UPPSALA
UNIVERSITET

UPTEC F15 056
Examensarbete 30 hp
September 2015

Matching of geometrically and topologically changing meshes

Kristoffer Jonsson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Matching of geometrically and topologically changing meshes

Kristoffer Jonsson

The aim for this thesis is to develop a foundation for a compression system for animated mesh sequences, specifically under dynamic change of mesh geometry and topology. Compression of mesh sequences is of special interest in the game industry and this particular thesis is a part of an ongoing series of projects at EA DICE. One of the primary challenges when creating a mesh compression system is creating a matching bijective subset of the mesh surfaces between two subsequent frames in the animation to guide remeshing of the sequence. This thesis describes a method for producing a bijective set of matching mesh patches between two meshes along with an error metric that captures the quality of the matching in terms of shape similarity and distortion.

Theory of mathematical topology and tensor algebra used in methods for high performance scientific digital 3D-image recognition are here adopted to extract similar local features between meshes. Techniques for creating parametrizations of mesh patches are combined with techniques for matching point clouds and deforming mesh geometry under energy minimization in order to produce a matching set of patches.

The presented algorithm successfully creates bijective sets of matched patches for subsequent meshes in a sequence as well as measures the error for the matchings. Results show an average matching set size of approximately 25% of the mesh areas over a sequence of meshes. This suggests that the data size of such a sequence could potentially be reduced by 25%.

Handledare: Jan Schmid
Ämnesgranskare: Stefan Engblom
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F15 056

Populärvetenskaplig sammanfattning

Syftet med detta examensarbete är att arbete fram ett verktyg som ska kunna användas för att komprimera animeringssekvenser av ostrukturerade diskretiseringsnät i tre dimensioner, så kallade 3D-meshes. Sådana sekvenser används ofta i datorspel vid grafikrendering av 3D-objekt och det är av stort intresse att minska datamängden de upptar för att öka spelens prestanda. Förtaget EA DICE driver en serie projekt vars syfte är att utveckla ett sådant verktyg och detta examensarbete är en del i den serien.

Idén för själva verktyget är att konstruera en algoritm som kan lokalisera bitar i sekvensens 3D-objekt som liknar varandra för att på så vis kunna bevara de bitarna genom hela sekvensen. Om stor del av sekvensen inte förändras under animeringen kan mycket bevaras och graden av komprimering kan potentiellt sett bli hög. Av intresse är också att kvantitativt kunna mäta hur pass lika bitarna är varandra i termer av form och storlek för att kunna avgöra hur pass destruktiv komprimeringen kommer bli.

Teori från metoder som används inom tredimensionell bildanalys av cellulära strukturer, deformering av meshes under energiminimering samt iterativa metoder för matchning av punktmoln används i detta examensarbete för att konstruera en bra algoritm. Det kvantitativa måttet av bitarnas likhet utvecklas med hjälp av ytintegraler på diskret geometri.

Mätresultat över algoritmens förmåga att lokalisera likheter mellan bildrutorna visar på att i snitt 25% av 3D-objektens yta kan bevaras genom mätsekvensen, och fortfarande hålla graden av likhet mellan bitarna hög. Mängden bevarad yta och graden av komprimering är någorlunda proportionella mot varandra vilket innebär en potentiell komprimering med ca 25%.

Acknowledgments

Thanks to EA DICE for providing the idea for this thesis as well as a good work space and especially thanks to Jan Schmid for providing good guidance during the project progression.

Contents

Chapter 1

Introduction

Representing an animation of a 3D-object undergoing a visual change of some sort by a series of polygon structured meshes is today a well used technique in computer graphics. An example of such an animation could be the movement of a fluid, e.g. water flowing down a river or being thrown out of a glass. In those kinds of 3D-animations it is the fluid surface that is normally represented by the polygon mesh. Compression of a series of polygon meshes could conceptually be achieved by analyzing what part of the animation is changing and what part is not. The part that does not change could be preserved in between frames and thereby reducing the total amount of data that need to be stored for that particular animation. Although compression of various data structures is a thoroughly studied field in computer science not all data structures are easily compressed. A stream of data consisting of sequences of animated meshes that constantly change in terms of geometrical and topological structure during the animation are generally cumbersome to compress.

A big part of the analysis of meshes lies in the mathematical realm of topology. Mathematical topology has been extensively studied throughout the years but a problem like this still presents a number of challenges in terms of problem description and solution method. The problem of creating a tool that can be used for compression of a stream of geometrically and topologically changing meshes has no intuitive formulation or solution.

The idea for this thesis is to construct a method for bijectively matching parts of meshes that look similar while not necessarily having anything in common in terms of polygon structure. A bijective match means that every element in the first set has exactly one matching element in the second set and every element in the second set has exactly on matching element in the first set. Figure 1.1 shows a schematic view of a bijective mapping of elements in two matched sets.

The matching should produce a way of mapping an arbitrary point on a source mesh (i.e. any point on the source mesh, not just vertices, edges or faces) to exactly one point on the target mesh, while keeping distortion of the matching to a minimum. It should also treat patches of different geometry and topology in a way so that the bijective criterion still is met. The idea is to later use this mapping of points to reconstruct the sequence of meshes so that it has similar polygon structure throughout the animation, but still preserves its own shape a look in every frame.

By applying this matching and remeshing to a sequence of meshes the whole animation can be compressed by creating a file format that utilizes information from previous frames in order to generate the next frame. An analogue can be made to the movie file format MPEG-4, which uses information such as rotation, translation and preservation of image areas to encode the movie file. In the case of MPEG-4 some properties are much more static than in the case of arbitrarily

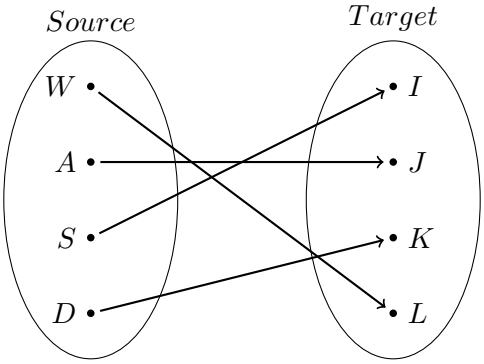


Figure 1.1: *A schematic view of a bijective mapping between matching source and target sets.*

shaped polygon structures. Fundamental properties that differ are for example that in MPEG-4 it is a well defined number of pixels that each frame contains compared to an arbitrary number of vertices in each mesh corresponding to a frame. Other differences are that in MPEG-4 the connectivity in each frame stays the same, i.e. the pixels are equally connected to each other in each frame as opposed to the case of the connectivity of the vertices in a sequence of meshes which can completely change. Despite theses differences the analogue is not all that bad in terms of describing properties that can be used for compression.

