a positive weight for the edge between the vertices i and v. Because L is real symmetric, the natural form (3.25) implies that L is semi-positive definite and therefore all eigenvalues are non-negative [Moh97]. Considering this, the smallest eigenvalue is $\lambda_{min} = 0$. The largest eigenvalue λ_{max} is bounded to

$$\lambda_{max} \leq 2d_{max}(G),$$

where $d_{max}(G)$ is the maximal valence of a vertex of the mesh G. A proof of this is given in [Moh97, p.10]. This spectral property, including the limits of the eigenvalues, opens the possibility of mesh compression. Similar to the JPEG compression of images, 3D meshes can be compressed by cutting off the high frequencies. See the state of the art report of Olga Sorkine [Sor05] as an introduction or other literature [BCG05, KG00] for details. Earlier in this section, we found out, that there are n real eigenvalues and that at least one $\lambda_1 = 0$ exists if G is connected. Therefore, the following theorem is posed:

Theorem 3.3 *The multiplicity of the value* 0 *as an eigenvalue of* $L(G)$ *is equal to the number of connected components of* G*.*

A proof of this proposition can be found in the work of Mohar [Moh97]. A graph G with its Laplacian L has k connected components if and only if $\lambda_1(L) = \cdots = \lambda_k(L) = 0$ and $\lambda_{k+1}(L) > 0$. As mentioned in Theorem 3.1, the rank of a matrix depends on the multiplicities of the eigenvalues and therefore, the rank of the Laplacian is $n - k$. In our mesh editing application, we assume that the mesh is connected and hence, the Laplacian matrix L has the rank $n - 1$. This fact proves that it is only necessary to add one constraint, in form of a linear independent equation, in order to make the equation system solvable. This can be done by determining the absolute position of one single vertex. More about this in the next section.

3.6 Laplacian Mesh Editing

Recall: on a mesh $M = (V, E)$ with n vertices V and a number of edges E, we introduced a differential coordinate

$$\delta_i = \vec{p_i} - \frac{1}{d_i} \sum_{j \in N(i)} \vec{p_j}$$

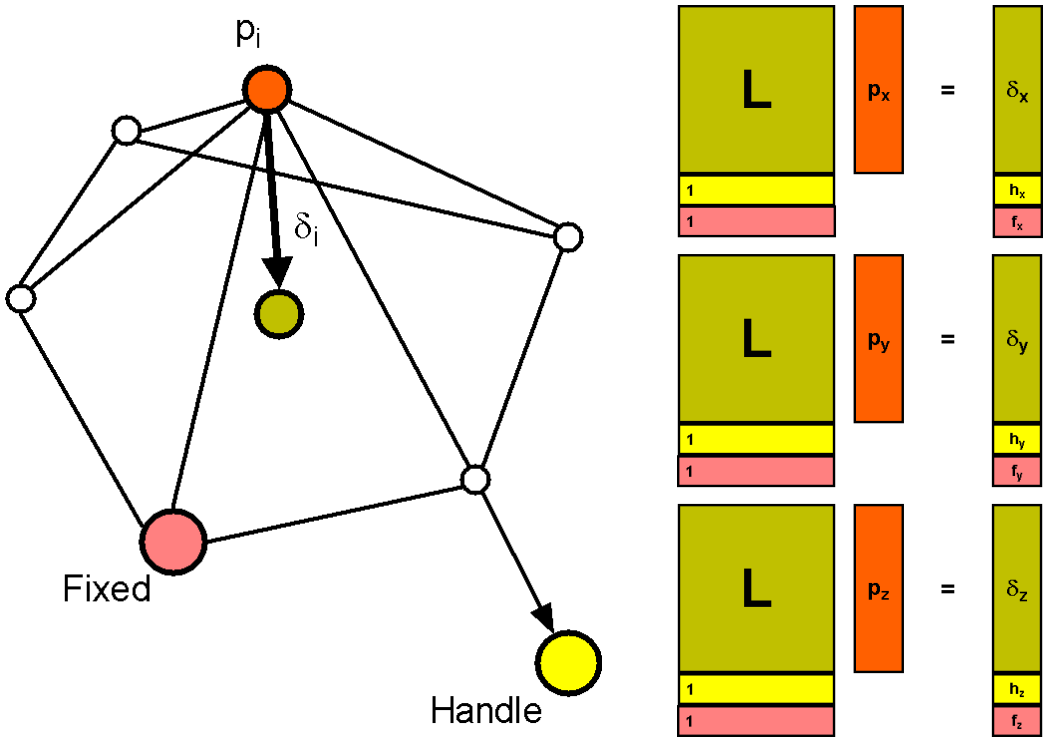
in equation (3.5). The differential coordinate describes the detail of the shape at the vertex i, and therefore, it is also called the *detail vector*. It is already mentioned that the Laplacian mesh editing scheme is based on the solution of a linear equation system. This system is built as

$$L\vec{p} = \delta, \tag{3.26}$$

with $L = D - A$ as the Laplacian matrix with size (n, n). $\vec{p}$ is a vector with the absolute position of each vertex and δ is a vector containing the differential coordinates. As mentioned earlier, the matrix

L has rank $n-1$. Therefore, the equation (3.26) is not analytically solvable. Marc Alexa found out that there are several possible applications by fixing at least one vertex [Ale03]. To solve the equation for the absolute coordinates, only the absolute coordinate of one vertex needs to be determined. For an editing application, we first need to specify the absolute coordinate of at least one vertex by adding a row $(0,\ldots,1,0,\ldots,0)$ with the 1 at the kth position in the row vector to the matrix L. In the vector δ, which contains the detail vectors of each vertex, the absolute position of the kth vertex $f_k = p_k$ has to be attached. Now the kth vertex is fixed at its absolute position. For editing another vertex with index m, also a row is added to L, but its new absolute position h_m, which is defined by the user, is attached to δ instead of the old position. See Figure 3.5 as a survey of the construction of the system. The vectors $\vec{p}$ and δ contain three-dimensional entries. Therefore, in the implementation, it is necessary to split the system into three parts, one for each dimension. According to this editing approach, the user has to define a so called region of interest (ROI). He selects the vertices he wants to edit directly as handle vertices, and the area of influence is determined by a ring of fixed vertices placed around the selected ones. The vertices in the area between the handle vertices and the fixed ring are reconstructed as explained in the following. More details about the selection scheme can be found in Section 4.3.2.

Now the system is over-determined and in general, there does not exist an exact solution. Sorkine and



(a) Mesh structure (b) Equation system

Figure 3.5: Scheme for Laplacian mesh editing.

Lipman et al. found out that a least squares solution is numerically more stable and easier to compute because of this user controlled over-determination [SCOT03, LSCO$^+$04]. The linear equation system

$$L'\vec{p} = \delta'$$

is over-determined. L' and δ' have the size n_0 and $\vec{p}$ has the size n with $n_0 > n$. But how is this numerical least squares solving realised? To answer this question we introduce an error vector $\vec{r}$ with

$$\vec{r} = L'\vec{p} - \delta', \qquad \vec{r} \neq \vec{0}.$$

The norm $\|\vec{r}\| = \sqrt{\vec{r}^T\vec{r}}$ is called the *residual*. $\vec{p}$ will be determined in the manner that

$$\sum_{i=1}^{m} r_i^2 = \vec{r}^T\vec{r} = (L'\vec{p} - \delta')^T(L'\vec{p} - \delta') \rightarrow \min. \tag{3.27}$$

This means that the quadratic error is minimised. This method was developed by Carl Friedrich Gauss, and it is also called the root mean square problem or least-squares method. The next solution step is called Gauss-transform. The vector $\vec{p}$ minimises the error in (3.27) if and only if $\vec{r}$ is orthogonal to all columns of L'. This can be written as

$$L'^T\vec{r} = L'^T(L'\vec{p} - \delta') = 0$$

or

$$L'^T L'\vec{p} = L'^T\delta'. \tag{3.28}$$

The equation (3.28) is a linear equation system where $L'^T L'$ is symmetric and also positive-definite. Solving a linear system with these properties is best done with a factorisation. The matrix $L'^T L'$ is decomposed into an upper and a lower triangular matrix. The Cholesky decomposition does such a factorisation in a very efficient way, and hence, it is used in this application.

3.7 Cholesky Decomposition

As described in [BSM00, p.916], Cholesky's method bases on the definition that every symmetric positive-definite matrix of real numbers A can be decomposed into

$$A = MM^T,$$

where M is a lower triangular matrix with positive diagonal entries. M is composed as

$$M = \begin{pmatrix} m_{11} & & & & \\ m_{21} & m_{22} & & 0 & \\ m_{31} & m_{32} & m_{33} & & \\ \vdots & & & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \dots & m_{nn} \end{pmatrix},$$

where

$$m_{kk} = \sqrt{a_{kk}^{(k-1)}}, \qquad m_{ik} = \frac{a_{ik}^{(k-1)}}{m_{kk}} \qquad (i = k, k+1, \dots, n).$$

The entries a of the matrix A are updated recursively in each step with

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik}m_{jk} \qquad (i, j = k+1, k+2, \dots, n).$$

After n steps $A^{(n+1)} = I$, and the primary linear equation system $A\vec{x} = \vec{b}$ can be solved in the following steps:

1. $A = MM^T$: do the Cholesky decomposition and substitute $M^T\vec{x} = \vec{c}$,
2. $M\vec{c} = b$: compute the auxiliary vector $\vec{c}$ by forward substitution,
3. $M^T\vec{x} = \vec{c}$: compute the solution $\vec{x}$ by back substitution.

The Cholesky decomposition runs in $O(n^3)$ for dense matrices. Since the Laplacian is sparse, the runtime of the algorithm can be shortened significantly by using fill-in reducing pre-ordering steps [Saa96]. In addition, in this application, the decomposition is performed only once for every ROI selection. Finding the solution by forward and back substitution should allow interactive rates, even though they have to be repeated three times, one for each dimension. These last two steps can be done in linear time because of the triangular structure of the matrix M. This makes the Cholesky decomposition a very suitable tool for Laplacian mesh editing.
However, there is one problem with the Cholesky decomposition. It is an unstable algorithm, which behaves in a reasonable way for large residuals $\|\vec{r}\|$ and small solutions $\|\vec{x}\|$. In order to examine when these conditions are satisfied, the conditioning of linear systems needs to be specified.

3.8 Condition Numbers

In the case of numerically solving linear systems, it is necessary to check how prone to error the solution is. One indicator can be the determinant of a matrix, but for large matrices, the determinant is mostly neither a good indication of quasi singularity nor a reasonable degree of sensitivity of the linear system. The reason for this is that $\det(A)$ is the product of the eigenvalues, which depends very much on the scaling of a matrix. Therefore, the condition number of a matrix is introduced, which is invariant of scaling. It is defined as

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|. \tag{3.29}$$

According to the matrix norms defined in Section 3.5.3, the condition numbers are indicated in the same way as the used norm. For example,

$$\kappa_1(A) = \|A\|_1 \left\|A^{-1}\right\|_1.$$

The larger the condition number, the more ill-conditioned the linear system is, and thus the greater the impact of an error on the solution. Linear systems with a large condition number are called ill-conditioned and if the condition number equals to one, the system is well-conditioned. For singular matrices such as the Laplacian the condition number is defined as $\kappa(L) = +\infty$. The condition of the current system for mesh editing will be examined in Section 4.4.3.2.

4 Implementation of Shape Editing Methods

4.1 Introduction

After presenting the mathematical foundations, this chapter describes the software application that was developed in this thesis. The source codes of the software can be found on the CD-ROM attached to this diploma thesis. Due to the fact, that DaimlerChrysler owns the software framework “veo”, there is no executable programme available. Recall that the software, as mentioned in 1.1, is intended to be used by design engineers in order to enhance the assembly simulations by allowing to deform the units that are tested for mounting. The assembly simulation is done with VR software “veo”, which has been introduced in Section 1.1. At the current state, all surface deformations have to be done in a CAD programme. The design engineers who do the assembly simulation are prevented from switching the software by allowing to deform objects in the “veo” framework. Both softwares work with different kind of data. Hence, the pluses and minuses of working in virtual environments are presented firstly. Then, the user interface is presented including user interaction and selection mechanisms. The user has different possibilities to transform the selected vertices. Different types of vertex transforms are presented, and then, the properties of meshes in general are described. After that, this chapter will close with a larger section examining the interpolation of the neighboured vertices. This interpolation is realised, as already mentioned, by the Laplacian shape editing scheme. Here, the key features of Laplacian mesh editing are examined in detail. In order to get a better access to the topic, the working environment is described firstly.

4.2 Working in Virtual Reality and on Triangles

There is plenty of literature on working in virtual environments, and there are two sources to emphasise according to car manufacturing which have been considered: a diploma thesis by Sylvie Schoeniger [Sch04], and a book by Engelbert Westkämper et al. [WBD02]. Sylvie Schoeniger wrote her diploma thesis recently at DaimlerChrysler in Ulm. She describes all requirements needed for implementing a deformation system for virtual reality. She presents different kinds of manipulation tools and her thesis contains a detailed part about the modelling metaphors in relation to the application at DaimlerChrysler. Westkämper, Bullinger and Deisinger describe in their book all the issues involved in developing a shape deformation system. They describe all the different aspects of the working environments such as display, input device, scaling issues, tracking devices, computation power requirements and so on. The book gives a good introduction to all the topics that are necessary to develop such a system.
The main difference between working in virtual reality environments and a CAD workspace is the data structure. In CAD, the geometric data of the 3D objects is stored and processed as exact analytical descriptions of the shape. These descriptions contain surface functions. This description needs a lot of computation power, so there is no way to add physical simulations such as an assembly test with realistic visual effects using this data structure in real-time. Modern graphic cards are optimised to display triangles. Every 3D object is approximated as a triangular mesh and the graphics hardware draws the triangles. This is what it is optimised for. Therefore, a high quality rendering of triangulated 3D surfaces

is possible in real-time. The quality of the approximation depends on the curvature of the approximated surface and on the amount of triangles. The triangulated models in car design need to be very accurate, because otherwise the assembly tests and other simulations do not lead to sufficiently precise results. Due to this fact, the amount of triangles rises rapidly with the size and complexity of the 3D objects. However, every triangulation is only an approximation of the accurate data and thus, the mesh editing system works only on an approximation, which has to be kept in mind. In the following, the user interface is regarded.

4.3 User Interface

4.3.1 Interaction

The user interface was developed together with Birgit Meisert who was working for an internship at DaimlerChrysler. At first, the working environment in which the programme will be used mainly, had to be investigated. It is a classic desktop environment with a standard mouse as the main input device and a 3D space mouse as a second device. The VR software "veo" is also used on desktop computers for simple planning or developing tasks. Especially if there is no need for a stereoscopic view, it is used for working on quick assembly simulations immediately. In this case, the space mouse is used to translate and rotate the objects in the scene. In addition, the assembly tests are made with this 6-DOF[5] input device. Because of the frequent use of the space mouse for other tasks in "veo", we decided to use a standard 2D mouse as the main input device.
Interviews with prospective users pointed out, that the users wished to interact with the system on a very low level. Each vertex of the object should be transformed with maximum precision. On the other hand, it should also be possible to deform large areas in a fast and intuitive way. So it was decided to implement an intuitive and well known selection mechanism at first. The modelling tasks is based on an exact and user controlled localisation, and therefore, the following selection scheme is necessary.

4.3.2 Selection

In order to achieve this exact localisation, the user is allowed to select faces and vertices. The selected vertices are marked with a blue cube around the vertex. Selecting a face leads to the automatic selection of all adjacent vertices. These marked vertices are called the *handle* vertices or *selected* vertices in the following. The size of the cube is adjustable to the size of the object. These cubic markers make it possible to grab the selected vertices and transform them. The grabbing and transforming task is implemented in a classic and intuitive drag and drop manner. This means, the user clicks on one of the selected vertices he intends to transform, holds the mouse button and moves the mouse, and thereby also the vertices, to the new position. Here, he releases the mouse button, and the vertices are placed at the new intended position. This usual method makes it easy to reach appropriate results rapidly without a long period of training. There are three kinds of transformations possible which will be described in detail in Section 4.4.1.
The user has the possibility to add more vertices to his selection by holding the shift key pressed and clicking on other vertices of the object. In this way, he can also deselect single vertices. For a better

[5] degrees of freedom

selection of large areas and greater amounts of vertices, three different tools for selection are implemented. They are the so called frame selection tools. The user draws, by drag and pull, a frame on the screen and all vertices that lie inside the frame are selected. If there is a connection to the backside of the object, the invisible vertices are selected as well. The frames can be rectangular, ellipsoidal or a polygon. These three possibilities enable an intuitive selection independent of the shape of the 3D object. This interface is a straightforward 3D extension of the well known interactive technique called *direct manipulation* [Pre94].

As mentioned in Section 3.6, the user has to define a ROI, including the neighbourhood around the

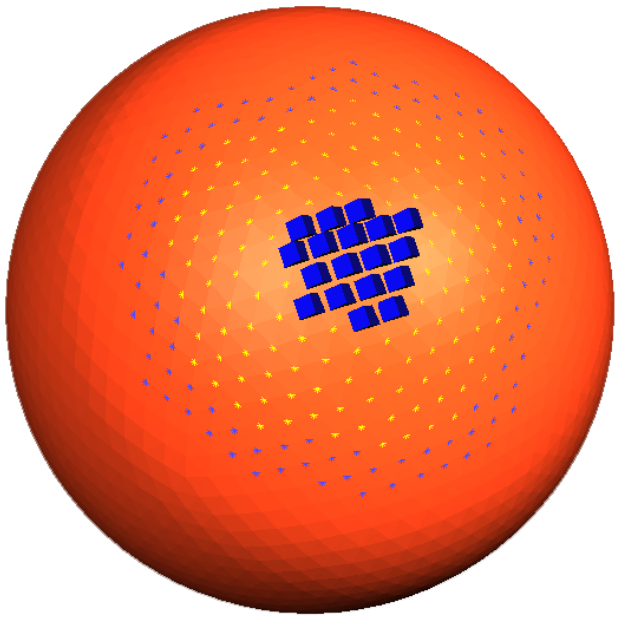


Figure 4.1: A classic ROI selection on a sphere

selected vertices and a number of fixed vertices around the neighbourhood. In Figure 4.1, a classic ROI selection on a sphere is shown. The blue cubes are the handle vertices, the yellow stars indicate the neighbour vertices, while the purple stars mark the ring of fixed vertices with a radius of two. On the one hand, he is able to define the radii of these regions, and on the other one, he can also define the neighbourhood region with the same tools as for the handle vertices. Here, some selections can be made where the neighbourhood region is not placed around the handle vertices. Due to the fact that such selections mostly lead to unrealistic deformations, even unskilled users know to avoid them. Independent from the selection, the software keeps stable, but such a ill-defined deformation can destroy the whole shape. For it, an “Undo” button is realised, that takes back the last deformation. The “Undo” option is a great advantage of working with models in virtual environments instead of solid materials.

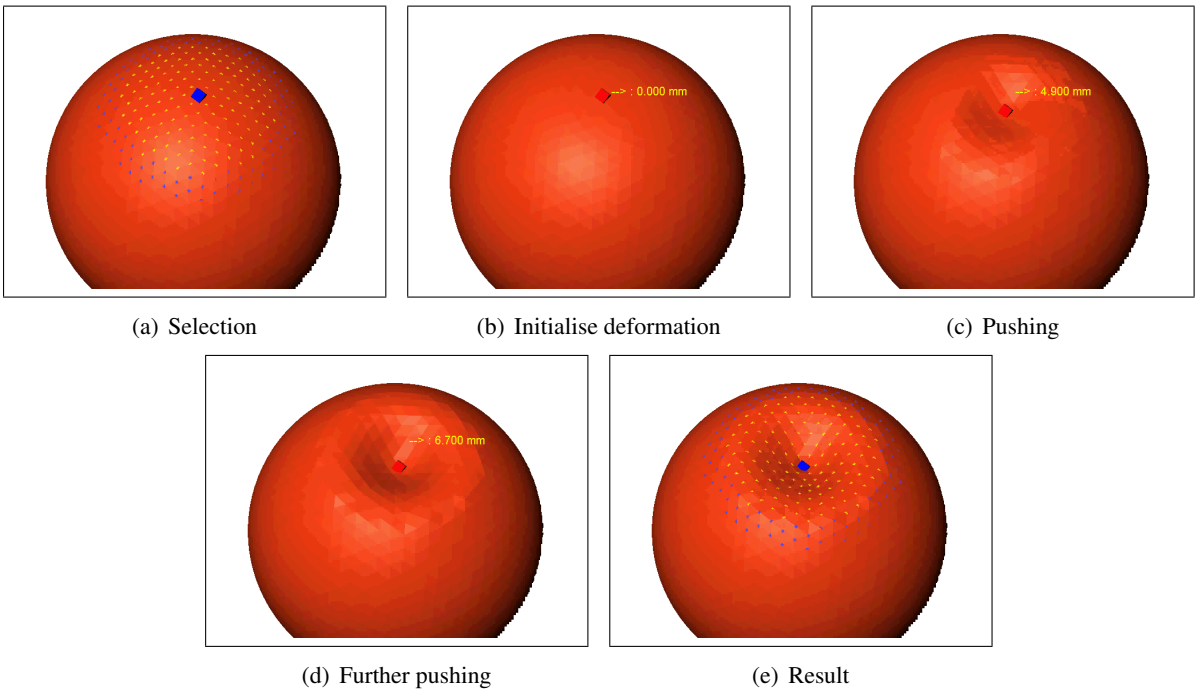
4.4 Modelling Aspects

Recall that the modelling framework should suffice the following constraints.

- A high precision is necessary.
- An intuitive user interaction is necessary to avoid long training periods.
- Both fine and large scale deformations need to be possible.
- The flexibility must be high.
- The user needs visual feedback about the measure of the deformations.

In the first instance, the vertex transformation ideas are explained. Then the properties of the edited meshes are mentioned. After that, the interpolation of the neighbouring vertices is regarded. There are several aspects that need to be considered.

4.4.1 Vertex Transformation



(a) Selection (b) Initialise deformation (c) Pushing

(d) Further pushing (e) Result

Figure 4.2: Pressing an indent into a sphere

The first vertex transformation allowed is a translation on the focal plane. The selected vertices, which are highlighted by a cube, are fixed on a plane at the current distance to the camera and the user can move them up, down, left and right from his point of view. The grabbed cube moves within the mouse cursor. This translation sometimes leads to unintentional results, because the user cannot see the depth of the object on a classic monitor. Therefore, it is hard to recognise the true shape of the object. Only the monocular depth criteria are available. These are typical perspective problems, and skilled users know to handle them, since they occur likewise in CAD environments.
The second transformation method is a translation along the mean normal of the selected vertices. The mean normal is calculated by summing up all vertex normals and normalizing the resulting vector. The

user translates all selected vertices along the direction of the mean normal. This method is displayed in the image series of Figure 4.2. After the ROI selection the vertex is translated in direction to the centre of the sphere. After releasing the mouse button, the ROI selection is still enabled, in order to adjust the deformation even more precisely. The yellow number shows the extent of the deformation. This method is very useful for the main use case of the software as described in Section 1.1. When the assembly test failed because of a very small penetration at one side of the object, the user can deform the collided side a little bit to the inside and redo the assembly test. Therefore, he marks some of the vertices at the objects side and deforms them contrary to the direction of the mean normal. This can be done rapidly and the assembly test can be repeated.

The third transformation is a rotation. Every rotation takes place around an axis. If the user activates the rotation mode, in the first instance, he has to place the rotation axis in the scene. He drops the axis on the focal plane and after that, he can rotate and translate it into the desired orientation. When the axis is positioned, the user grabs one of the selected vertices and turns all selected ones around the axis. In Figure 4.3 the vertex transformations are pictured. The three methods are indicated by the red arrows, the green line indicates the rotation axis. The double arrow in the middle signifies the transformation along the normal, the crossed arrows on the right side of the image denotes the transformation on the focal plane.

These three methods enable a high flexibility to deform an object while preserving the option of intuitive

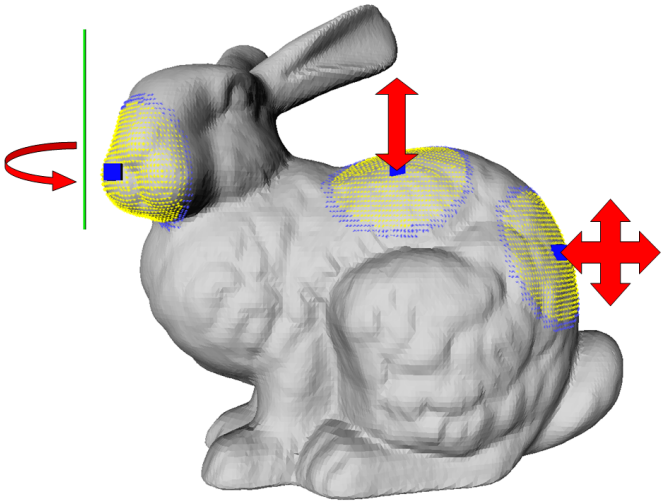


Figure 4.3: The vertex transformations

deformations. All methods are supported by an on-screen display of the current deformation measure. In the first case, the length of the translation of the selected vertices is displayed, but the true direction is not clearly visible because of the perspective problem mentioned above. In the second case, the direction is known, and therefore, a very precise deformation is possible. And in the third case, the rotation angle is displayed on the screen. All three deformation degrees can be controlled by the user. He specifies the magnitude of the transformation, that is executed when the mouse is moved for one pixel, by changing a slider in the dialog window. This combination of visual feedback and measurement control allows both a

very high precision and very fine deformations.

4.4.2 Meshes

The editing software is able to load data of 3D models. This data is stored in the OpenInventor file format, which contains the vertex data as a list of triangles. The data is processed in order to build a mesh, because it is necessary to know the adjacent vertices of each vertex. A mesh can be seen mathematically as a graph. Thus, it is obvious that there are plenty algorithms and methods that can be used on these structures. The implementation according to this thesis takes advantage of this fact. Constructing the mesh is done in a so called preprocessing step. When the user enters the deformation mode, the 3D object data is separately stored in order to enable a rapid access to the connection properties of each vertex. The deformation object class contains a list of all vertices and one of all triangles. In addition, each vertex knows its adjacent vertices and triangles and the same for each triangle vice versa.

4.4.3 Interpolation of the Neighbourhood

If only the selected vertices are deformed, on the objects' surfaces sharp edges and bendings may occur. In order to avoid these effects, the neighbourhood around the deformed area can be interpolated, so that the transition between moved and fixed vertices is kept smooth. To do this, two methods have been developed, a straight and simple one, and a more robust and solid one. At first, the straight one is introduced, and after that, the second one, which is based on Laplacian mesh editing, is described in detail.

4.4.3.1 Direct Distance Weighted Translation

For the two translation transformation schemes, a simple way to interpolate a smooth transition has been realised. This method is similar to a method for animation presented in [Par02]. As mentioned earlier, the user defines a ROI which contains handle, neighbour and fixed vertices. During one deformation step, all handle vertices are translated in the same direction for the same displacement. The fixed vertices do not move. In other words: the handle vertices are translated by the factor 1 and the fixed vertices by the factor 0. It is obvious that the neighbour vertices will move in the same direction, but with a smaller magnitude. Each neighbour vertex located between them has to be assigned to a factor. This factor depends on the distance of the corresponding vertex to the handle vertices. The ratio

$$r(\vec{x}) = \frac{|\vec{x} - \vec{x_h}|}{\min(|\vec{x_f} - \vec{x_h}|)}$$

is defined as the distance of the neighbour vertex $\vec{x}$ to the next handle vertex $\vec{x_h}$ divided by the minimal distance of a fixed vertex to a handle one. For this ratio, it is valid that

$$r(\vec{x}) \in [0, 1] \quad \forall \quad \vec{x} \in N,$$

where N is the set of neighbour vertices, if the neighbourhood is defined by a radius around the handle vertices. Using this ratio directly, would lead to a linear behaviour which is not smooth in an ordinary

sense. Thus, the ratio is mapped to a Gaussian. The searched factorisation function $f : \mathbb{R}^+ \rightarrow [0, 1]$ is defined by

$$f(r(\vec{x})) = e^{-(a*r(\vec{x}))^2}. \tag{4.1}$$

With the factor $a = 2$, the function gets the value $f(1) \approx 0.02$, which means that the function maps the ratio to a smooth transition, see Figure 4.4. There are several alternative functions conceivable. In Rick Parent's book, the factorisation function depends on the number of edges between the vertices and not on the Euclidean distance [Par02]. Such definitions require a regular mesh which is not available in this case. Using the Gaussian (4.1) as transition function led to good results, and therefore, it was used in the implementation. Thereby the neighbour vertices were determined by a user controlled frame selection, some problems can occur, because the ratio can exceed one. Using the Gaussian, this problem could be overcome since it asymptotically approaches the x-axis. But this way to interpolate the vicinity of the handle vertices has its limitations, if the transformation is more than just a translation. This induces the implementation of a Laplacian mesh editing tool.

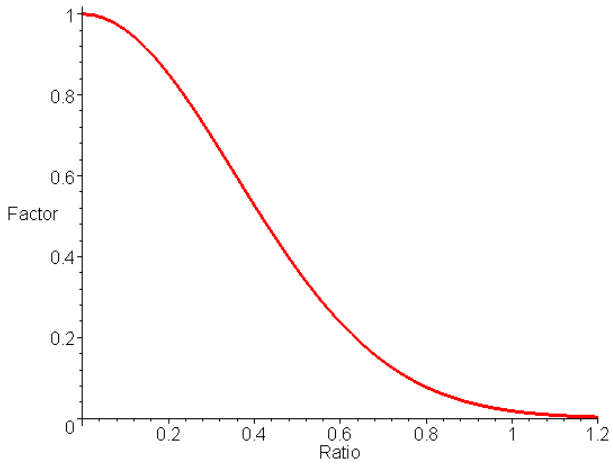


Figure 4.4: Plot of the direct distance weighted translation function $f(r(\vec{x}))$

4.4.3.2 Laplacian Mesh Editing

As mentioned in Section 2.4, the main idea of Laplacian mesh editing is restoring the absolute coordinates from the differential coordinates of a mesh. If the differential ones are known, the user needs to define at least one absolute coordinate to reconstruct the whole mesh. The idea of editing the mesh is to fix a group of vertices at their original positions and another group at new positions. Then the reconstruction is done as a minimisation.

Firstly, we consider the Laplacian matrix L of the whole mesh. According to the number of vertices, this matrix is quite big but sparse. Regarding Section 4.3.2, the user defines a ROI. Through this localisation, the size of the system to be solved can be reduced. This ROI contains three kinds of vertices: handle, neighbour and fixed vertices. Only the vertices belonging to one of these groups are taken into account. The number of the fixed vertices should be large enough to keep the minimisation stable. This is reached by building a ring of fixed vertices around the handle and neighbour vertices. According to [SLCO$^+$04],

Area	**Number of vertices**
Mesh V	n
Submesh V' (only ROI)	m
handle vertices $\vec{h}$	h
neighbour vertices $\vec{g}$	g
fixed vertices $\vec{f}$	f
Matrix	**Size (rows,columns)**
Mesh Laplacian L	(n, n)
Submesh Laplacian L_{ROI}	(m, m)
Enhanced Submesh Laplacian L_E	$(m + h + f, m)$
Conditions	
$m = h + g + f$	
$m \ll n$	
$f \approx 0.1 \cdot m$	

Table 4.1: Vertex distribution in ROI

the radius of the ring of fixed vertices should be around 10% of the radius of the editing area. In this implementation, the user adjusts the radius with a slider. After the user has defined the ROI, he is able to transform the handle vertices as described above.
Once this is done, the equation system in the form $A\vec{x} = \vec{b}$ is constructed. Therefore, the Laplacian of the submesh V' is constructed, containing only those vertices located in the ROI. This leads to a significantly smaller matrix L_{ROI} which is symmetric and has the size m, where m is the number of vertices in the ROI. According to this matrix we build the vector δ containing all differential coordinates of the vertices inside the ROI. After that, L_{ROI} gets attached by rows including the handle vertices and fixed ones. These rows have the form $(0, \ldots, 0, 1, 0, \ldots, 0)$ with the 1 at the kth position, if the correspondent vertex has the index k in the mesh. In addition, each of these rows can be weighted with different values w_h and w_f, corresponding for the handle vertices and the fixed ones. The enhanced matrix L_{ROI} is called L_E. The vector on the right side is called δ'. It contains the differential coordinates of the m vertices attached to L_{ROI} and the new absolute positions of the handle vertices plus the old positions of the fixed vertices. See Table 4.1 for an overview of the size of the equation system that has to be solved.
When the user drops the handle vertices to a new position, the equation system

$$L_E\vec{x} = \delta'$$

is built. After that, the new positions of the neighbour vertices are computed. This computation is done by solving the equation system. The system is over-determined and so, as mentioned in Section 3.6, it is best to use the least squares method. Therefore, the error function

$$E(V') = \|L_{ROI}\vec{x} - \delta\|^2 + \sum_{i \in H} (w_h(x_i - h_i))^2 + \sum_{j \in F} (w_f(x_j - f_j))^2 \tag{4.2}$$

is defined, where w_h and w_f are the weights for the handle and respectively the fixed vertices. How these weights are determined, is explained in Section 5.3. H and F are the sets of the according indices, hence, the sum variables i and j indicate the selected respectively the fixed vertices. Note that the x_i and x_j are elements of $\vec{x}$. The equation system is not quadratic any more, it is overdetermined and so the error function $E(V')$ is minimised over the variable $\vec{x}$. As also mentioned in section 3.6, the Gauss transform is executed as follows:

$$L_E\vec{x} = \delta'$$
$$L_E^T L_E\vec{x} = L_E^T\delta'.$$

As $L_E^T L_E$ is symmetric and positive definite, the Cholesky decomposition is used for solving the system. We substitute

$$L_E^T L_E := M$$

and factorise

$$M = R^T R$$

into an upper and a lower triangular matrix R and respectively R^T. These matrices are used to calculate

$$R^T\xi = L_E^T\delta',$$

where ξ is an auxiliary vector computed by forward substitution. Finally,

$$R\vec{x} = \xi$$

can easily be solved by back substitution.