Matching a mesh can be done in numerous ways depending on under what criteria the matching should be done. Many mesh matching algorithms does unfortunately rely on either static topology, static connectivity or both of them. In [DP13] it is shown how to utilize precise differentiation in connectivity representation, similar to algorithms used when analyzing change lists in coding, in order to extract similar and changed parts of a mesh. However, this method works very poorly when meshes are completely restructured each frame i.e. not having any similar connectivity structure at all.

In terms of problems of matching topology some work has been done to achieve matching of features in between meshes [EGKT09, EEG+08]. While parts of these algorithms have potential to work in this problem setting of changing topology, the fact that they entirely rely on somewhat constant connectivity is hard to work around.

Other papers describe how to measure the degree of similarity between two closed meshes based on curvature mapping of the meshes. Being able to measure how similar two meshes are is of great value for this problem as well. Unfortunately the method described in [SHI96] relies on uniform distribution of vertices, which is something that is definitely not guaranteed in the current problem setting. Another drawback of the approach described in [SHI96] is that it tries to measure the degree of matching of the entire meshes at the same time. This presents a problem if the relevant part is to extract local features of meshes that are similar as opposed to determining if the entire meshes can be replaced with each other or not.

Other approaches of mapping closed three dimensional meshes onto two dimensional cylinders and then comparing the mapping for different meshes in order to measure the degree of matching has been tried in [ONIT04]. However this relies on static topology and will completely fail to describe any furled features of a mesh resembling a fluid shape.

At first glance it might seem intuitive to compress an animation by analyzing the structure of the meshes and see what vertices, edges and faces are preserved in between frames. While this method works for analyzing changes in a mesh that preserve the connectivity structure as well

as the topology it is not well suited for analysis of meshes that completely change connectivity structure or topology. Properties such as number of vertices, edges, faces or manifolds are in general not preserved in between frames in fluid animations since the mesh for each frame is generated as a result of a fluid simulation at a particular time step. Two different meshes can for example describe two surfaces that to the human eye would look almost identical and still have completely different polygon structure. The non preservative nature of the polygon structure is true for even tiny changes in between frames, which means compression by analysis of polygon structure produces quite poor results for these kinds of animations. As described above, previous work often assume some property being constant or equal in between meshes. Since these assumption are often completely invalid when dealing with meshes constructed for fluid animations much of previous work revolving around the topic of mesh matching is more or less inapplicable for this particular problem.

This thesis will therefore focus on developing an algorithm with a completely new approach to the problem. Focus will be on the structure of the algorithm itself and the necessary steps needed in order to achieve a valid result rather than choices affecting computational performance. This thesis will not prove nor claim that the algorithm presented for creating a bijective set of matching patches in between two frames are the optimal solution to this problem, only that it works to create some degree of matching. Furthermore the aim is to develop an error metric which enables a way of quantitatively measuring the quality of a matching between two meshes. The error metric needed for this should focus on capturing the level of distortion of area and shape for a matching, i.e. how well two meshes locally resemble each other.

1.1 Thesis structure

This thesis has a structure that successively describes how the algorithm works and the mathematics that lay the foundation for the various methods used to overcome certain obstacles.

The first chapter starts with a brief introduction, including previous work and problem description. Chapter 2 presents necessary criteria for the input data as well as some mathematical background concerning manifolds and mathematical topology. In Chapter 3 the steps for defining and extracting corresponding local features for the meshes are described. The process of making an initial matching of actual parts of the meshes along with the mathematics behind matching error evaluation is shown in Chapter 4. The full mesh matching is finally wrapped up in Chapter 5 showing how the global bijective matching criteria is fulfilled. Algorithm 1 describes all steps taken in Chapter 2 to 5 from a bird's eye view showing the general structure for how to produce a matching between two meshes.

Algorithm 1 *Outline of the algorithm for matching two meshes.*

1: Input: raw data of two meshes.
2: **for** $i \in [1, 2]$ **do**
3: Preprocess data for mesh i to make sure input is okay.
4: Extract local features of mesh i.
5: **end for**
6: Find, extract and match mesh patches based on similarity in terms of local features.
7: Successively cover as great mesh area as possible by matching several mesh patches.
8: Return: matching data for the two meshes.

Succeeding the mathematically heavy chapters is Chapter 6 which shows interesting measur-

able properties that capture the quality of the mesh matching as well as the potential compression gained by using this algorithm.

The final chapters (Chapter 7 and 8) consist of a discussion concerning how well the algorithm performed in solving said problem, suggestions of future work and final conclusions.

Chapter 2

Preprocess data

This section briefly describes how the raw input data, consisting of positions and connectivity for all vertices in a mesh, is treated and what criteria it must fulfill in order to be valid input data.

2.1 Closed triangular meshes

The input data must first of all for each frame pair consist of a set of closed meshes, $M_1 \subset \mathbb{R}^3$ and $M_2 \subset \mathbb{R}^3$. This is a requirement since the meshes will be voxelized, which is not intuitively defined for open meshes. Worth noting is that if additional software for handling voxelization of open meshes is provided or developed the rest of the algorithm (apart from the voxelization steps) does not need to be changed.

The set of closed meshes must also only consist of triangles, any other polygon type must be converted to a set of triangles. This is a requirement since triangles have extremely useful properties that polygons of higher order does not posses, e.g. all vertices in a triangle are guaranteed to lie in the same plane defined by the face they span.

2.2 Manifolds

The theory of manifolds is a complex subject of mathematics and is not easily understood over night. Analysis of manifold theory can be found in [Whi35, CdGDS13, GP74, MW97, BT82].