Summing up the equation system properties; the matrix L_E is sparse, for this reason the matrix $M = L_E^T L_E$ is still sparse. Through the symmetry and the structure of the Laplacian, M is also positive definite. Therefore, the Cholesky factorisation is possible and the decomposition can be done quite fast. It would be reasonable to use a free software library of sparse matrix algorithms in order to decrease the computation times for solving the sparse linear systems, but the usage of such a library is not possible in commercial applications at DaimlerChrysler. In this implementation, the classic algorithm for arbitrary occupied matrices is used and nevertheless the performance allows real-time interaction. The main advantage of this decomposition is the fact that the matrices M and L_E^T do not change during the editing process, because they only depend on the ROI. If the ROI is set, these matrices can be computed and stored. The two steps of forward and back substitution can be done in linear time. That makes the algorithm really fast and allows shape editing at interactive rates.

But what about the conditions of this minimisation? The Laplacian of a connected mesh is ill-conditioned, because it is not even regular. However, we must consider the matrix $M = L_E^T L_E$. Considering the condition number from equation (3.29):

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|,$$

and let us take a look at the norm of the matrix M. L_E is constructed from the Laplacian of a connected submesh and additional rows containing only one entry, the weight w. Each row and column of a Laplacian matrix sums up to zero. For this reason, we have

$$\|L_E\|_1 = \|L_E\|_\infty = w = \left\|L_E^T\right\|_1 = \left\|L_E^T\right\|_\infty.$$

This equality leads to

$$\|M\|_1 = \left\|L_E^T L_E\right\|_1 \leq \|L_E\|_1 \left\|L_E^T\right\|_1 = w^2,$$

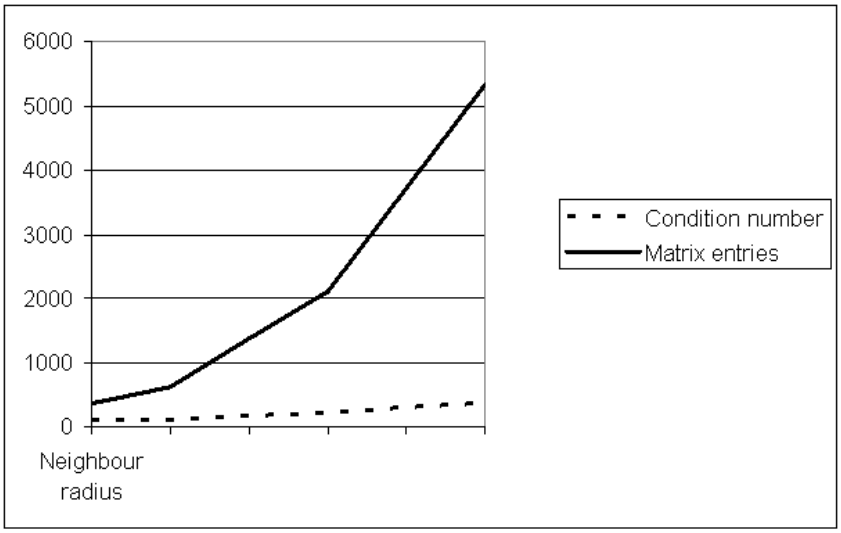
if equation (3.21) is taken into account. Regarding equation (3.22), it is valid that

$$\left\|M^{-1}\right\|_1 \leq \|M\|_1^{-1}.$$

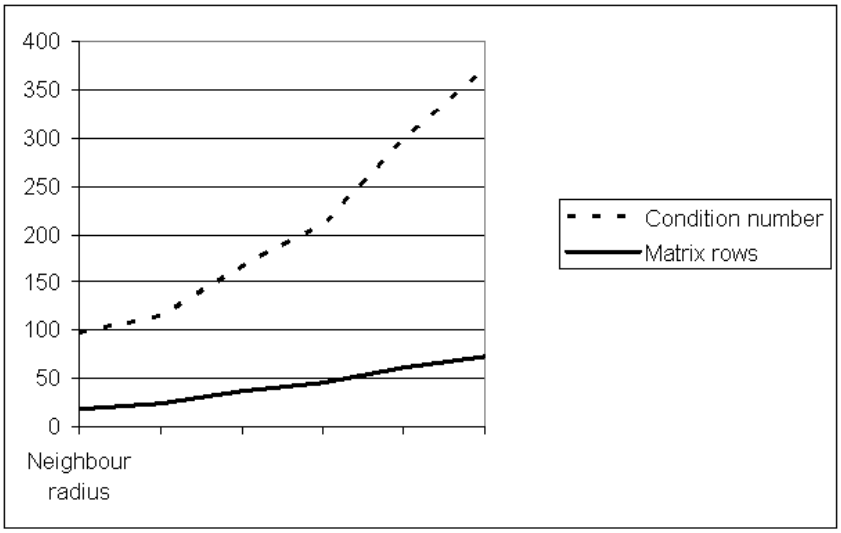
However, due to the fact that

$$\|M\|_1^{-1} \geq \frac{1}{w^2},$$

it is not possible to estimate the condition number in this way. Therefore, the condition number has been



(a) Condition and entries



(b) Condition and rows

Figure 4.5: Coherence between condition number and matrix size

computed for small matrices. Figure 4.5 shows the coherence of the condition number and the size of the matrix M. For a matrix A in relation to a small ROI (one handle, 44 neighbour and 28 fixed vertices),

the condition number is about $\kappa(A) = 373$. Both plots show that there is no extreme increase of the condition number and that the ratio between the condition number and the number of rows in the matrix is even almost constant. Hence, the condition number will not grow noticeably, and it can be said that the equation system is well-conditioned even for a larger ROI. An ill conditioning could only occur when an particularly unfortunate ROI has been chosen by the user. In this case, the "Undo"-button will help.

4.5 Problems and Disadvantages

Having examined the potential problem of the condition of the over-determined equation system, we can move on to the handling of degenerated triangles. A degenerated triangle has no neighbours and hence, the computation of its differential coordinate does lead to instabilities. If the mesh is not properly connected, the construction of the Laplacian matrix may fail. This problem might occur if the user is working with scanned data which has not been preprocessed or smoothed. In this application in the manufacturing process, triangulated CAD data is edited, thus the problem of degenerated data can be shifted to an external programme. These programmes can be controlled in order to avoid degenerated triangles. Hence, this problem is shifted to another type of application that is not handled in this thesis and can be solved without great effort.

Another problem may be that Laplacian shape editing does not work with point sampled data. The connectivity of the vertices must be known, in other words: a mesh must exist. However, the processed objects are all constructed with CAD programmes and hence, they need to be transferred to the VR framework "veo". Therefore, the objects are approximated by a mesh. These are no drawbacks because of the environment the application is used, but there is one disadvantage of differential coordinates: They are not rotation invariant.

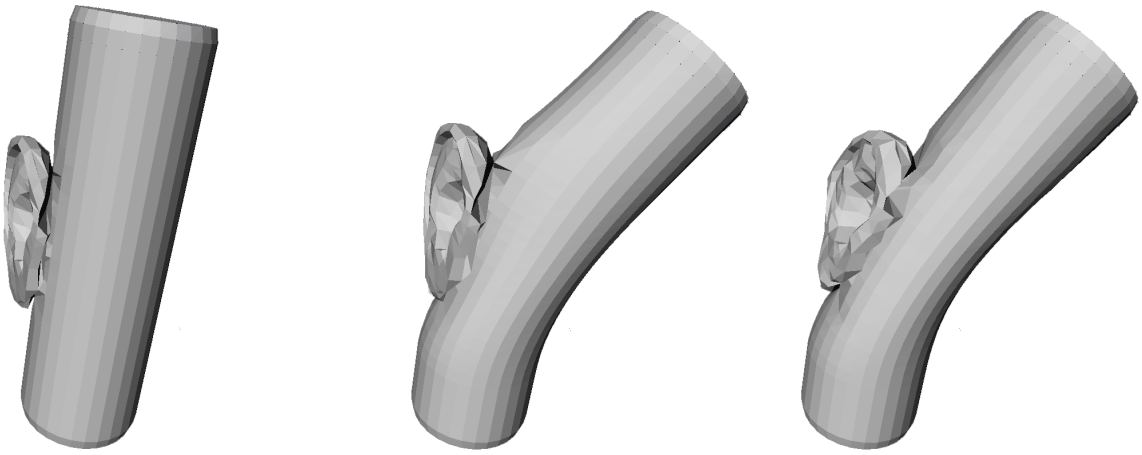
To solve this problem, there exist several approaches. In the surveying state of the art report [Sor05], Olga Sorkine outlines that the approaches to approximate the rotation of the differential coordinates. In each case the energy minimisation (4.2) function is extended to

$$E(V') = \|L_{ROI}\vec{x} - T\delta\|^2 + \sum_{i=m+1}^{m+h} \|w_h(x_i - h_i)\|^2 + \sum_{i=m+h+1}^{m+h+f} \|w_f(x_i - f_i)\|^2,$$

where T is a vector containing all the approximated local transformations T_i for each vertex i. These local transformations approximate the rotation of the corresponding differential coordinate δ_i. One approach has been done by Lipman et al. in [LSLCO05]. They execute the minimisation twice. At first, they compute the new coordinates without transformation, and then they calculate T_i from the received results by minimizing

$$\min_{T_i} \left(\left\|T_i x_i - x_i'\right\|^2 + \sum_{j \in N(i)} \left\|T_j x_j - x_j'\right\|^2 \right),$$

where x' are the new absolute coordinates derived from the first computation. This does not increase the computation time strongly because the matrix L does not change and so there is only one additional back substitution necessary. Olga Sorkine et al. [SLCO$^+$04] enhanced this scheme by constraining the transformation T_i. The free parameters of T_i are reduced to parameters describing an rotation with anisotropic



(a) Original model (b) Bending without rotation of detail vectors (c) Bending with rotation of detail vectors

Figure 4.6: Differences in bending strong details

scaling. This makes the transformation linear and hence it can be solved explicitly during the first step. See [Ale05, SLCO$^+$04, Sor05] for details about the construction and computation of T_i.

Another approach is used by Yu et al. and Zayer et al. [YZX$^+$04, ZRKS05]. They approximate the rotation of the differential coordinates, or the gradients in Yu et al.'s case, by propagating the handle transformation to the neighbour vertices. Zayer built a harmonic map ranging from 1 to zero depending on the mesh structure connection. Yu et al. stick to the euclidean distance of the vertices. In the implementation according to this thesis, the rotations are propagated as well. The chosen method lies between Sorkine's and Yu's method. It uses a distance function ranging from one for the handle vertices to zero for the fixed vertices, such as the harmonic map of Zayer et al. According to this factor, the rotation is interpolated by a spherical linear interpolation (slerp) for all vertices inside the ROI. Through the construction of the ROI along the connections of the mesh, the impact of the handle depends both on the mesh topology and on the euclidean distance. It is not necessary in this implementation to use a whole transformation matrix because of the trisection of the available handle transformations.

A rotation of details also occurs on exaltations of a single vertex from a plane. The main application of the software is a fine and precise translation on which no strong rotation of details occurs. Therefore, only the rotation deformation case is processed with a rotation approximation of the detail vectors. Since in industrial design, the details sometimes should not be rotated, this feature can be switched on and off by the user. In Figure 4.6 the differences are pictured. We can see that the ear is rotated in the left image (c). It keeps its orientation when the rotation is switched off like in the middle image (b). The trisection of transforms and the other modelling aspects have been patterned on the basic functionalities of a classic 3D modelling software. However, in order to avoid long periods of training, the user interaction is kept simple. The Laplacian mesh editing scheme supports this approach. In relation to the intended working environment, there is the need for some controlling mechanisms for the user. The user should be able to vary the deformations. Therefore, the weighting possibilities have been examined and extended. The next chapter gives a review of these features.

5 Effects of Weighting in Laplacian Mesh Editing

5.1 Introduction

As mentioned in the last chapter, the Laplacian mesh editing has to be extended in order to increase the user impact on the deformations. Therefore weighting schemes are introduced. In the following paragraphs, it is described how the Laplacian mesh editing scheme has been extended and alternative methods of weighting have been submitted to a careful examination. At first, some simple and well known examples for the usage of weights in general are described.

5.2 Common Applications of Weights

Weights are generally used in signal processing, for adapting the measurements to human needs. In the following, several good examples are described which can be found in [Wik06]. It starts with measuring the loudness of an audio signal. Here, a weighting filter is used to emphasise the frequencies around 3-6 kHz, which the human ear is most sensitive for, so that the measure of the loudness complies with the human perception. There is also a special physical unit called “phon”, that only considers the signal intensity at 1kHz. In this case, it can be spoken of a binary weighting scheme, where only one frequency is considered and all the others are blocked. In signal processing, the terms weighting and filtering can be swapped.
Another example is measuring the danger of radioactive γ-rays. Here, the strength of those ray frequencies which cause the most harm on humans is considered more intensive than the frequencies which do less harm. Due to this fact, the measure does not indicate the strength of the radiation, but its danger for human beings.
A comparable method is the weighting of the colour channels red, green and blue in order to compute the luminance of an RGB-signal. Due to the fact that the three base colours do not have the same luminance in the human perception, they are weighted differently in order to balance their impact on the luminance of the whole signal. The signal luminance

$$Y = 0,299R + 0,587G + 0,114B$$

is a weighted mean of the colour channel intensities R, G and B. These values correspond to the so called YUV-model. Without this weighting, it would not be possible to transform a colour signal into grey-scale images with the same contrasts.
These examples showed that one main aspect in weighting is to adopt the measuring to the human sensorial perception of signals. In the Laplacian mesh editing application, there is no need to transfer any signal into a human recognizable region, but there is a need for the human user to determine the geometrical impact or weight of some areas in the chosen ROI, in order to receive full control over the shape deformation. How this works is described in the following section.

5.3 Utilisation of Weights

There are two different approaches for the utilisation of weights. The first one adjusts the differential coordinates to the mesh structure of the shape that should be deformed. The second one weights the minimisation of the energy functional and for this reason, it has a strong impact on the deformation itself. These weights are attached to vertices directly, and therefore, this method is called vertex weighting. In the following, these two approaches are described.

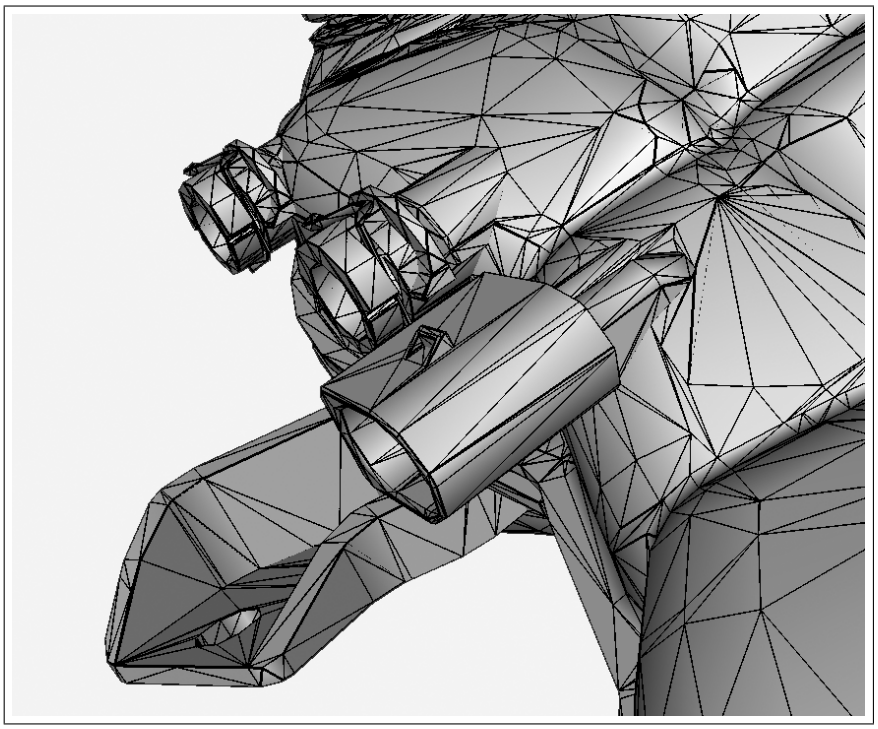


Figure 5.1: An irregular mesh

5.3.1 Differential Coordinate Weighting

First of all, let us recall the Laplacian mesh editing scheme in short in order to get a general idea where the weights are used. Suppose we have a mesh $M = (V, E)$ with n vertices V and a number of edges E. Recall equation 3.5

$$\delta_i = \vec{p_i} - \frac{1}{d_i} \sum_{j \in N(i)} \vec{p_j},$$

which describes the differential coordinate of a vertex i. This coordinate is constructed as the difference of the position vector and the mean over all neighbouring vertices j. The vector $\vec{p}$ contains the absolute coordinates of the vertices. As written in Section 3.4, this equation is extended with weights to

$$\delta_i = \frac{1}{\sum w_{ij}} \sum_{j \in N(i)} w_{ij}(\vec{p_i} - \vec{p_j}). \qquad (3.7)$$

This opens many different possibilities for choosing the weights w_{ij}, which can be seen as a function $w : E \to \mathbb{R}^+$, where E is the set of edges and each edge is defined by the two vertices with the indices

i and j. Therefore, this extension can be seen as the step from an unweighted graph to a weighted graph. Considering meshes, we need to add some information about the geometry to the graph. This is the idea behind the weighting schemes. An unweighted graph approximates the mesh satisfactorily, if and only if all edges have uniform length. Such a mesh is called regular. Often, 3D shapes are triangulated into nearly regular meshes. In this case, the impact of the weighting is not that big. However, in the main application of this thesis triangulated CAD data is used. Such data is not triangulated regularly as shown in Figure 5.1.
In order to avoid problems with this missing regularity, there really is a need for a well chosen weighting scheme. The uniform weighting

$$w_{ij} = w_{UNI} = 1$$

is just *topologically* correct, because only the connections inside the mesh are considered but not the geometrical properties of the triangulation. However, for editing applications, geometric correctness is necessary. The first and simplest approach for weighting is using the edge length l as the variant factor for defining the weights. Both Gabriel Taubin and Koji Fujiwara [Tau95, Fuj95] use powers of the edge lengths. This leads to $w_{ij} = l(i,j)^{\alpha}$, where α is a scalar. They both got the best results with $\alpha = -1$. That leads to the already mentioned scale-dependent umbrella operator with

$$w_{ij} = w_{SDU} = 1/l(i,j).$$

In addition, the effect for $\alpha = 1$, leading to

$$w_{ij} = w_{LEN} = l(i,j),$$

has been considered as well.
Marc Alexa mentioned in his tutorial talk [Ale05] two other weighting schemes which are both based on the angles of the adjacent faces: tangent and cotangent weighting. Remembering the angle notations from Figure 3.4, there is the tangent or mean value weight

$$w_{ij} = w_{TAN} = \frac{1}{l(i,j)}(\tan\frac{\alpha_{ij}}{2} + \tan\frac{\alpha_{ij-1}}{2}),$$

which derives from the mean value coordinates of [FKR05]. Michael Floater defines the mean value coordinates of a star shaped polygon as tangent weighted average of all edges inside the kernel of the polygon. Starting from the proposition that the integral over all the normals of a sphere becomes zero, he constructs the tangent weights. But according to the kernel approach, the tangent weights are only exact for convex polygons. However, they are examined about their behaviour in this thesis.
The other weight mentioned by Marc Alexa is the cotangent weight

$$w_{ij} = w_{COT} = \cot\gamma_{j-1} + \cot\beta_j,$$

deriving as explained in Section 3.4.1 from [EDD+95]. Table 5.1 illustrates an overview of the existing weighting schemes. A general survey can be found in [Tau00].

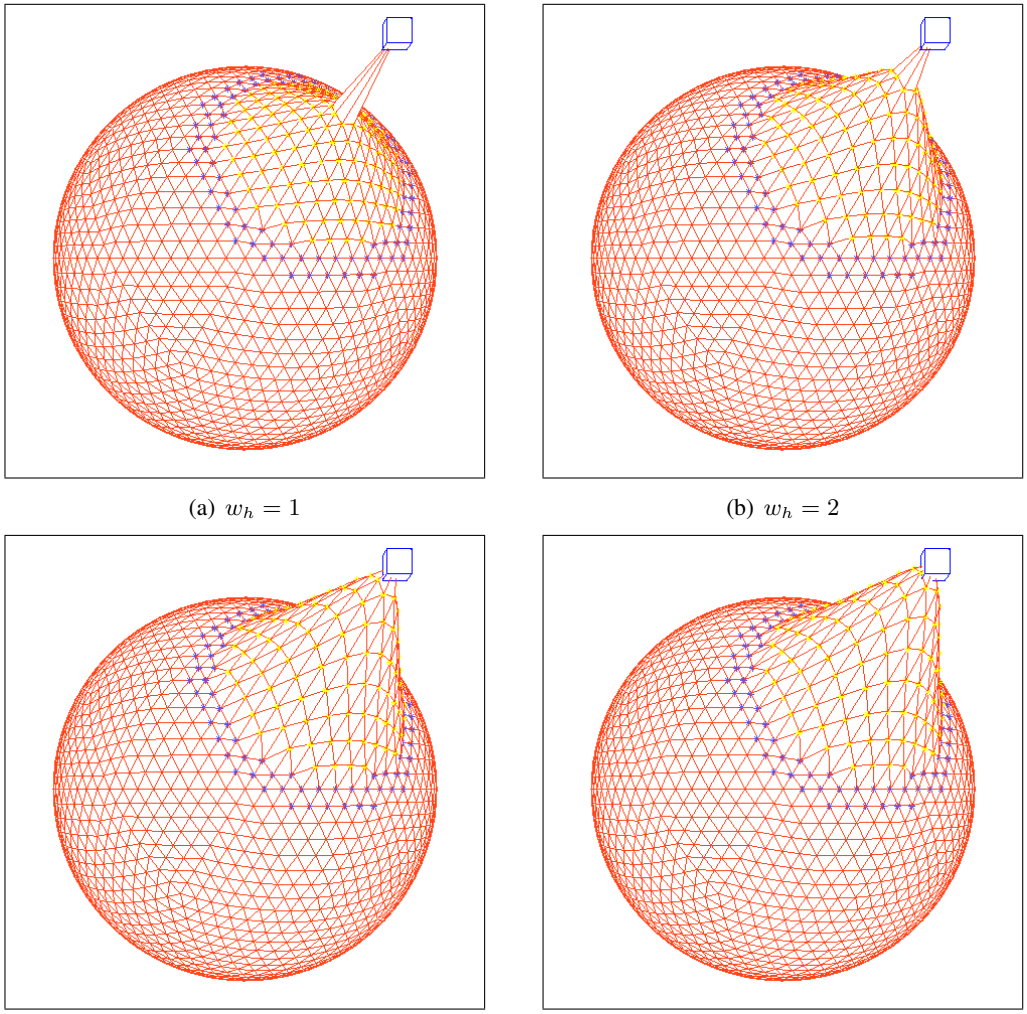
Name	**Definition**
Uniform or umbrella	$w_{UNI} = 1$
Scale-dependent umbrella	$w_{SDU} = 1/l(i,j)$
Edge length	$w_{LEN} = l(i,j)$
Tangent	$w_{TAN} = \frac{1}{l(i,j)}(\tan\frac{\alpha_{ij}}{2} + \tan\frac{\alpha_{ij-1}}{2})$
Cotangent	$w_{COT} = \cot\gamma_{j-1} + \cot\beta_j$
Handle vertices weights	$w_h \in [1, 20]$
Fixed vertices weights	$w_f \in [1, 20]$

Table 5.1: Overview of weights

5.3.2 Vertex Weighting

These are the different possibilities to influence the definition of differential coordinates. However, there is a further option of weighting. Remembering the addition of rows to the submesh Laplacian L_{ROI} in order to build up the matrix L_E for the linear system $L_E\vec{p} = \delta'$. As mentioned in section 4.4.3.2, the attached rows of L_E can be multiplied by a weighting factor w. Each row has only one entry at the position according to the index of the vertex, therefore, the weighting factor has an direct impact on the specified vertex. In the implementation, it is possible for the user to control the weight w for the handle vertices and for the fixed vertices. This restriction to weighting in these two groups makes the editing application easier to handle on the one hand, and on the other one, it offers a simple controlling mechanism for the transitions inside the ROI. In the following, w_h stands for the handle weights and w_f for the fixed vertices weights. For better usability the user changes these weights with a slider and hence, the weights are kept a natural number between 1 and 20, except in the automatic weight setting function which will be described later in Section 5.5.

The weights control the smoothness of the boundary from their region in the ROI to the neighbour vertices. The greater the value, the smoother the boundary. Due to the fact that triangulated surfaces always have C^0 continuity, the weights cannot be directly mapped to a degree of continuity. However, it can be said that a weight $w = 1$ leads to C^0, and $w = 20$ to a C^2 continuity. The continuity depends additionally on the transformation of the handle vertices. The user is able to deform the surface so strongly that all transitions could become sharp edged if a small amount of vertices does not allow any smoothing. The visual impact of these weighting parameters is much bigger than the one of the weighting of the differential coordinates. Figure 5.2 and Figure 5.3 show this visual effect. The image captions list the corresponding weights. The blue cubes at the top of the peak highlight the handle vertices. In Figure 5.2, the sphere is rendered as a mesh and the ROI is visible. The vertices which are marked yellow belong to the neighbourhood and the purple marked ones belong to the ring of fixed vertices. This sequence of images shows the impact of the handle weights. Setting $w_h = 1$ leads to a peak of only one vertex while the rest of the sphere remains nearly unchanged. Increasing the weight smooths the peak into a bump. As Figure 5.3 is rendered with flat shading, the boundary conditions are easy to detect. Here, the weights for the fixed



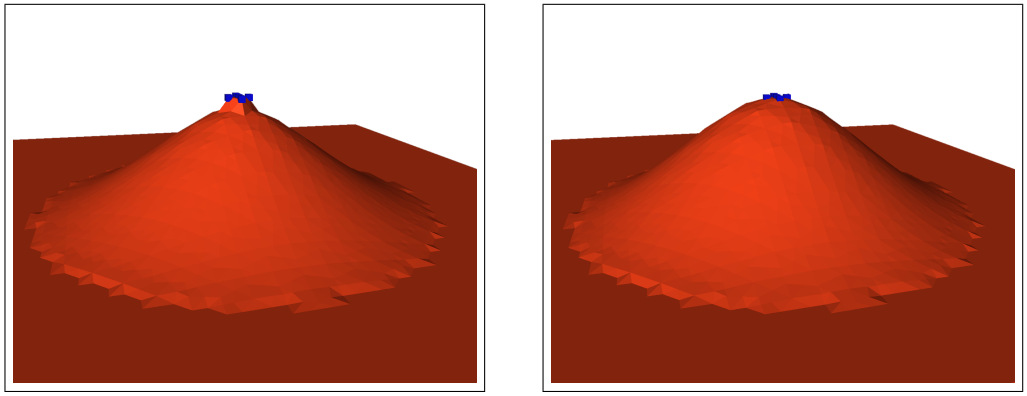
(a) $w_h = 1$ (b) $w_h = 2$

(c) $w_h = 5$ (d) $w_h = 20$

Figure 5.2: Exalting one vertex from a sphere with different handle vertices weights on constant fixed vertices weight $w_f = 1$

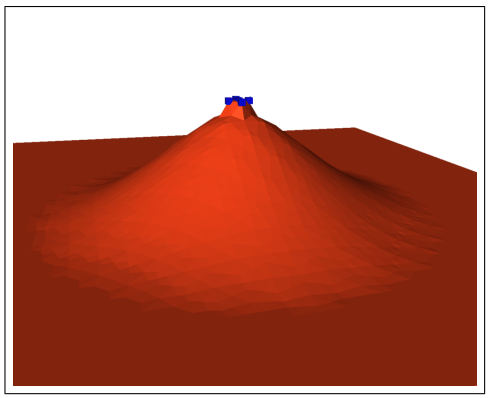
vertices do not remain constant and so, different transitions at the boundary of the peak appear. Both image series show the strong influence of the vertex weights on the deformation. This influence allows to control the deformation for the user. Imagine the following use case: The design engineer wants to simulate the mounting of several plastic units. He knows that a mechanic easily can press the units if they have to be pushed through a small gap. Imagine a large side area of a plastic tank which can be pressed by hand for some millimetres. In order to simulate the pressing, the design engineer deforms the unit while setting the weight in a way that simulates the behaviour of the plastic material. He selects some vertices at the centre of the side where the mechanic would touch the tank. If the ROI is set to the whole side of the tank so that the edges remain fixed, the weights should be set as follows. The handle vertex weight is set to the maximum because the transition from the centre to the border should be smooth. The fixed vertices weight is set small because the plastic is bent sharper at the edges of the tank. This imaginary application demonstrates the advantage of vertex weighting.

The interval $[1, 20]$ is chosen by heuristics. The upper boundary 20 is the limit of the impact by the

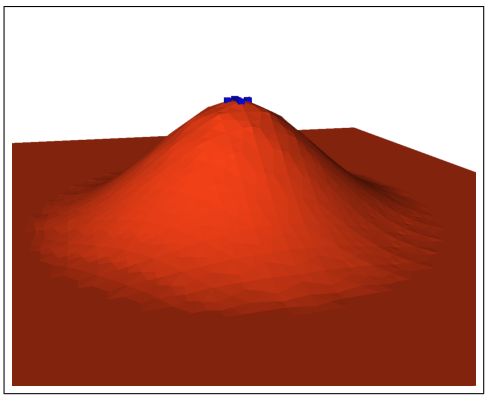


(a) $w_h = 1, w_f = 1$

(b) $w_h = 20, w_f = 1$



(c) $w_h = 1, w_f = 20$



(d) $w_h = 20, w_f = 20$

Figure 5.3: Exalting five vertices from a plane with different vertex weights

weights. Figure 5.4 shows a weight of $w_h = 200$. The image shows that it is not possible to significantly increase the weighting effect in relation to $w_h = 20$ in Figure 5.2(d). The chosen interval led to good results, despite the fact that Zhou et al. used a weighting in $[0, 1]$ and had with 0.2 the best results [ZHS+05]. In addition, Figure 5.4 demonstrates the limitations of the vertex weighting scheme. A further increase of the handle weight does not lead to a deformation in the shape of a dome. However, these limitations are not relevant to the main application field of this software. For controlling the transitions between areas of fine deformations and their surrounding regions, the vertex weighting scheme realises an easy and intuitive tool. It is kept simple by enabling only two sliders which change the value of each weight. Hence, the user is able to learn rapidly how the weights need to be adjusted in order to reach the intended results.

5.4 Effects of Weighting

As seen in Figure 5.3, weighting the handle or fixed vertices has a strong impact on the surface continuity. But, what about the impact of the differential coordinate weights?
In order to answer this question, consider Figure 5.5. Here, one vertex is pulled for 7mm from a thin plate. The ring of fixed vertices is placed about 12mm around the handle vertex. The two images indicate the difference between uniform and cotangent weighting with regard to the smoothness of the neighbourhood.

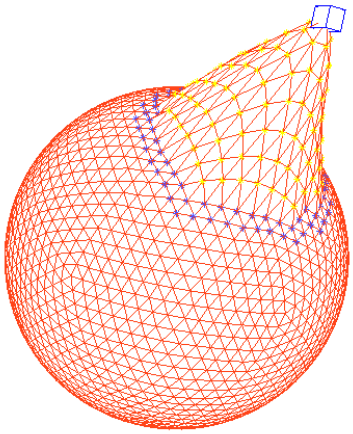
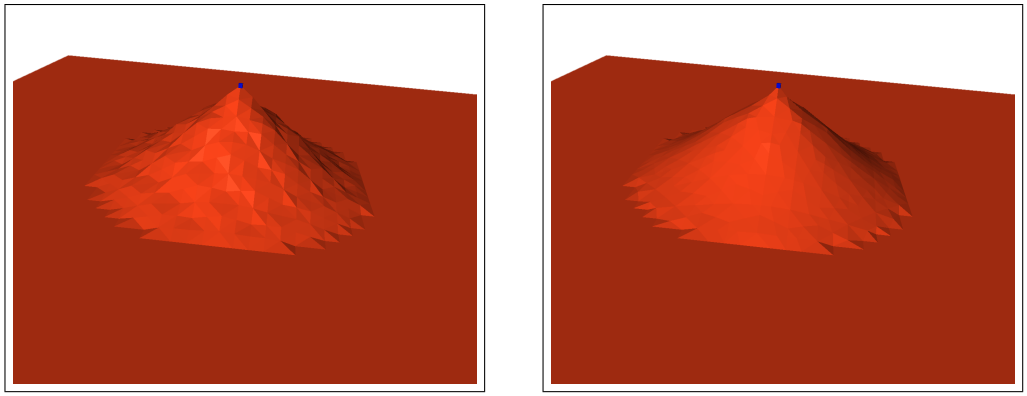


Figure 5.4: Exaggerated weighting



(a) Uniform (b) Cotangent

Figure 5.5: Exalting one vertex from a plane with different weighting

Looking at the side of the peak, the smoothing effect is obvious. But where does it come from?

To answer this question let us look at the differential coordinates for the vertices on the plate. A plate is flat, and hence there is no detail. The detail vectors or differential coordinates should be zero vectors for each vertex. Let us consider the mesh structure of the plate the vertices are pulled from in Figure 5.6. There are four different cases of vertex neighbourhoods. The vertices are emphasised by blue squares and in addition, they are numbered so that the corresponding differential coordinate δ_i has i as the index. The coloured areas around the vertices indicate the faces which are inside the one-ring and therefore are considered in calculating the detail vector δ_i. In the cases $i = 2, 3, 4$, the differential coordinate is a zero vector, independent of the kind of weight used. This can be deduced by the fact that the polygons around the vertices are all rotation symmetric. From this symmetry, it follows that every face has the same edges and angles. Due to the fact that the edge lengths and angles are considered in the computation of the differential coordinate weights, the weights of one vertex are all equal. The differential coordinates are defined by the sum of all edge vectors around these vertices in equation (3.7). This sum results therefore in the zero vector.