There is a distinction between the theory of manifolds in discrete geometry and continuous geometry. The introduction here will not necessarily make a distinction between the two and will not dive deep into heavy mathematics but rather give a short conceptual view of manifolds. Only an extremely brief introduction to the elements that are necessary in order to get a better grasp of the various concepts used are therefore given here. This statement describes necessary, but not sufficient, criteria that the meshes must fulfill in order to be valid:

Manifold criteria. *The set of closed triangular meshes must be two dimensional, locally orientable and differentiable, pure manifolds embedded in* $\mathbb{R}^3$*. If fulfilled the mesh is said to be a valid mesh in terms of the manifold criteria.*

This statement can be broken down into sub criteria which are more easily explained one by one for a manifold M.

First of all a pure manifold is a manifold that has a fixed dimension around every point which is equal to the global dimension of the manifold. This can be thought of as if the manifold is of

global dimension n then in a small local neighborhood around every point the dimension is also n. This might seem as an intuitive property but it is not always the case that the dimension is the same for all points. See [Ria08] for further reading concerning this matter.

Secondly the manifold must be orientable. The mathematical definition for manifold orientability involves a way of representing a manifold as a set of *charts*. A set of charts that together describe a manifold is called an *atlas*. An example where the word atlas makes perfectly good sense is in the case of the planet earth. Let's say the surface of the earth is a two dimensional manifold, $M \in \mathbb{R}^2$, embedded in $\mathbb{R}^3$. When looking at maps of different parts of the earth it is essentially an atlas of the manifold that is observed, i.e. an atlas of a manifold is a way of cutting the manifold into pieces so that it can be realized in the same dimension as it self describes. The charts of an atlas describing a manifold resembling a two dimensional surface embedded in $\mathbb{R}^3$ are subsets of $\mathbb{R}^2$ but they are also embedded in $\mathbb{R}^2$ instead of being embedded in $\mathbb{R}^3$ like the manifold is. Such an atlas can be described as the set given by

$$\{(U_i, \varphi_i : U_i \to V_i \subset \mathbb{R}^2)\}, \tag{2.1}$$

where each chart, (U_i, φ_i), is composed out of a *patch*, U_i, with $U_i \subset M$ and $\bigcup U_i = M$, as well as a so called *homeomorphism*, φ_i. A homeomorphism is essentially a transformation between two topological spaces that fulfills

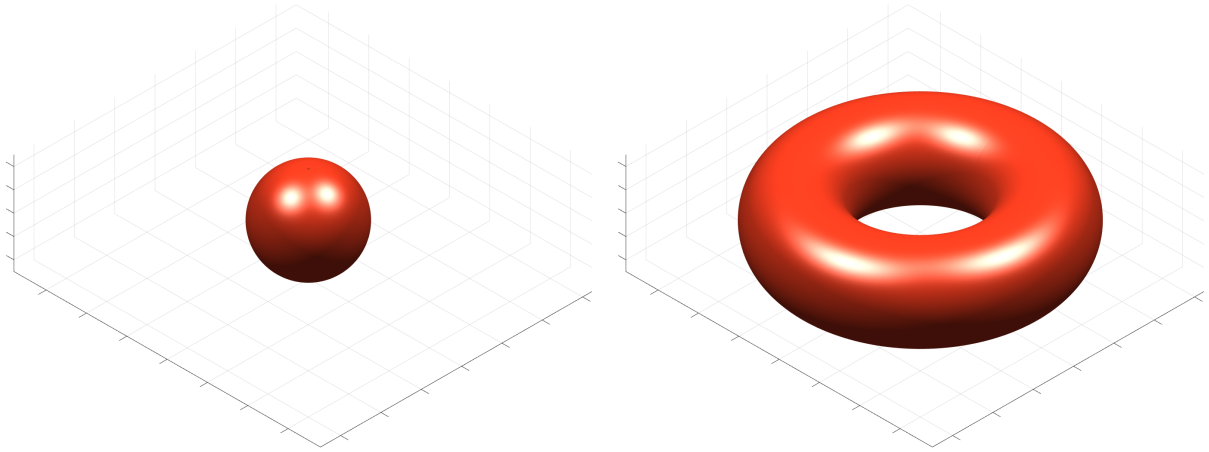
- φ_i is bijective,
- φ_i is continuous,
- φ_i^{-1} is continuous.

The orientability of a manifold can in the case of a two dimensional manifold be conceptually understood by an example. Let M be the classical, two dimensional topological manifold the *Möbius strip* shown in Figure 2.2b. Imagine that an atlas is created for M and that there is only one chart in it. This chart is essentially the two dimensional surface given by just cutting the Möbius strip in two and flattening it on a plane. The chart now looks like a rectangle where the right and left side of the rectangle are the boundary segment that was created by cutting the Möbius strip. If picking a point on the right side of the rectangle a point on the opposite side is essentially picked as well (since both the sides describe the same curve on the actual Möbius strip). Now if moving the point up on the right side the point is moved down on the left side, i.e. the chart is not bijective and the Möbius strip is therefore a nonorientable.

This conceptual reasoning can be generalized for an arbitrary number of charts, it is just a matter of moving across the charts to finally come the same conclusion at the chart that "closes" the Möbius strip. The idea here suggest that the Möbius strip is nonorientable because it is impossible to create an atlas that has a consistent representation of how to move on the surface, i.e. how the tangents of the surface are organized. The tangent of a surface is closely related to the derivatives of the same surface and can through extended mathematical reasoning boil down to the general definition of orientable manifolds [CdGDS13].

An n-dimensional manifold M is *orientable* if for all coordinate changes in between an atlas, $\{(U_i, \varphi_i : U_i \to V_i \subset \mathbb{R}^n)\}$, and any new atlas, $\{(U_j, \varphi_j : U_j \to V_j \subset \mathbb{R}^n)\}$, the determinant of the derivative of

$$\varphi_i \varphi_j^{-1} : \varphi_j(U_i \cap U_j) \to \varphi_i(U_i \cap U_j), \tag{2.2}$$



(a) Sphere in $\boldsymbol{R}^3$. *(b) Torus in* $\boldsymbol{R}^3$.

Figure 2.1: *Two orientable and two dimensional pure manifolds embedded in* $\boldsymbol{R}^3$.

is positive. This statement is only true for smooth manifolds and is not directly realized for discrete manifolds. For further reading concerning the orientability of manifolds in general and orientability of discrete manifolds in particular please see [CdGDS13].

Apart from being orientable the manifolds must also be differentiable. This can be given a sloppy explanation by saying that the manifolds must locally resemble an euclidean space well enough that it is possible to perform calculus on it. This criteria is especially tricky to condense into mathematics for discrete manifolds but is well explained in [CdGDS13].

Furthermore the manifold must have a connectivity constructed in such a way that it describes a two dimensional structure embedded within a three dimensional space. In other words the manifold must in the local neighborhood around every point describe a two dimensional surface that is composed out of three dimensional components, no other combination of dimensionalities are accepted.

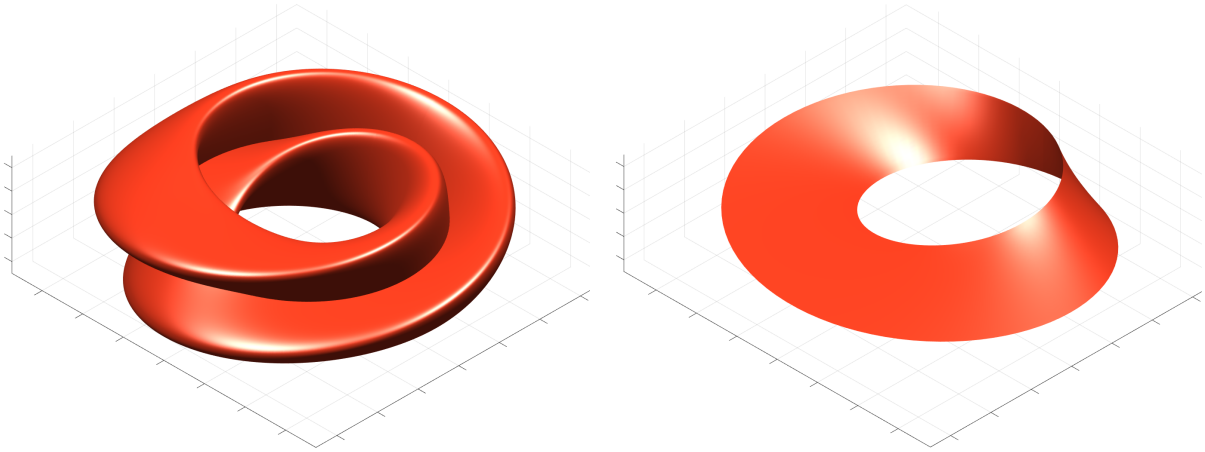
Typical examples of manifolds that globally fulfill these conditions are spheres and toruses shown in Figure 2.1 while examples of manifolds that does not fulfill these criteria are shown in Figure 2.2. The examples in Figure 2.2 does not fullfill the criteria since they are nonorientable.

The reason for theses manifold criteria only being demanded locally is because there might be local defects on the input mesh which if treated globally would render the entire mesh invalid even if there is just one tiny defect causing trouble. Hence the criteria must only be locally met if some matching is to take place at all. Naturally, matching is skipped for local part where these conditions are not met.

Figure 2.3 shows a pair of input meshes that satisfy these conditions. Figure 2.3a shows the mesh that throughout this thesis will be the example *source mesh* while 2.3b shows a corresponding example *target mesh*. The source mesh is the mesh from which patches are extracted and the target mesh is the mesh onto which these patches are matched.

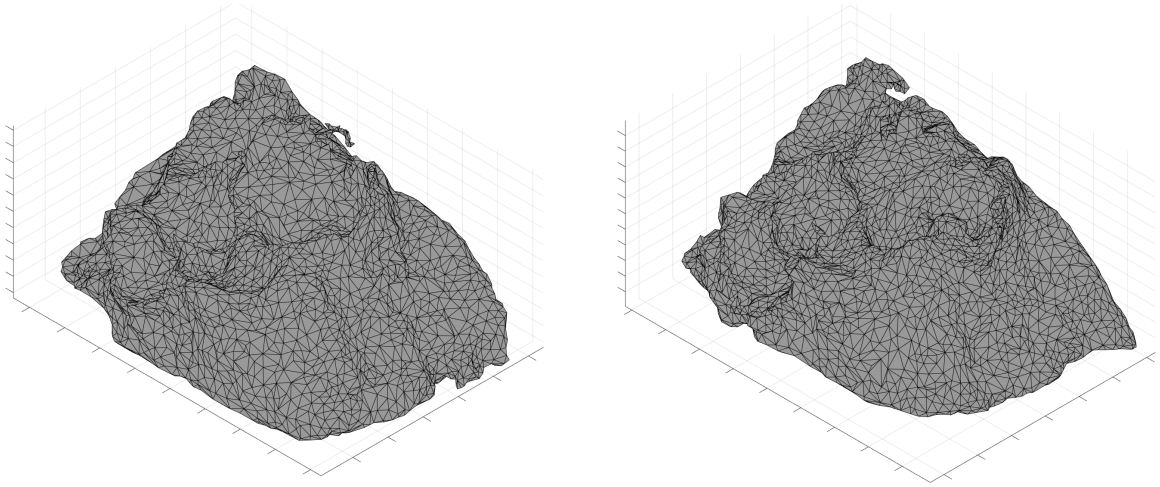
2.3 Mesh cleaning

Both the source and target mesh are also "cleaned" before entering the next step in the algorithm. This procedure of cleaning removes all manifolds (an non manifolds) below a certain size, in terms



(a) *Klein bagel in* $\boldsymbol{R}^3$. *(b)* *Möbius strip in* $\boldsymbol{R}^3$.

Figure 2.2: *Two nonorientable and two dimensional manifolds embedded in* $\boldsymbol{R}^3$.



(a) *Source mesh.* *(b)* *Target mesh.*

Figure 2.3: *A pair of meshes that satisfy the conditions for allowed input meshes.*

of vertex count, since it is convenient to only deal with as few manifolds as possible. The removal of small manifolds will in most cases not strongly affect the compression ratio since the biggest part of the sequence preferably consists of larger manifolds, rendering the small manifolds of much less interest from a compression point of view.