However, when $i = 1$, and the polygon is not rotation symmetric, we have a different situation. For

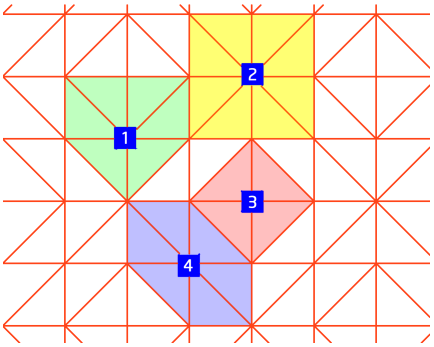


Figure 5.6: Mesh structure of plate

cotangent weighting, the four adjacent edges building a cross are weighted with $w = 2$, because all considered angles have $45°$ and $\cot 45° = 1$. The two transverse edges are weighted with $w = 0$, because the angles considered are $90°$ and therefore, the cotangent is zero. Due to the fact that the weighted edges build a cross, they even out each other. This leads to a zero detail vector $\delta_1^{COT} = \vec{0}$. If all edges are weighted uniformly, the detail vector is non-zero, because the two transverse edges are not weighted with zero like in the cotangent case. Thus, the detail vector directs along the bisecting line of the two transverse edges and is one third of the cross edges length long. For this reason, it is valid that $\delta_1^{UNI} \neq \vec{0}$. In regard to the continuous surface of the plate, the detail vector of all vertices should be zero. Hence, in this case, the use of the cotangent weight compensates the drawback of the irregular meshing completely. Examining this behaviour for the other weighting schemes leads to the following. The tangent weighting

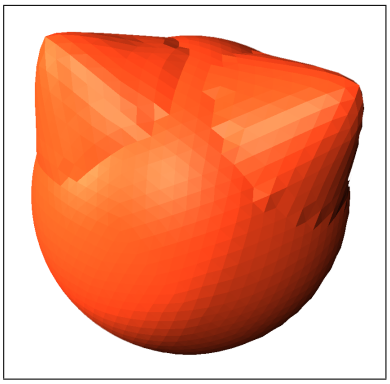
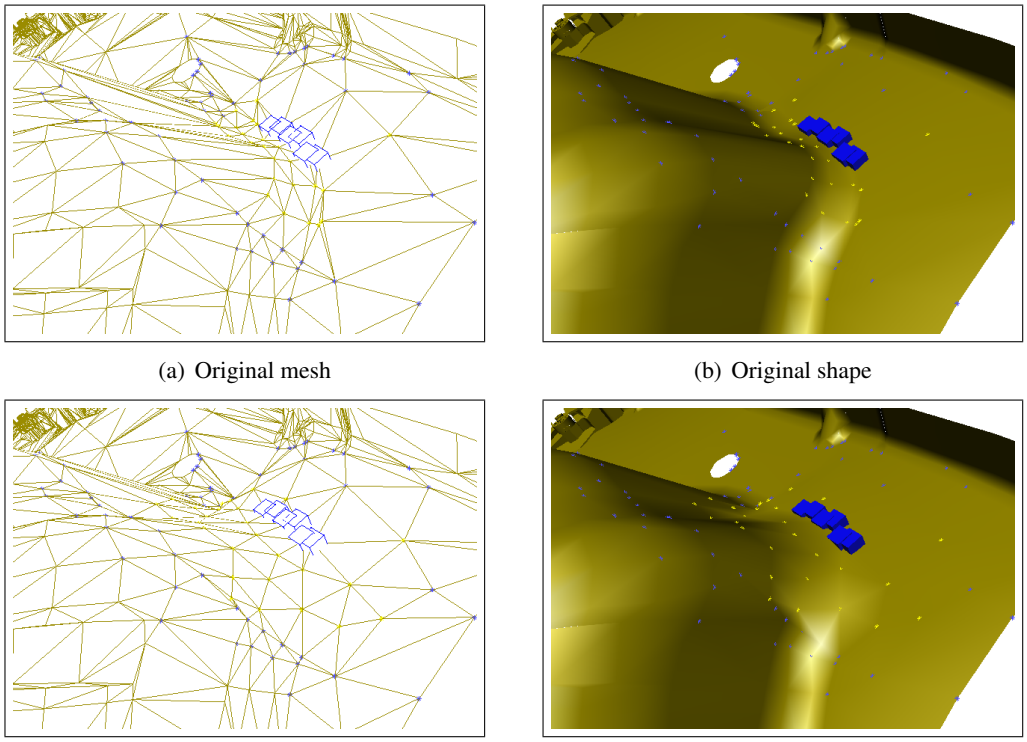


Figure 5.7: Two exalted vertices from a sphere

is also incorrect in the case $i = 1$, but the detail vector is about the factor $\tan 22.5° \approx 0.414$ smaller than with uniform weighting, and therefore, the error is significantly smaller. The parameter w_{LEN} even increases the error, because the two transverse edges are longer than the other ones, and therefore, they are weighted in a more intense way. This fact leads to an error, which is slightly bigger than $\sqrt{2} \approx 1.41$. If the edge length is used as weight with a negative power, like in the scale dependent umbrella weighting approach w_{SDU}, the error is reduced by the factor $\sqrt{2}$. Hence, there is a direct coherence between the power of the edge length in the weighting scheme and the error factor. However, this behaviour is only valid for this irregular triangulated coplanar surface. There is no prediction possible for arbitrary triangulated surfaces. This argumentation is supported by the results shown in Figure 5.7. Here, two

vertices are pulled out of a sphere which is triangulated regularly. There is no difference recognizable between the two weighting schemes. In this example, all edges in the mesh have the same length, and in addition all faces have the same corner angles, and therefore, all weights of the edges are equal and the weighting has no effect.

This behaviour is important for working with triangulated CAD data. Here, the meshes are not regular.



(a) Original mesh (b) Original shape

(c) Deformed mesh (d) Deformed shape

Figure 5.8: Remeshing effect after fine deformation with cotangent weights

It seems obvious to use a weight through which the differential coordinates approximate the geometry as good as possible. In Figure 3.3 it has already been shown that the use of uniform weighting can produce differential coordinates that do not describe the geometric detail properly. However, the cotangent weights have an additional effect.

The cotangent weight performs a kind of *remeshing effect*. This means that a deformation of the shape using the cotangent weight in the definition of the differential coordinates makes the deformed mesh much more regular than the original one. In Figure 5.8, this effect is pictured. The upper images show the original data and the lower row shows the deformed shape. All vertices which belong to the neighbour region inside the ROI are marked yellow. Especially, the yellow marked vertices below the blue cubes at the rounded edge are displaced in a way that makes the triangulation much more regular. This remeshing is performed only by displacement of vertices. This effect is positive because it smooths the deformed surface additionally. The triangulation becomes more regular. A regular mesh has the advantage that even unweighted differential coordinates approximate the detail precisely. The computed geometric weights such as cotangent and tangent weighting become smaller for each edge and therefore, the numerical stability of the minimisation increases.

However, the remeshing effect of cotangent weighting does not need to be positive at all because this remeshing can change the shape accidentally. Especially when fine deformations are executed in order to preserve a given structure of a unit, this remeshing effect might destroy the shape by displacing the neighbour vertices too strongly. An unwished smoothing then occurs. The examinations have shown that the use of the cotangent weights for the differential coordinates is not mandatory. This fact is contrary to the statements in literature [Ale05, Sor05]. In most of the papers it is written that the cotangent weights do approximate the details best. That is true, but the effect the cotangent weights by remeshing the surface inside the ROI has yet not been considered. The resulting remeshing might even distort a shape, especially if it is tessellated strongly irregular. However, in this implementation the negative effects can be avoided by the use of other weights such as umbrella weighting. Once again, the control is given to the user. The user decides which weighting scheme is used and therefore, he controls whether a remeshing effect should occur or not.

The last paragraph explained the effect of weighting the edges in the computation of the differential coordinates in relation to the regularity of the triangulated mesh. Now, we will study the behaviour of the vertex weighting. To do this, consider the minimisation function (4.2). We have

$$E(V') = \|L_{ROI}\vec{p} - \delta\|^2 + \sum_{i \in H} (w_h(p_i - h_i))^2 + \sum_{j \in F} (w_f(p_j - f_j))^2,$$

where w_h and w_f are the weights that are examined in detail. h_i and f_i are the new vertex positions of the handle vertices respectively the kept positions of the fixed vertices, while H and F are sets with the according indices. The squares guarantee that each part is positive and hence each part must become minimal. Therefore, the minimisation can be splitted into the three parts

$$\begin{aligned} \|L_{ROI}\vec{p} - \delta\|^2 &\to \min, \\ \|w_h(p_i - h_i)\|^2 &\to \min \qquad (i \in H), \\ \|w_f(p_j - f_j)\|^2 &\to \min \qquad (j \in F). \end{aligned} \tag{5.1}$$

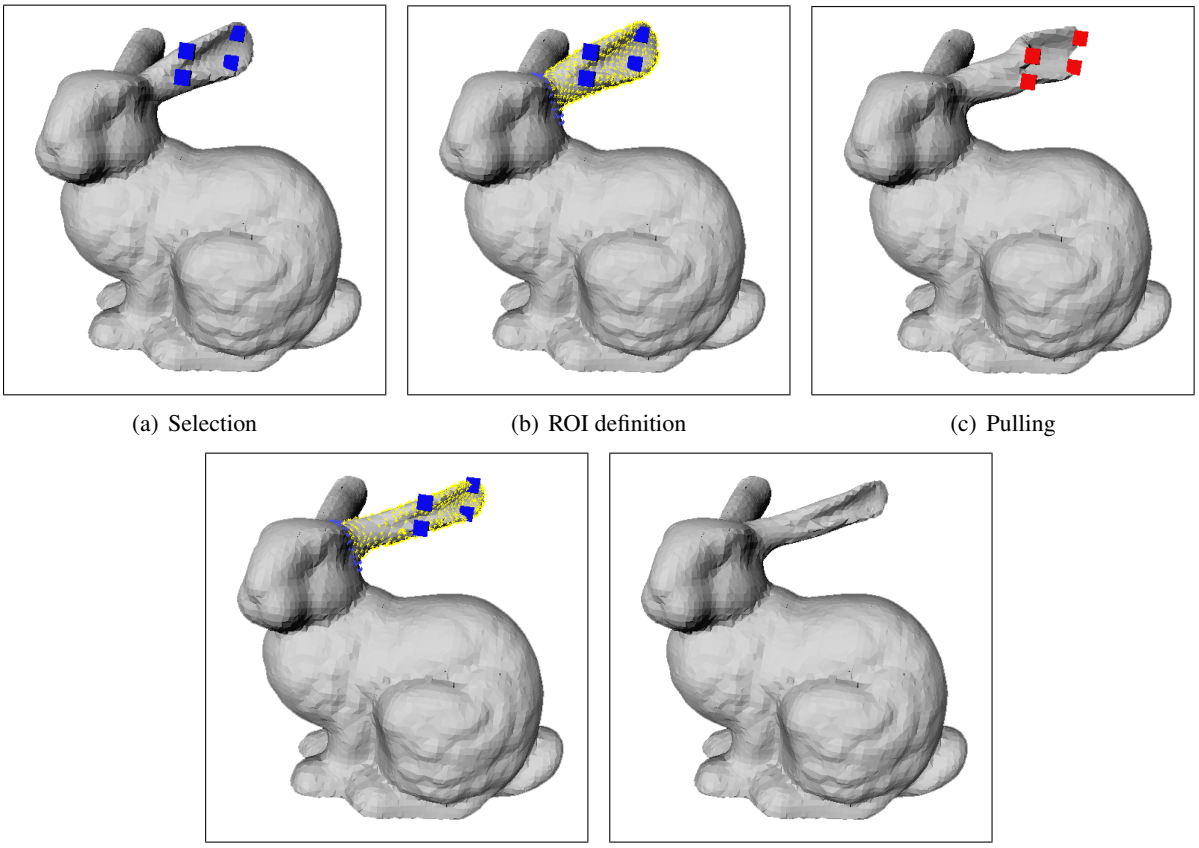
This representation shows how the weights w_h and w_f influence the solution. To approximate the impact of the weights, it is necessary to check the size of the interval they are used in. Considering an error value ϵ for each element of the solution vector $\vec{p}$ with

$$\epsilon = |\tilde{p}_i - p_i|,$$

where $\tilde{p}_i$ is the "correct" position, and p_i the result of the minimisation. Hence, the frequency of this error value must be checked. In the first part, there is the Laplacian L_{ROI} with the size $m \times m$. But each row has only $d + 1$ entries, if d is the valence of the vertex. Due to this fact, ϵ appears $m(d + 1)$ times. In the remaining two parts, the error value ϵ appears for each element of H respectively F. In conformity with the definitions from Section 4.4.3.2, they occur f, respectively h times and are multiplied with their weights. According to the condition from Table 4.1, after what $f \approx 0.1m$, w_f needs to be $10 \cdot (d + 1)$ times as large as the mean valence of the mesh in order to achieve the same influence on the minimisation as the Laplacian.

These observations lead to the idea of automatic weighting. The question is, if it is possible to define a function that sets the weights in a manner the deformation automatically becomes smooth.

5.5 Automatic Weighting



(a) Selection (b) ROI definition (c) Pulling

(d) Interpolation (e) Result

Figure 5.9: Pulling the bunny's ear by selecting only four vertices

How can the estimations from the last paragraph be used in order to enable an automatic weighting option for the user? Testing the application pointed out that it is sometimes annoying to adjust the sliders, if it is intended to deform the shape in a way which keeps all transitions inside the ROI smooth. Therefore, the automatic weighting function has been implemented. According to the consideration about the balances in the minimisation function (5.1), the vertex weights are set as

$$w_f = \frac{m(d+1)}{f}$$

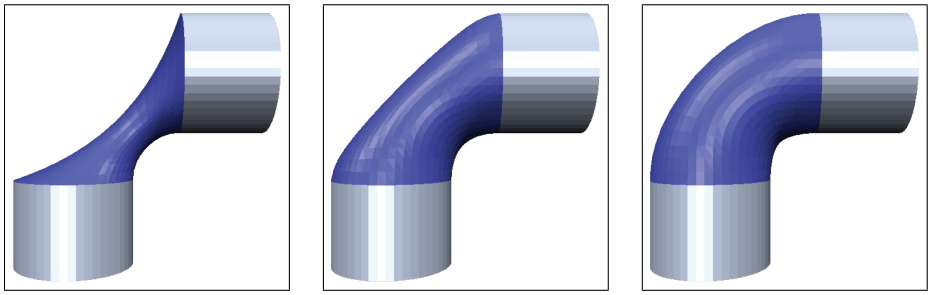
and

$$w_h = \frac{m(d+1)}{h},$$

where d stands for the mean valence of all vertices in the ROI. This weighting has two advantages. On the one hand, all three parts of the minimisation are balanced, and on the other one, a very fast and simple surface editing mechanism is available. Using automatic weighting, it is possible to change a shape very smoothly, without selecting a large amount of vertices. It is no more necessary to select the deformation region exactly. It is sufficient to select only a few handle vertices and extend the neighbourhood until the ROI defines the intended deformation area. Automatic weighting allows a smooth detail preserving deformation.

Figure 5.9 shows a practical example of a quick and simple editing task. Suppose the user wants to pull the left ear of the bunny. The user selects only four, more or less arbitrary, vertices of the ear (a). Then the neighbourhood ring is increased until the whole ear belongs to the neighbourhood. The region is marked with the yellow stars (b). The purple stars mark the ring of the fixed vertices at the head of the bunny. In the next step (c), the four vertices are translated to the intended position. After releasing the mouse button, the new vertex positions are interpolated (d). After deselecting the ROI, it is clearly recognizable that the ear is pulled, while its detail structure is kept (e). This example shows a very fast and efficient editing task. This task can easily be transferred to the main application of deforming units in automotive design. Instead of pulling an ear, some distant parts such as valves or other connections need to be deformed, because they block the mounting of a unit. In this case, the design engineer can easily select some vertices of this distant part, increase the neighbourhood until the whole part is inside the ROI and push it in. The automatic weighting scheme prevents complex selections for simple editing tasks.
The automatic weighting scheme has the additional benefit that there is almost no additional computation time necessary. The mean valence is calculated during the construction of the Laplacian matrix, where each valence has to be checked. Hence, there are only one multiplication and two divisions necessary to compute each of the two weights. Due to these reasons, the automatic weighting enhances the intuitiveness of the user interaction in the system. It rapidly reduces the necessary experience of the user getting satisfying results. In the next section, it is analysed upon which extent the weighting can control the boundary conditions inside the ROI. To do this, the system is again compared with other shape editing frameworks.