Chapter 3

Matching voxels

This section describes the way of converting a pair of meshes to their corresponding voxelizations thereby enabling easier analysis of the similarity between local features. The analysis of similar features is strongly based on tensor algebra and spherical harmonics. The foundation for this analysis is used in effective 3D-imaging processing techniques for analyzing images of cellular structures.

3.1 Voxelization

A property of meshes that can prove quite cumbersome to work around is the lack of a well defined resolution. A mesh constructed of connected vertices have no resolution since it is not a discrete function in $\mathbb{R}^3$ but rather a space itself on which a function can be defined. Comparing local neighborhoods of a mesh with each other in hopes of finding similarities is therefore cumbersome since there is no simple way of defining what local neighborhoods are in between spaces. This problem can however be partly resolved by voxelezing the mesh, presuming it is a closed mesh, i.e. defining a discrete function, $F : \mathbb{R}^3 \to \mathbb{R}$, with a well defined resolution that represents a voxelized volume confined by the mesh.

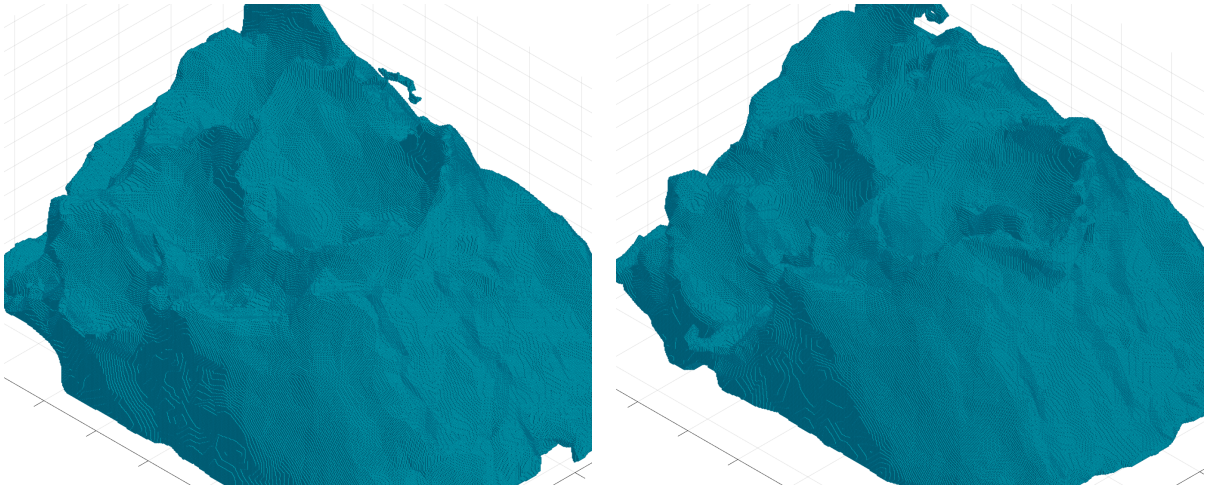
Let V define the volume enclosed by a closed manifold $M \subset \mathbb{R}^3$, the function $F_{fine}(\mathbf{x})$ can then be defined as

$$F_{fine}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in V \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

at some resolution, λ_{fine}, defined as the highest resolution (the finest grid). The function F_{fine} is regarded as a representation of the ground truth shape of M, i.e. the resolution used in F_{fine} is considered good enough to represent the finest details of M. All discrete functions, $F(\mathbf{x})$, defined for any lower resolution can be expressed by

$$F(\mathbf{x}) = \begin{cases} \alpha(\mathbf{x}), & \text{if } \mathbf{x} \in V \\ 0, & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\alpha(\mathbf{x}) \in [0\ 1]$ is the ratio between filled and empty voxels at the finest resolution for those fine resolution voxels that are contained in the voxel at the resolution of interest, λ. Figure 3.2 shows the voxelizations of the meshes shown in Figure 2.3 for a resolution, $\lambda = 1/16$, which is lower than the finest resolution, $\lambda = 1/265$, voxelization shown in Figure 3.1.



(a) *Closeup of the finest resolution voxelization of mesh shown in Figure 2.3a.*

(b) *Closeup of the finest resolution voxelization of mesh shown in Figure 2.3b.*

Figure 3.1: *A closeup of the finest resolution, $\lambda = 1/265$, voxelizations of the meshes shown in Figure 2.3.*

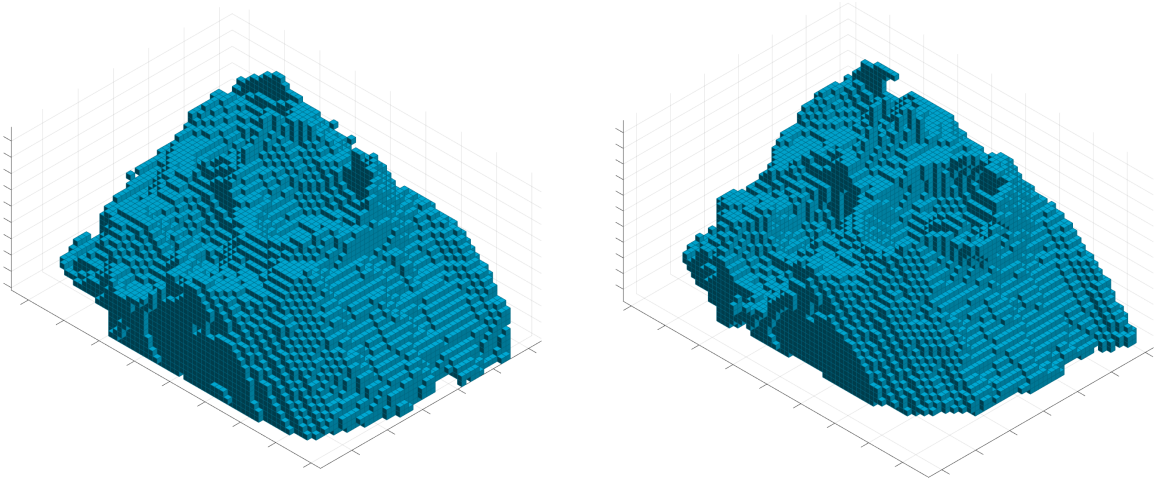
The process of representing a surface with a voxelization could potentially destroy some vital information in the case of self intersecting surfaces. These cases are considered as missed opportunities for a matching rather than a failure. The same goes for potential matchings of patches in between different manifolds within the raw data since the raw data, for simplicity, is initially filtered so that each frame only contains a low amount of manifolds.