5.6 Boundaries through Weighting

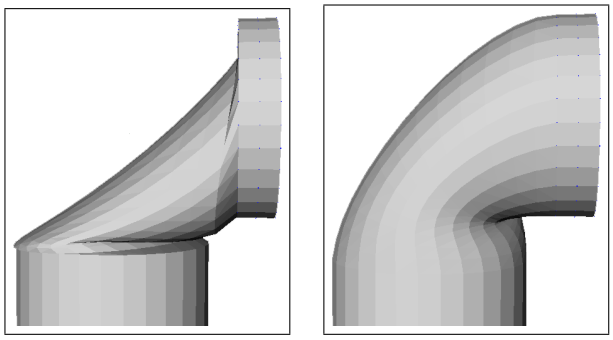


(a) $k = 1$,membrane surface (b) $k = 2$,thin plate surface (c) $k = 3$,minimal curvature variation

Figure 5.10: The order k defines the stiffness of the surface in the ROI, courtesy: [BK04b]

As mentioned in Section 2.4, Mario Botsch and Leif Kobbelt [BK04b] showed that the order of the Laplacian, respectively to the energy functional, controls the transition inside the ROI. They examined the first three degrees and this led to the results pictured in Figure 5.10. Based on the bending of a cylinder, the alternative transitions are illustrated. In the left image (a), the bending is realised as a direct connection. Only the connection points are considered but not the transitions at the boundaries. In image (b) the transition is much smoother but the bending still looks unnatural. Only with the order $k = 3$ in image (c) the bending has been realised in a natural way. In our implementation the Laplacian

of degree $k = 1$ is used, and hence it should lead to the membrane surface solution. Such a transition is not desired in the most cases because it creates sharp edges which do not represent natural behaviour if a shape is pressed slightly for a task like assembling. Increasing the weights for the fixed and selected vertices nearly equalises this drawback. In Figure 5.11 on the left side, the weights have been set equal to one in order to reduce their impact. That led to transitions which look similar to them from the membrane solution in Figure 5.10(a). On the right side, the weights have been increased to the maximum and hence, the bending resulted in an stiffness that lies between the thin-plate surface and the minimal curvature. These results showed that in this application, the weights can approximate the higher order



(a) $w_h = 1, w_f = 1$ (b) $w_h = 20, w_f = 20$

Figure 5.11: The vertex weights increase the stiffness

Laplacian. This has an enormous advantage because the use of higher order Laplacians implies a dense linear system and for this reason, the efficiency of such systems decreases. With the extended weighting scheme it is possible to dispense from the use of a higher order Laplacian if smooth transitions are intended. The weighting scheme does not increase the computation time for solving the linear system. Hence, the weighting enhances Laplacian mesh editing in such a manner that deformations become much more natural. This high degree of naturalness is necessary to simulate the deformation tasks of mechanics during the assembly. The addition of weights is therefore capable to simulate hand made deformations that do not destroy the shape, and therefore, they are similar to what a mechanic does when assembling or disassembling units.

Table 5.2 summarises the effects of each weight. The five differential coordinates weights are described according to remeshing and smoothing. As mentioned, both effects are not clearly positive or negative. They depend on the intentions of the user and the application. The two vertex weighting approaches, handle and fixed weights, control the boundaries to the neighbour vertices. They support the volume preservation by increasing the boundary region. The table is to give a review about the developed and utilised weights in this thesis.

Name	**Effect**
Uniform or umbrella	There is neither a remeshing nor a smoothing effect. Well suited if the mesh structure is to maintain.
Scale-dependent umbrella	Both remeshing and smoothing effects are latent. Compromises uniform and cotangent weighing.
Edge length	Leads to bad and distorting effects.
Tangent	Strong remeshing and smoothing effect. Geometrically not totally correct. Well suited for regular meshes.
Cotangent	Strong remeshing and smoothing effect. Geometrically totally correct. Well suited for smoothing and regularisation of the mesh.
Handle vertices weights	Controls the boundary between handle and neighbour vertices. Enables peak and dome deformation. Supports volume preservation at bending operations.
Fixed vertices weights	Controls the boundary between neighbour and fixed vertices. Smooths outer ring of ROI. Supports volume preservation.

Table 5.2: Summary of weights

6 Conclusion and Future Work

6.1 Conclusions

Purpose of this work was to develop a software for interactive systems to simulate deformations. The implemented software is based on Laplacian mesh editing. The user is able to determine a ROI in order to localise the editing process. A usual selection scheme is used which is easy to handle and allows a precise localisation. With Laplacian mesh editing it is possible to deform 3D objects while their surface details are preserved. Detail preserving is an important feature because it allows the simulation of realistic deformations. Deformations on real objects normally do not destroy the detail structure of the objects' surfaces. In the assembly simulation, the precise reproduction of mechanic tasks is fundamental. Due to this fact, the contact handling in the simulation is very elaborate. However, simulating a realistic deformation is much more complex than the contact of solid objects and thus, the deformation is kept highly interactive.
In the implementation, the deformation is mainly controlled through the use of weights. The weights are set by the user and they influence the deformation. The weights influence the boundaries and therefore realise smooth boundary conditions. This has been implemented successfully in the context of deformations during assembly simulations. Even for large deformations the influence of the weights is strong enough to control the deformation sufficiently. However, the weights lose their impact if large deformations are performed or if the mesh structures are too fine. This is a general drawback of the Laplacian mesh editing scheme because it considers only the one-ring neighbourhood of a vertex. The usage of weights can only limitedly compensate this drawback. But an extension of the one-ring to define a differential coordinate with stronger influence implies a dense linear system which can hardly be solved in real-time. The weights do support the volume preservation. The effects are not as positive as the ones, Zhou et al. have reached by introducing volumetric details, but the are adequate for fine mesh deformations. In addition, the use of weights does not increase the computation time in contrast to the additional volumetric details. Laplacian mesh editing is well suited for large scale and locally exact deformations at the same time. This is a great advantage in respect to other mesh editing methods. FFD and its extensions do not allow exact deformations. Multiresolution editing does not consider the mesh connection and have a disposition to create unwished artefacts. Radial basis functions are as suited as Laplacians for mesh editing but they are much more complicated and thus, there is a loss of flexibility. Users would need a long period of training in order to understand the system and its parameters. The utilisation of weights in Laplacian mesh editing is easy to learn and allows a broad spectrum of possibilities. The introduced weighting schemes are a simple method to control and bias the mesh deformations. They are easy to manage and their impacts are clear to the user. This makes the whole system very intuitive to use. Vertex weighting influences the boundaries between handle and neighbour vertices on the one hand, and the transition from neighbour vertices to them who remain fixed on the other one. The weights of the differential coordinate should be chosen depending on the tessellation. The umbrella weighting does not consider the mesh tessellation and thus, it does not construct precise details. The scale dependent umbrella operator weights the edges at the construction of differential coordinates and therefore increases the geometric precision of the coordinates.

Those weights who consider the angles of the edges, such as cotangent and tangent weighting, describe the detail of the surface correctly. This leads to better results for asymmetrically tessellated surfaces as shown in Section 5.4. In addition, the discussions throughout this thesis have also shown that the use of the cotangent weights for the differential coordinates is not mandatory. As mentioned in Section 5.4, these distortions must be avoided by the user selecting other weighting schemes.

The basic advantage of the programme developed in this thesis is that it is embedded in the VR framework "veo" which is used for the assembly simulations. The software developed allows the user to edit the shapes rapidly. The inclusion of a STL export makes the software a useful tool for rapid prototyping. Due to these facts, the manufacturing process can be accelerated distinctly. From now on, design engineers can easily do fine deformations during the VR simulation without switching the software. In order to simulate deformations which are possible because of the elasticity of the units, the user can press and deform objects within the VR software. The user does not need to transfer the object data to a complex CAD environment, execute the deformations there and transfer the edited data back to the VR environment. This induces a great saving of time. These deformations can be simulated easily by the use of the vertex translation along the mean normal direction from Section 4.4.1. The weighting schemes allows an intuitive controlling mechanism that on the one hand enhances the deformation options, and on the other hand allows throughout automatic weighting most common deformations without the need of adjusting parameters by hand.

The system has been implemented aiming at maximum flexibility on minimum complexity of the user interaction. The interactivity by user control makes the developed system usable for supporting the assembly tests. The system provides all intended specifications from Table 1.1. The deformations are easy to handle and hence, there is almost no additional training needed for the users. This aspect is supported by the fact that the software is embedded in the already used VR framework as a so called *surface deformation module*. Especially for the intended tasks, such as small deformations along the normal, the software works absolutely well.

The new system enhances the manufacturing or better said the prototyping process crucially. Avoiding the indirection over the CAD system for changing a shape, and in addition enabling an STL formatted export, makes the system predominantly suitable for testing purposes in automotive design.

However, the programme also has parts which need to be extended and improved in the future. The main drawback of the implemented system might be the high computation time for the Cholesky decomposition. The Cholesky decomposition has not yet been optimised to the sparseness of the Laplacian matrix. Due to the lack of a software library such as TAUCS [Tol03] which is used in several comparative solutions [LSCO$^+$04, LSLCO05, Sor05, SLCO$^+$04], the deformation is limited to approximately thousand vertices at interactive rates. If large areas are intended to be deformed, this limitation in speed is a great drawback. Fortunately, the main application of this software will be locally precise deformations, so a large amount of data to be computed will rarely occur.

6.2 Future work

In the following, several feasible methods are introduced that can refine the implemented system. Some of them base on the work of different authors, who have already been mentioned in the related work section in Chapter 2. At first, the potentialities of sparse linear solvers are examined and after that, some ideas of Mario Botsch and Leif Kobbelt are exposed. These ideas have been realised in their framework [BK05] and are adaptable to this implementation as well. They include the fast calculation of the surface normals on the fly and the translocation of several computations onto the GPU. After that, the including of remeshing algorithms is examined and finally, observations about the combination of the editing system with the collision detection are made.

6.2.1 Direct Linear Solvers

The main computation costs of the system are used by the interactive solving of the sparse linear system in the least squares sense. As mentioned, solving sparse linear systems can be accelerated by the use of a sparse matrix algorithms library. Olga Sorkine and Yaron Lipman recommend the use of the TAUCS library [Tol03] developed by Sivan Toledo. This library is state of the art in solving sparse linear system and is embedded in Mathematica 5 and Matlab 7. The main concept for sparse solvers is a decomposition of the matrix in an upper and lower triangular matrix, like the Cholesky decomposition explained in Section 3.7. The key for optimizing the methods is to reduce the fill-in. The fill-in is the amount of matrix entries that change during the process from zero to non-zero. In case of TAUCS, a fill-reducing reordering of the entries is done. For this reason, the computation time can be decreased. Yousef Saad [Saa96] describes the four phases of a typical sparse direct solution solver for positive definite matrices. It starts with the mentioned preordering step to reduce the fill-in. There are two popular variants available: minimal degree ordering and nested-dissection ordering. Factorisation is done symbolically and after that numerically. The symbolic factorisation can estimate the fill-in. If numerical pivoting is necessary, these steps are merged together. Finally, the system is solved by the already explained forward and back substitution. Since the main steps are already implemented in this application, it would be necessary to use a faster factorisation method that is optimised for sparse matrices. The next section deals with the computation of the surface normals.

6.2.2 Normal Calculation

Deforming the shape of an object obviously changes its surface normals. Recomputing the normals is essential for rendering, because otherwise the lighting will be computed wrongly. Currently, normals are calculated newly from the changed vertex data. In Laplacian mesh editing only the new vertex position after the deformation is known, not the transformation that led to this new position. In other frameworks, e.g. [BK05], the deformation is expressed as a function $F : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. This makes it possible to compute the new normals easily. If the new positions $\vec{p}'$ are calculated by

$$p_i' = F(p_i),$$

the transformation of the normals can be expressed as

$$n_i' = \det J_F (J_F(p_i)^{-1})^T n_i,$$

where $J_F(p_i)$ denotes the Jacobian of the deformation of the vertex p_i, which is a 3×3 matrix constructed by

$$J_F(p_i) = \left[\frac{\partial F(p_i)}{\partial x}, \frac{\partial F(p_i)}{\partial y}, \frac{\partial F(p_i)}{\partial z} \right],$$

where x, y and z are the values for each dimension of the coordinate. If the transformation of each vertex is known, it is possible to compute the transformation of its normal with the help of the Jacobian of the deformation. The new normal can be calculated by multiplying the original normal with the inverted and transposed Jacobian of the corresponding vertex. Since the normal has unit length, the calculation of the determinant value is obviously not necessary [Bar84]. The inversion of the Jacobian can be done analytically and so the computation is really efficient. The problem in the Laplacian mesh editing scheme is to find an exact or approximate Jacobian for each vertex transformation in order to calculate the normals rapidly.

Currently, the normals need to be recalculated from the character of their surrounding surface regions. In relation to the size of the deformed region, this can be very inefficient. In order to accelerate this task, the normal computation could be combined with the creation of the differential coordinates. This approach bases on the fact that the differential coordinate at each vertex has the same direction as the normal. So, recomputing of the normals could be connected with the recalculation of the detail vectors when a new ROI is selected. In the software presented in this thesis, normal recomputing has not been implemented yet because the main interest has been laid in shape deformation and its effects for assembly simulations and not on high quality rendering.

6.2.3 Porting to the GPU

As mentioned in Section 2.4 Mario Botsch and Leif Kobbelt developed a real-time shape editing system [BK05] which makes intense use of the GPU. The key concept of their work is the previous computation of basis functions which are sent to the GPU as textures. The new vertex positions are computed by a simple displacement function which is implemented as a shader. Due to this fact, only a local control frame which is comparable to the ROI data in this implementation has to be transferred in every step. Through the decrease of data transfer and the parallelisation of computations, the system is accelerated so that about 1.5 million vertices can be processed per frame. However, this approach is a volumetric space deformation method and for this reason, it has problems with exact handling of discrete vertices. Since we implemented a surface-based editing method, it would be possible to speed up the solving of the sparse linear system by relocating calculations on the GPU.

There is a paper about solving large sparse system by Jeff Bolz et al. [BFGS03]. They introduce an approach for solving sparse linear systems with the method of conjugate gradients. The sparse matrix is stored in textures. There are only two operations necessary during this algorithm, namely a sparse matrix-vector multiplication and the inner product of two vectors. For the sparse matrix-vector product the matrix rows are gathered in groups of rows with the same number of entries. These groups can be processed in parallel. For the inner product a sum reduction method is used that is based on

> "rendering a quadrilateral with half the dimension along either axis, summing four elements."
> From [BFGS03, p.4]

The GPU based algorithms developed by Bolz et al. can compute nearly twice as many matrix multiplications as on the CPU at the same time. Hence, in future one might use such a GPU-based solver instead of the Cholesky decomposition in order to speed up the system. First observations showed that there is a high potential for optimisations through GPU programming, but also that it is not easy to transfer the GPU based implementations for space-deformation techniques to surface based shape editing schemes.

6.2.4 Remeshing

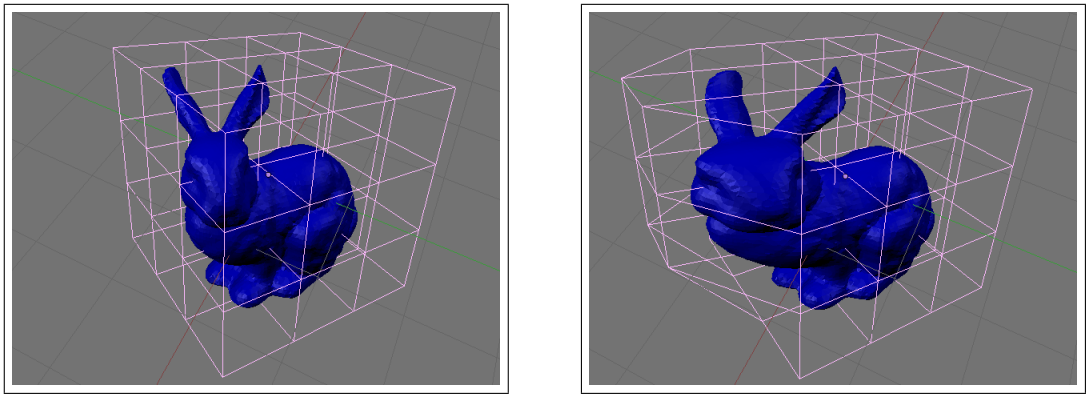
Another problem that might occur during mesh deformation is the rigidity of the mesh topology. If small features are pulled out of the surface for a large distance in relation to the triangles size, the triangles become strongly distorted. Or, in the case of irregular meshes, the triangles are already distorted and therefore, they prevent the deformation in the intended smoothness. This problem could be solved by remeshing. Remeshing simply means the change of the surface triangulation in order to increase the regularity of the mesh. In contrast to the remeshing effect the cotangent weight can cause, remeshing algorithms change the connections of the mesh. If some edges are too long or too short, the corresponding vertices are merged or split. There are several papers about remeshing algorithms [AMD02, SG03, SAG03, VRS03]. However, in Laplacian shape editing, it is not possible to remesh dynamically because this would change the Laplacian of the mesh. There is a dynamic mesh approach by Leif Kobbelt et al. [KBS00] for multiresolutional mesh editing and another one of him together with Mario Botsch [BK04a]. However, none of them can be adapted to Laplacian mesh editing. The key is to find a fast and easy way for remeshing on the one hand and for the corresponding change of the Laplacian on the other one. Currently, the computational effort of remeshing methods is too big to be computable at interactive rates.

6.2.5 Combined Collision Detection

As mentioned in Section 1.1 the software developed is used for assembly tests. There is one future approach for the industrial usage of the system that needs to be implemented. One could combine collision detection and contact simulation with the mesh editing algorithms in order to automate the deformation of the components after the assembly tests. The system would then examine the contacts during the simulation of the assembly and calculate the deformations that are necessary to enable a trouble free mounting. One way of doing this is adding the tool metaphor to the modelling framework and regard one of the colliding objects as the tool and the other one as the modelling object. In this case the contact of the objects directly causes a deformation. Another idea could be to compare the paths of the objects and analyse the penetrations. All these approaches require an accurate description of the structural elements' stiffness. It is necessary to determine which object is flexible and how intense is its elasticity. The complexity of all the approaches grows immediately if all possible cases of deformation and contact are considered. The objects are of arbitrary shape and topology. A consideration of all components as solid primitives would lead to a high degree of inaccuracy and would make the results unemployable.

A possible extension of the modelling system would be a deformation restriction for the structural elements. Based on material characteristics, a stiffness factor could be added as an attribute for the deformation object. This factor could control the shape manipulation and avoid unrealistic deformations. During long term utilisation of the system, a kind of database of different materials could be build up in order to provide an additional deformation control for unskilled users.
There is a high potential to extend the system for automation and support of the manufacturing process. It helps the user, but does not deprive the user of the full control over the components. The possible extensions and improvements presented in the last sections have not been implemented because they would constitute enough material for a second diploma thesis.

A Colour plates



(a) Bunny with lattice

(b) Deformed lattice and bunny

Figure 2.1: FFD on the bunny model with a regular lattice

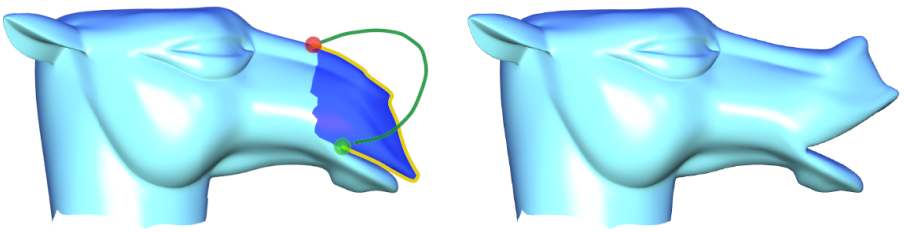


Figure 2.3: Sketch-based editing, courtesy: [NSACO05]

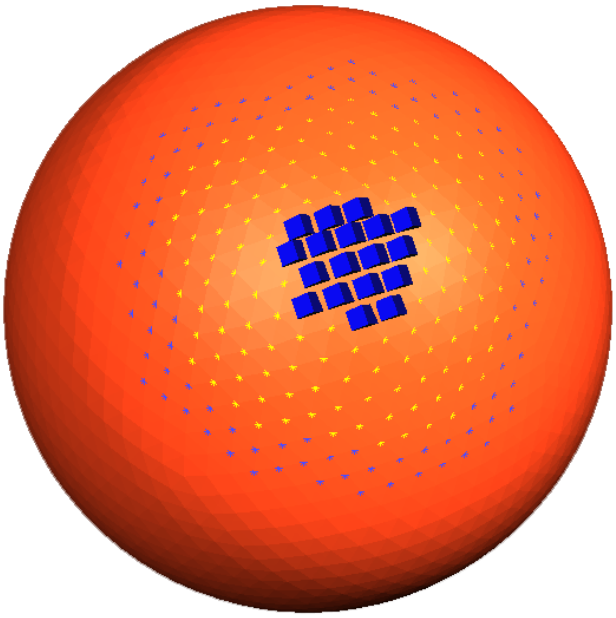
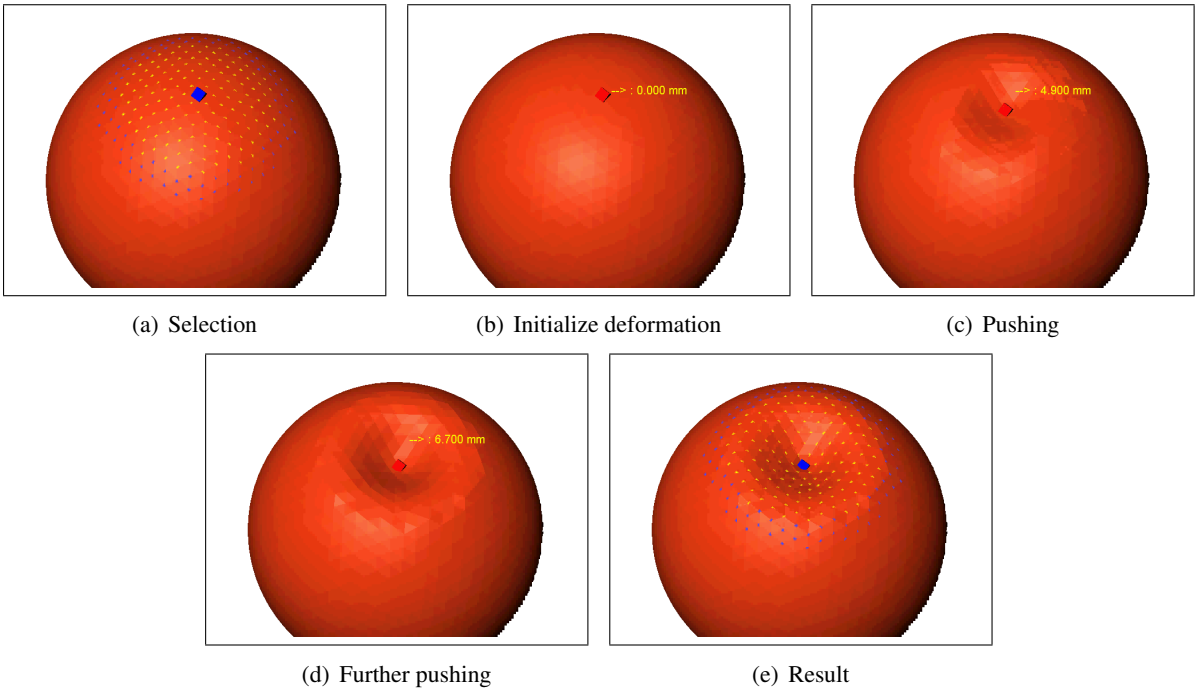


Figure 4.1:A classic ROI selection on a sphere



(a) Selection (b) Initialize deformation (c) Pushing

(d) Further pushing (e) Result

Figure 4.2:Pressing an indent into a sphere

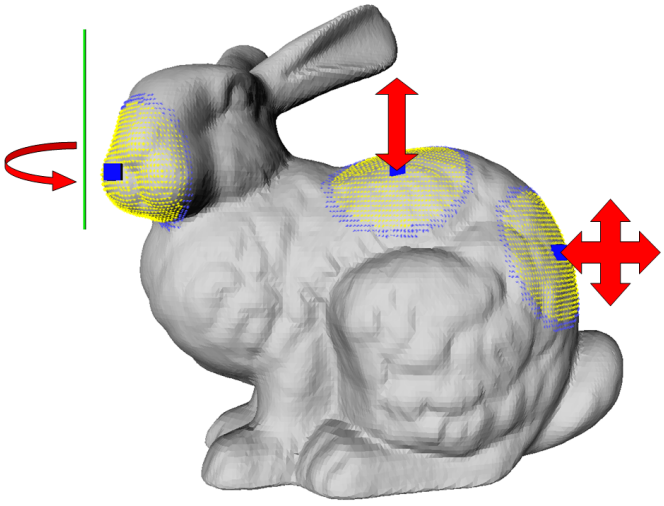
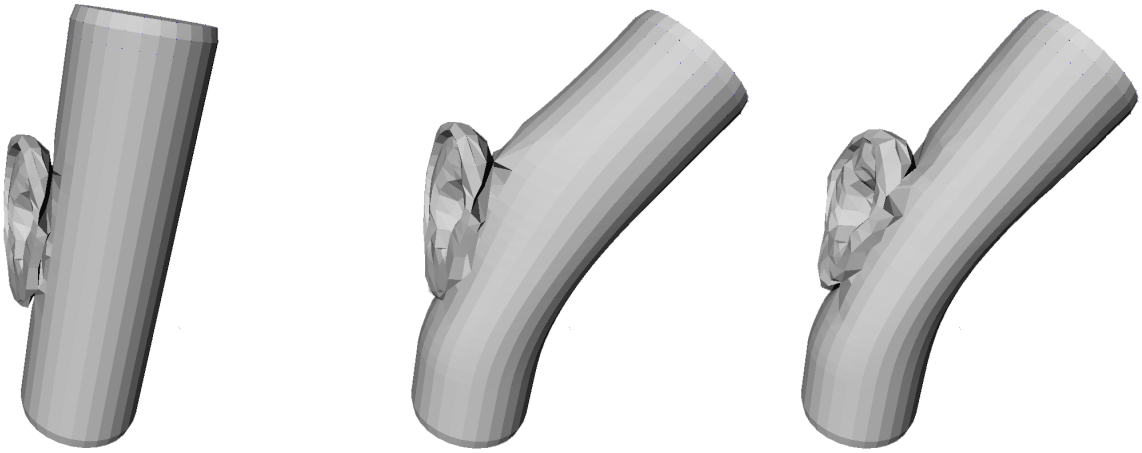


Figure 4.3:The vertex transformations

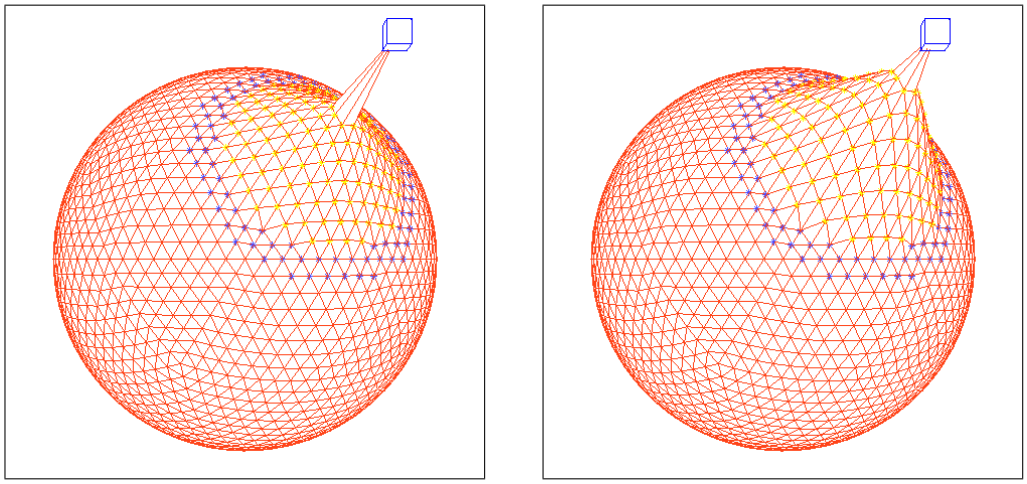


(a) Original model

(b) Bending without rotation of detail vectors

(c) Bending with rotation of detail vectors

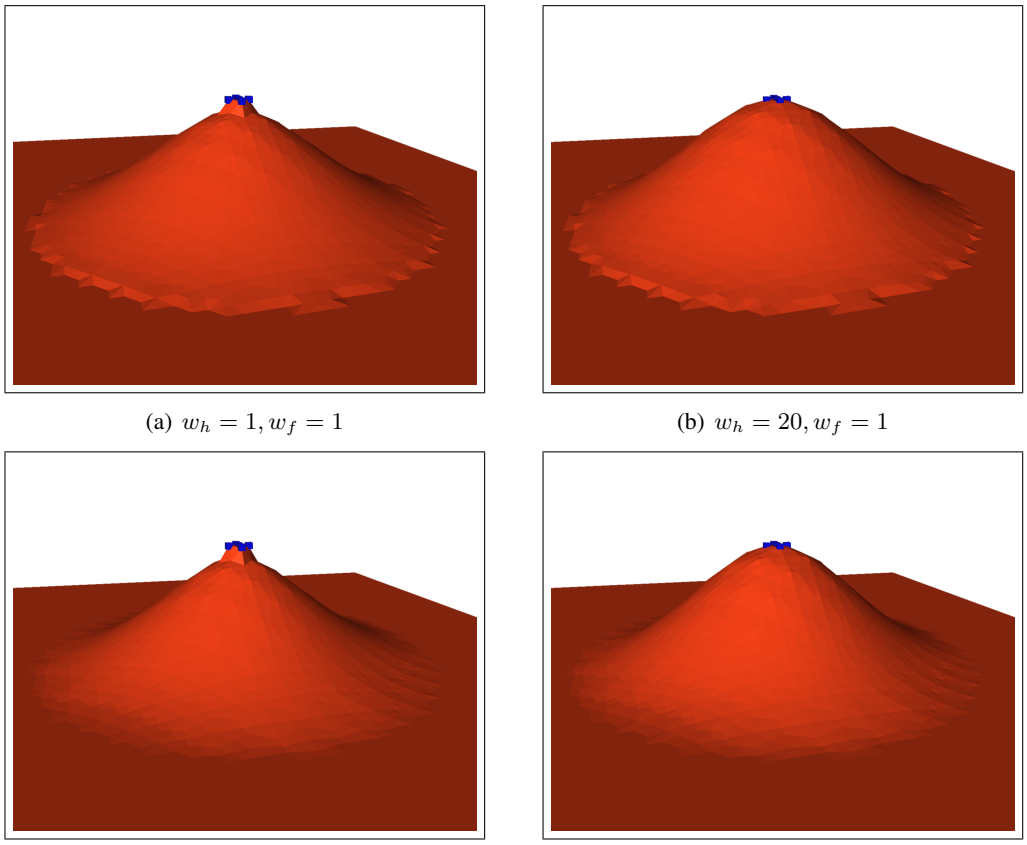
Figure 4.6:Differences in bending strong details



(a) $w_h = 1$ (b) $w_h = 2$

(c) $w_h = 5$ (d) $w_h = 20$

Figure 5.2:Exalting one vertex from a sphere with different handle vertices weights on constant fixed vertices weight $w_f = 1$



(a) $w_h = 1, w_f = 1$

(b) $w_h = 20, w_f = 1$

(c) $w_h = 1, w_f = 20$

(d) $w_h = 20, w_f = 20$

Figure 5.3:Exalting five vertices from a plane with different vertex weights

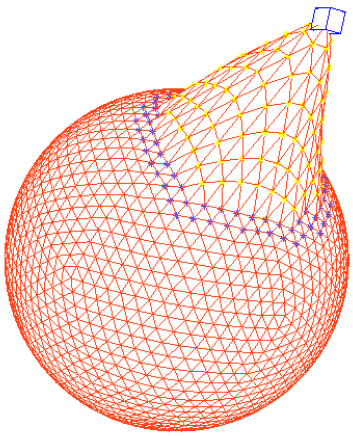
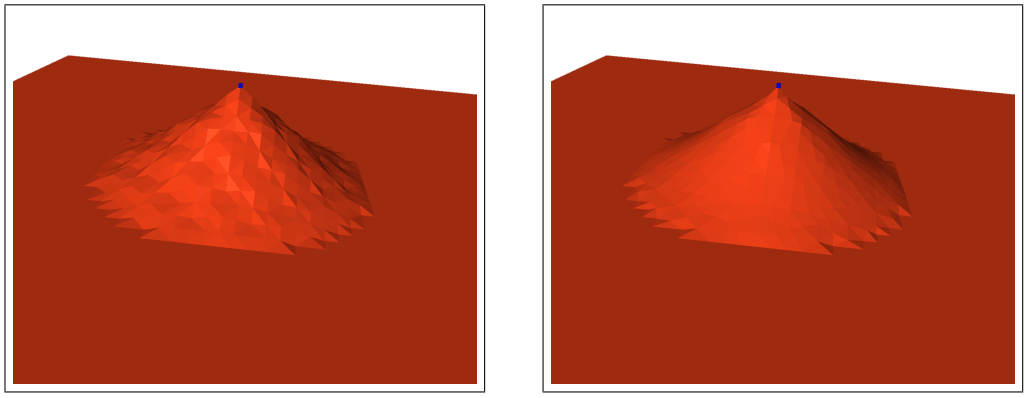