3.2 Rotationally invariant Gabor descriptors

When a discrete function representing the shape of the mesh is acquired local rotational invariant features, called descriptors, can be extracted based on spherical harmonics functions. These descriptors are the key to extract what parts of the voxelization resemble similar local structures. Several papers, e.g. [SRS$^+$11, LSS$^+$11, SR13, RRK$^+$06], describe different methods for extracting rotationally invariant features with different approaches, benefits and drawbacks. In [SRS$^+$11] it is shown how to extract these descriptors with different spherical harmonics basis. The spherical Gabor descriptors was in [SRS$^+$11] the type of descriptor that showed best performance when trying to find similar local features of various 3D-images, i.e. voxelizations. Therefore the Gabor descriptors approach, calculated with the second order scheme suggested in [SRS$^+$11] is chosen as the descriptor for the extraction of rotational invariant features.

The extraction of the Gabor descriptors is only done for the voxels in the shell of voxelization since those voxels are the ones that resembles the mesh. The interior of the voxelization is of less interest since it is not related to any particular part of the mesh. Still, the interior is important since it basically represents the orientation of the voxelization and therefore the mesh it is derived from.

The extraction of the rotational invariant Gabor descriptors for a voxel can be described by the following steps. Let $f : S_2 \to \mathbb{C}$ be a spherical tensor valued function and $\mathbf{Y}^l : S_2 \to \mathbb{C}^{2l+1}$ form an orthogonal tensor basis for functions on the unit sphere of rank l. Then f can be



(a) *Voxelization of mesh shown in Figure 2.3a.*

(b) *Voxelization of mesh shown in Figure 2.3b.*

Figure 3.2: *The voxelizations of the meshes shown in Figure 2.3 for resolution* $\lambda = 16$.

described as

$$f(\mathbf{r}) = \sum_{l=0}^{\infty} (\mathbf{a}^l)^T \mathbf{Y}^l(\mathbf{r}). \tag{3.3}$$

Since $\mathbf{Y}^l$ have orthogonal properties the expansion coefficients $\mathbf{a}^l \in \mathbb{C}^{2l+1}$ are obtained by projecting f onto $\mathbf{Y}^l$ yielding $\mathbf{a}^l := \frac{2l+1}{2\pi}\langle f, \mathbf{Y}^l \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the inner product. Rotating f by $\mathbf{U}_g \in \mathbb{R}^{3\times 3}$ results in the rotated function $f'(\mathbf{r}) = f(\mathbf{U}_g \mathbf{r})$, where $\mathbf{U}_g$ is a rotation matrix with g denoting an element of the rotation group $g \in SO(3)$. The rotated function, $f'(\mathbf{r})$, can then be described as

$$f'(\mathbf{r}) = \sum_{l=0}^{\infty} (\mathbf{D}_g^l \mathbf{a}^l)^T \mathbf{Y}^l(\mathbf{r}), \tag{3.4}$$

where $\mathbf{D}_g^l \in \mathbb{C}^{(2l+1)\times(2l+1)}$ is the unitary, irreducible representation of an element of the rotation group $g \in SO(3)$ also known as Wigner D-Matrices [Ros63]. The rotationally invariant property needed to extract the descriptors is shown in (3.5) where $(\mathbf{a}')^l$ are rotated expansion coefficients and the rotational invariant property of each descriptor is the power spectrum of the expansion coefficients [KFR03].

$$||(\mathbf{a}')^l||^2 = \langle (\mathbf{a}')^l, (\mathbf{a}')^l \rangle = \langle \mathbf{D}_g^l \mathbf{a}^l, \mathbf{D}_g^l \mathbf{a}^l \rangle = \langle (\mathbf{D}_g^l)^* \mathbf{D}_g^l \mathbf{a}^l, \mathbf{a}^l \rangle = \langle \mathbf{a}^l, \mathbf{a}^l \rangle = ||\mathbf{a}^l||^2 \tag{3.5}$$

This property basically says that if the power spectrum of two descriptor's expansion coefficients are similar there is a great chance they describe similar looking, underlying mesh structures.

The expansion coefficients $\mathbf{a}^l$ can be computed by *coupling operations* on the spherical tensors. Let the function $\mathbf{f} : \mathbb{R}^3 \to \mathbb{C}^{2l+1}$ be a spherical tensor field of rank l that satisfies

$$\forall g \in SO(3): \quad (g\mathbf{f})(\mathbf{r}) := \mathbf{D}_g^l \mathbf{f}(\mathbf{U}_g^T \mathbf{r}), \tag{3.6}$$

where $\mathbf{U}_g \in \mathbb{R}^{3\times3}$ is the corresponding real valued rotation matrix, $\mathbf{D}_g^l \in \mathbb{C}^{(2l+1)\times(2l+1)}$ is the unitary irreducible representation of an element of the rotation group $g \in SO(3)$. Now all spherical harmonics, $\mathbf{Y}^l$, of a certain rank l can be described as

$$(g\mathbf{Y}^l)(\mathbf{r}) := \mathbf{D}_g^l \mathbf{Y}^l(\mathbf{U}_g^T \mathbf{r}) = \mathbf{Y}^l(\mathbf{r}), \tag{3.7}$$

where $\mathbf{Y}^l : S_2 \to \mathbb{C}^{2l+1}$, i.e. $\mathbf{Y}^l$ form an orthogonal basis for functions on the unit sphere of rank l. In spherical tensor algebra there is a way of coupling two tensors of rank $l_1 \geq 0$ and $l_2 \geq 0$, respectively, so that the result of the coupling has rank l. Let $\circ_l : \mathbb{C}^{2l_1+1} \times \mathbb{C}^{2l_2+1} \to \mathbb{C}^{2l+1}$ be the family of bilinear forms that couples two tensors of rank $l_1 \geq 0$ and $l_2 \geq 0$ with the property of preserving the tensor rotation yielding a new tensor of rank l [AF09] . It then follows that $\mathbf{Y}^l = \mathbf{Y}^{l_1} \bullet_l \mathbf{Y}^{l_2}$ where $\bullet_l$ denotes the normalized tensor product between two tensors $\mathbf{v}$ and $\mathbf{u}$ of rank $l_1 \geq 0$ and $l_2 \geq 0$, respectively, defined as

$$(\mathbf{v} \bullet_l \mathbf{u}) = \frac{(\mathbf{v} \circ_l \mathbf{u})}{\langle l_1 0, l_2 0 | l 0 \rangle}. \tag{3.8}$$