Figure 5.4:Exaggerated weighting



(a) Uniform (b) Cotangent

Figure 5.5:Exalting one vertex from a plane with different weighting

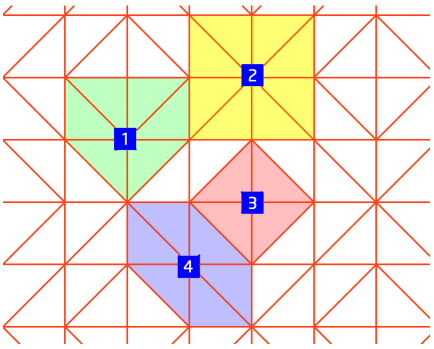


Figure 5.6:Mesh structure of plate

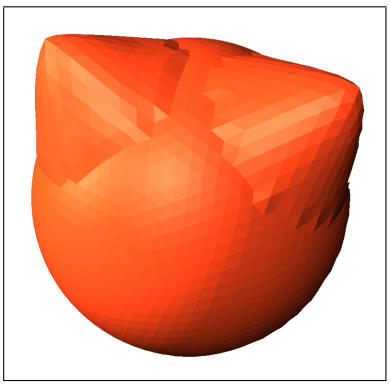
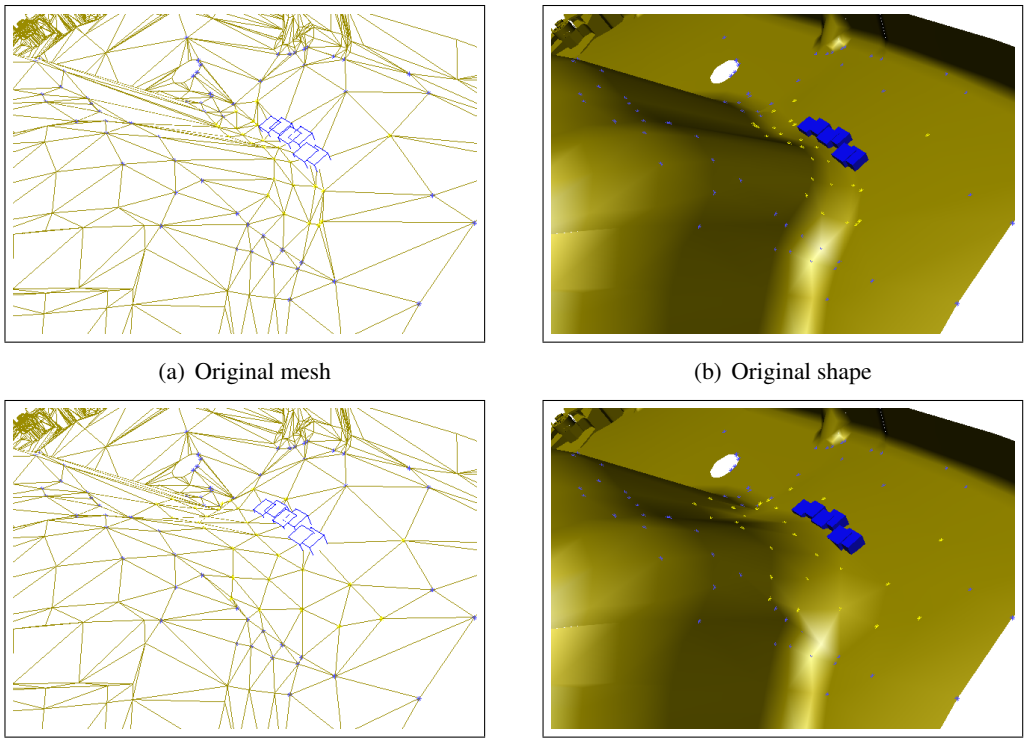


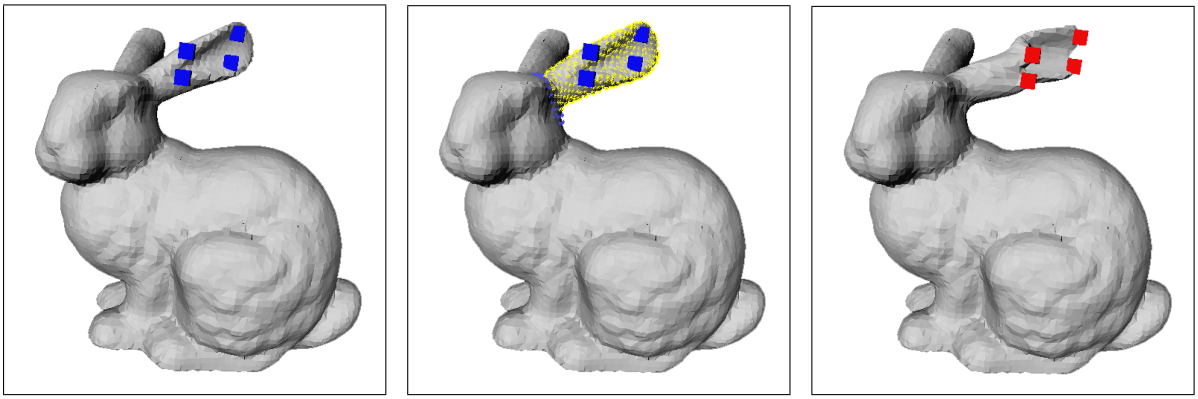
Figure 5.7:Two exalted vertices from a sphere



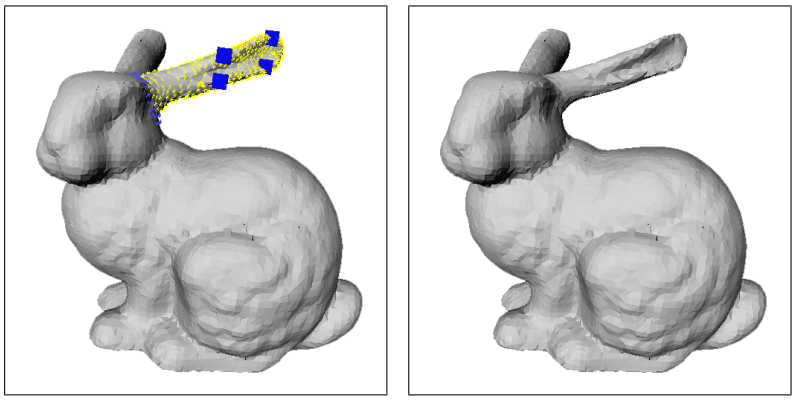
(a) Original mesh (b) Original shape

(c) Deformed mesh (d) Deformed shape

Figure 5.8:Remeshing effect after fine deformation with cotangent weights

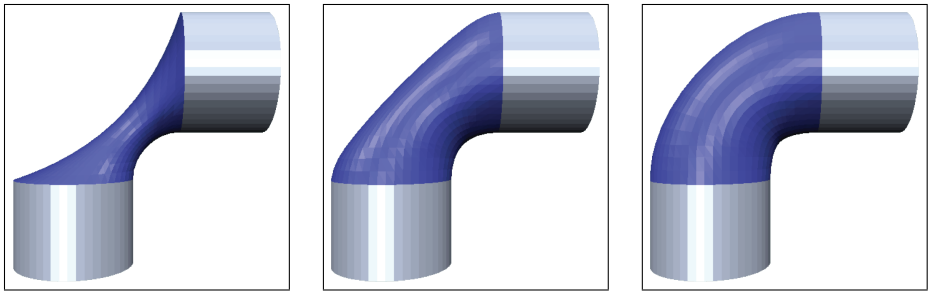


(a) Selection (b) ROI definition (c) Pulling



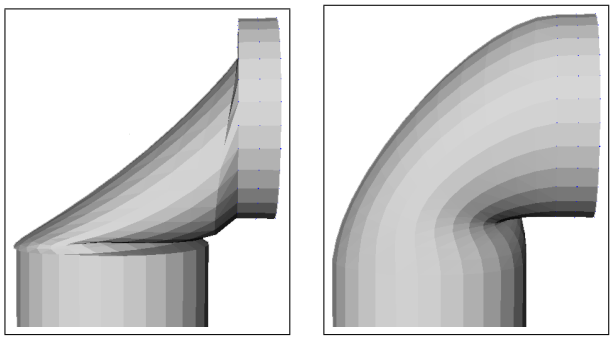
(d) Interpolation (e) Result

Figure 5.9:Pulling the bunny's ear by selecting only four vertices



(a) $k = 1$,membrane surface (b) $k = 2$,thin plate surface (c) $k = 3$,minimal curvature variation

Figure 5.10:The order k defines the stiffness of the surface in the ROI, courtesy: [BK04b]



(a) $w_h = 1, w_f = 1$ (b) $w_h = 20, w_f = 20$

Figure 5.11:The vertex weights increase the stiffness

References

[ACOL00] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Ale03] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114, 2003.

[Ale05] Marc Alexa. Mesh Editing based on Discrete Laplace and Poisson Models. In *Eurographics Tutorial 5 - Interactive shape modelling*, 2005.

[AMD02] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Trans. Graph.*, 21(3):347–354, 2002.

[Ang05] Alexis Angelidis. Sweepers and Swirling sweepers. In *Eurographics Tutorial 5 - Interactive shape modeling*, 2005.

[Bar84] Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press.

[BCG05] M. Ben-Chen and C. Gotsman. On the Optimality of Spectral Compression of Mesh Data. *ACM Transactions on Graphics*, 24(1):60–80, January 2005.

[BFGS03] Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, 2003.

[BK01] M. Botsch and L. Kobbelt. A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes, 2001.

[BK03a] G. H. Bendels and R. Klein. Mesh Forging: Editing of 3D-Meshes Using Implicitly Defined Occluders. In *Symposium on Geometry Processing 2003*, June 2003.

[BK03b] Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. In *Computer Graphics Forum*, volume 22, 2003.

[BK04a] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 185–192, New York, NY, USA, 2004. ACM Press.

[BK04b] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, 2004.

[BK05] Mario Botsch and Leif Kobbelt. Real-Time Shape Editing using Radial Basis Functions. In *Computer Graphics Forum*, volume 24 of *Eurographics 2005 proceedings*, pages pp. 611 – 621, 2005.

[BKK04] Gerhard H. Bendels, Ferenc Kahlesz, and Reinhard Klein. Towards the next generation of 3D content creation. In *Proceedings of International Working Conference on Advanced Visual Interfaces (AVI 2004)*, pages 283–289. ACM Press, May 2004.

[BKS03] G. H. Bendels, R. Klein, and A. Schilling. Image and 3D-Object Editing with Precisely Specified Editing Regions. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Vision, Modeling and Visualisation 2003*, pages 451–460. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[BR94] Paul Borrel and Ari Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Trans. Graph.*, 13(2):137–155, 1994.

[BSM00] Ilja N. Bronstein, Konstantin A. Semendjajew, and Gerhard Musiol. *Taschenbuch der Mathematik. Mit CD-ROM.* Deutsch (Harri), 2000.

[Buc99] Matthias Buck. *Simulation interaktiv bewegter Objekte mit Hinderniskontakten*. PhD thesis, Universität des Saarlandes, 1999.

[Coq90] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1990. ACM Press.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182, New York, NY, USA, 1995. ACM Press.

[FKR05] M. S. Floater, G. Kos, and M. Reimers. Mean value coordinates in 3D. *Comp. Aided Geom. Design*, 22:623–631, 2005.

[Flo03] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[Fuj95] K. Fujiwara. Eigenvalues of Laplacians on a closed Riemannian manifold and its nets. In *Proc. AMS.*, volume 123, pages 2585 – 2594, 1995.

[GM05] James Gain and Patrick Marais. Warp Sculpting. In *IEEE Transactions on Visualization and Computer Graphics*, volume 11, pages pp. 217–227. IEEE Computer Society, March/April 2005.

[GSS99] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 325–334, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[HHK92] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformations. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 177–184, New York, NY, USA, 1992. ACM Press.

[HML99] Gentaro Hirota, Renee Maheshwari, and Ming C. Lin. Fast volume-preserving free form deformation using multi-level optimization. In *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 234–245, New York, NY, USA, 1999. ACM Press.

[HQ02] Jing Hua and Hong Qin. Haptics-based volumetric modeling using dynamic spline-based implicit functions. In *VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, pages 55–64, Piscataway, NJ, USA, 2002. IEEE Press.

[HQ03] Jing Hua and Hong Qin. Free-form deformations via sketching and manipulating scalar fields. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 328–333, New York, NY, USA, 2003. ACM Press.

[HTK+04] B. Heidelberger, M. Teschner, R. Keiser, M. Müller, and M. Gross. Consistent Penetration Depth Estimation for Deformable Collision Response. In *Proc. Vision, Modeling, Visualization VMV'04*, pages pp. 339–346, Stanford, USA, Nov. 16-18 2004.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko 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, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[JP04] Doug L. James and Dinesh K. Pai. BD-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3):393–398, 2004.

[JSW05] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, 2005.

[KBS00] Leif P. Kobbelt, Thilo Bareuther, and Hans-Peter Seidel. Multiresolution Shape Deformations for Meshes with Dynamic Vertex Connectivity. *Computer Graphics Forum*, 19(3):C249–C260, 2000.

[KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM Press.

[KG00] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[KZ03] J. Klein and G. Zachmann. ADB-Trees: Controlling the Error of Time-Critical Collision Detection. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Vision, Modeling and Visualisation 2003*, pages 37–46. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[Lee99] S. Lee. Interactive multiresolution editing of arbitrary meshes. *Computer Graphics Forum (Eurographics 99)*, Vol. 18(3):pp. 73–82, 1999.

[Loc04] Werner Loch. Kollisionserkennung virtueller Objekte, 2004.

[LSCO+04] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential Coordinates for Interactive Mesh Editing. In *SMI*, pages 181–190, 2004.

[LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear Rotation-invariant Coordinates for Meshes. In *Proceedings of ACM SIGGRAPH 2005*, page accepted for publication. ACM Press, 2005.

[MDSB02] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Al Barr. Discrete Differential Geometry Operators for Triangulated 2-Manifolds. In *VisMath '02 Proceedings*, 2002.

[MJ96] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1996. ACM Press.

[MJBF02] Tim Milliron, Robert J. Jensen, Ronen Barzel, and Adam Finkelstein. A framework for geometric warps and deformations. *ACM Trans. Graph.*, 21(1):20–51, 2002.

[Moh97] B. Mohar. Some applications of Laplace eigenvalues of graphs, 1997.

[NSACO05] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, 2005.

[Par02] Rick Parent. *Computer Animation - Algorithms and Techniques*. Morgan Kaufmann Publishers, 2002.

[Pha] *PHANTOM and SensAble Technologies, Inc. are registerd trademarks.*

[PKKG03] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. In *Experimental Mathematics*, volume 2, pages 15–36, 1993.

[Pre94] Jenny Preece. *Human Computer Interaction*. Addison-Wesley, 1994.

[RSB95] Ari Rappoport, Alla Sheffer, and Michel Bercovier. Volume-preserving free-form solids. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 361–372, New York, NY, USA, 1995. ACM Press.

[Saa96] Yousef Saad. *Iterative methods for sparse linear systems*. PWS, 1996.

[SAG03] Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In *Proceedings of 12th International Meshing Roundtable*, pages 215–224, Santa Fe, New Mexico, USA, September 2003.

[Sch04] Sylvie Schöniger. Deformation for Real-Time Interaction in Virtual Environments, 2004.

[SCOT03] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 42–51. Eurographics Association, 2003.

[SF98] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414, New York, NY, USA, 1998. ACM Press.

[SG03] Vitaly Surazhsky and Craig Gotsman. Explicit Surface Remeshing. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 17–28, Aachen, Germany, June 2003.

[SK04] A. Sheffer and V. Kraevoy. Pyramid Coordinates for Morphing and Deformation. In *Proc. Second International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)*, pages 68–75, 2004. invited.

[SLCO+04] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian Surface Editing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 179–188. Eurographics Association, 2004.

[Sor05] Olga Sorkine. Laplacian Mesh Processing. In *STAR - State of the Art Report EUROGRAPHICS*. EG, 2005.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.

[Tau95] Gabriel Taubin. A Signal Processing Approach to Fair Surface Design. *Computer Graphics*, 29(Annual Conference Series):351–358, 1995.

[Tau00] Gabriel Taubin. Geometric signal processing on polygonal meshes. In *Eurographics - STAR - State of the Art report*, 2000.

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision Detection for Deformable Objects. *Computer Graphics Forum*, Volume 24:pp. 61–81, 2005.

[Tol03] Sivan Toledo. *TAUCS: A Library of Sparse Linear Solvers, version 2.2., Available online at http://www.tau.ac.il/ stoledo/taucs/.* Tel-Aviv University, September 2003.

[VRS03] J. Vorsatz, Ch. Rössl, and H.-P. Seidel. Dynamic remeshing and applications. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 167–175, New York, NY, USA, 2003. ACM Press.

[WBD02] Westkämper, Bullinger, and Joachim Deisinger. *Entwicklung eines hybriden Modellier-systems zur immersiven konzeptionellen Formgestaltung*. Harfensteller, Ursula, u. Erika Petersdorf, 2002.

[Wer93] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3d Graphics with Open Inventor, Release 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[Wik06] Wikipedia. Weighting — Wikipedia, The Free Encyclopedia, 2006. [Online; accessed 7-May-2006].

[WW92] William Welch and Andrew Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992.

[WW94] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 247–256, New York, NY, USA, 1994. ACM Press.

[YZX+04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[ZHS+05] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph.*, 24(3):496–503, 2005.

[Zor05] Denis Zorin. Modeling with Multiresolution Subdivision Surfaces. In *Eurographics 2005 Tutorial - Interactive shape modelling*, pages pp. 11–36, 2005.

[ZRKS05] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic Guidance for Surface Deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

[ZSD+00] Denis Zorin, Peter Schröder, Tony DeRose, Leif Kobbelt, A. Levin, and Wim Sweldens. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000.

[ZSS97] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 259–268, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.