This property can be used to describe the spherical derivative operators acting on a tensor field which can both raise and lower the rank of a tensor field [AF09]. Let

$$\begin{aligned} \nabla^1 \mathbf{f}_l &:= \nabla \bullet_{l+1} \mathbf{f}_l \quad \text{and} \\ \nabla_1 \mathbf{f}_l &:= \nabla \bullet_{l-1} \mathbf{f}_l, \end{aligned} \tag{3.9}$$

be the spherical derivative for raising and lowering the rank of the tensor field $\mathbf{f}_l$ of rank l with $\nabla = \left(\frac{1}{\sqrt{2}}\left(\partial_x - i\partial_y\right), \partial_z, -\frac{1}{\sqrt{2}}\left(\partial_x + i\partial_y\right)\right)$ with respect to cartesian coordinates. The partial derivatives ∂_x, ∂_y and ∂_z can be discretized with any type of common differential scheme. The scheme here is a standard centralized second order scheme on the form

$$\partial f_i = \frac{-f_{i-1} + f_{i+1}}{2}. \tag{3.10}$$

The spherical derivative ∇_n^m, where m denotes the number of times the derivative should be applied to raise the rank and n denotes the number of times the derivative should be applied to lower the rank, can now be described as

$$\nabla_n^m \mathbf{f}^0 := \underbrace{\left(\nabla \bullet_{m-n} ... \left(\nabla \bullet_{m-1}\right.\right.}_{n \text{ times } \nabla_1} \underbrace{\left.\left(\nabla \bullet_m ... \left(\nabla \bullet_1 \mathbf{f}^0\right)\right)\right)}_{m \text{ times } \nabla^1}, \tag{3.11}$$

where $\mathbf{f}^0$ is a scalar field. In [AF09] it is shown that this can be simplified into

$$\nabla_n^m \mathbf{f}^0 := \nabla^{m-n} \Delta^n \mathbf{f}^0, \tag{3.12}$$

where Δ^n denotes the standard laplacian applied n times to the scalar field $\mathbf{f}^0$. Defining a proper basis for the spherical harmonics analysis can be done in many ways. However, it is important to define a basis that describes the properties of interest well. In [SRS+11] several different basis functions are presented and the Gabor basis is the one that shows the best performance. The Gabor basis of rank 0, $\mathbf{G}_s^0(\mathbf{r}, k)$, is essentially a Gaussian windowed Bessel function of the first kind, defined as

$$\mathbf{G}_s^0(\mathbf{r}, k) := j_0\left(k\frac{r}{\sigma}\right) e^{\frac{-r^2}{s\sigma^2}}. \tag{3.13}$$

By applying a Gaussian window to the Bessel basis it is possible to filter the data in a way that treats features that are close to the origin as more important than features that are far away. This filtering ensures that small anomalies in the outskirts of the local neighborhood are ignored while searching for similar features, which is desirable since an identical descriptor match is unlikely to be found.

Applying the method in (3.12) of increasing the rank of a tensor to the Gabor basis of rank 0 makes it possible to describe the expansion coefficient tensors of higher tensor rank. The expansion coefficient of rank l for any k, $\mathbf{a}_k^l(\mathbf{x}) \in \mathbb{C}^{2l+1}$, located at $\mathbf{x}$ in the voxelization $F(\mathbf{x})$ is then given by

$$\mathbf{a}_k^l(\mathbf{x}) = \left(\frac{\sigma}{-k}\right)^l \bar{\nabla}^l (F \star \mathbf{G}_s^0(k))(\mathbf{x}), \tag{3.14}$$

where $\bar{\nabla}^l$ denotes the complex conjugate of the operator described in (3.12) and the $\star$ denotes the standard cross correlation between the voxelization and the Gabor basis of rank 0 [SRS+11]. Note that since $g \in SO(3)$ the coordinate system has to be translated so that the origin is placed in the center of the voxel for which the descriptros are currently calculated. Algorithmically this method of increasing the tensor field rank can be implemented recursively

$$(F \star \mathbf{G}_s^0(k))(\mathbf{x}) = \mathbf{a}_k^0 \xrightarrow[T_0 \to T_1]{\bar{\nabla}^1} \mathbf{a}_k^1 \xrightarrow[T_1 \to T_2]{\bar{\nabla}^1} \mathbf{a}_k^2 \xrightarrow[T_2 \to T_1]{\bar{\nabla}^1} \mathbf{a}_k^3 \ldots \mathbf{a}_k^l \tag{3.15}$$

where $T_l \to T_{l+1}$ denotes that the rank is increased from l to $l+1$. The indices l and k can be interpreted as frequency components in the spherical tensors, i.e. if the $\bar{\nabla}^l$ operator is applied a higher number of times it is possible to capture higher angular frequency patterns in the shape of the local neighborhood of the voxel, and the same is also true for higher values of k which captures higher radial frequencies. The implementation for applying the ∇^1 operator with a second order scheme can be observed in Algorithm 2 with input being the voxelization and the expansion coefficients of one rank lower than the one desired.

Calculating how well two voxels match each other in terms of their descriptors can be done in numerous ways. The way to calculate the degree of matching used is in correspondence with the suggested way in [SRS+11]. Consider the matrix constructed as

$$A_{k,l}(\mathbf{x}) = \begin{pmatrix} ||\mathbf{a}_1^1(\mathbf{x})||^2 & ||\mathbf{a}_1^1(\mathbf{x})||^2 & \cdots & ||\mathbf{a}_1^l(\mathbf{x})||^2 \\ ||\mathbf{a}_2^1(\mathbf{x})||^2 & ||\mathbf{a}_2^1(\mathbf{x})||^2 & \cdots & ||\mathbf{a}_2^l(\mathbf{x})||^2 \\ \vdots & \vdots & \ddots & \vdots \\ ||\mathbf{a}_k^1(\mathbf{x})||^2 & ||\mathbf{a}_k^1(\mathbf{x})||^2 & \cdots & ||\mathbf{a}_k^l(\mathbf{x})||^2 \end{pmatrix}, \tag{3.16}$$

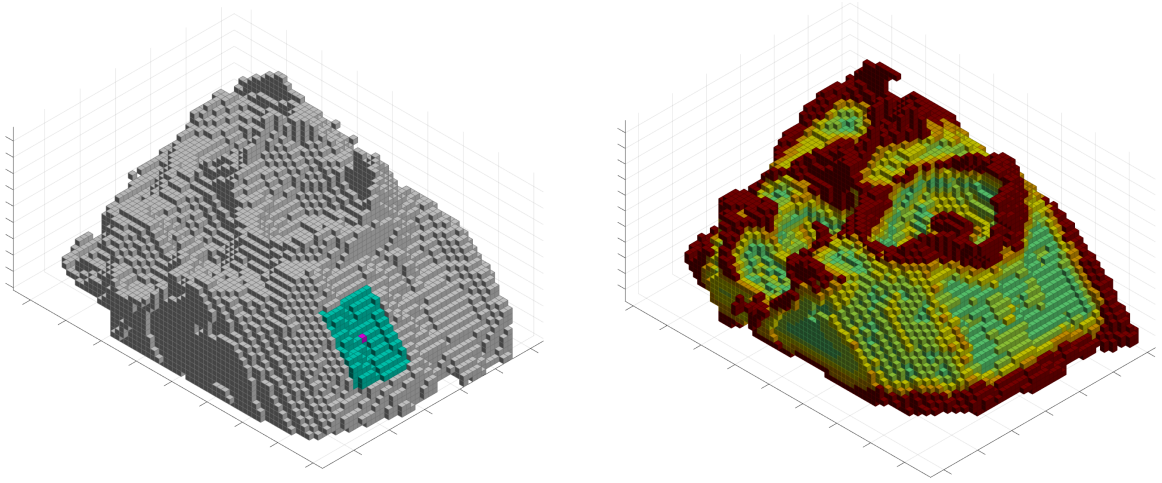
for each voxel. The measure, $E_{k,l}(\mathbf{x}_1, \mathbf{x}_2)$, for how well two voxels, at location $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, match each other is suggested as

$$E_{k,l}(\mathbf{x}_1, \mathbf{x}_2) = |A_{k,l}(\mathbf{x}_1) - A_{k,l}(\mathbf{x}_2)|_1. \tag{3.17}$$

Note that $E_{k,l}(\mathbf{x}_1, \mathbf{x}_2)$ is a scalar entity and a low value represents a good match while a higher value represents a poor match. Applying this method of calculating the degree of matching between two voxels to a set of two voxelized frames can be seen in Figures 3.3, 3.4 and 3.5. The local domain around each voxel, with the side length, β, of the domain being $\beta = 11$ in each cartesian dimension, from which information used in the Gabor descriptor calculation is extracted is colored in turquoise. In Figure 3.3 the shape matching against is a flat surface having the source voxel shown in 3.3a, while in Figure 3.4 it is the shape of an edge. In Figure 3.5 the shape matched against is a quite unique shape only present in a small amount of voxels.

Algorithm 2 *Outline of the algorithm used for implementing the ∇^1 operator, where $\mathbf{N}^3$ denotes the set of voxels which should be included in the calculation. A larger set will yield higher precision. Note that the set $\mathbf{N}^3$ has an allowed minimum, dependent on the desired number of angular frequency components l_{max}. The width, W, of the set $\mathbf{N}^3$, i.e. the number of voxels contained withing the set along a dimension, needs to satisfy the relation $W_i \geq 2l+1$ in all three euclidean dimensions. The special case of calculating the descriptor coefficients for a voxel close to the boundary of the voxelization is simply done by regarding all points outside of the voxelization as empty voxels, i.e with an $\alpha(\mathbf{x}) = 0$.*

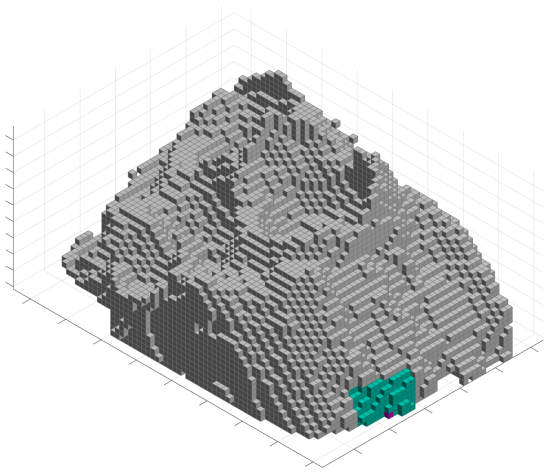
Function: $\nabla^1 : T_l \to T_{l+1}$
Input: $\mathbf{N}^3$, $\mathbf{f}^l \in T_l$
for $\forall \mathbf{x} \in \mathbf{N}^3$ **do**
  for $m = -(l+1)$ to $(l+1)$ **do**
    $\mathbf{f}_m^{l+1}(\mathbf{x}) = 0$
    if $|m+1| \leq l$ **then**
      $\mathbf{f}_m^{l+1}(\mathbf{x}) = \mathbf{f}_m^{l+1}(\mathbf{x}) + \frac{\sqrt{(l-m)(1+l-m)}}{2(l+1)}(\partial_x \mathbf{f}_{m+1}^l - i\partial_y \mathbf{f}_{m+1}^l)(\mathbf{x})$
    end if
    if $|m| \leq l$ **then**
      $\mathbf{f}_m^{l+1}(\mathbf{x}) = \mathbf{f}_m^{l+1}(\mathbf{x}) + \frac{\sqrt{(1+l+m)(1+l-m)}}{l+1}(\partial_z \mathbf{f}_m^l)(\mathbf{x})$
    end if
    if $|m-1| \leq l$ **then**
      $\mathbf{f}_m^{l+1}(\mathbf{x}) = \mathbf{f}_m^{l+1}(\mathbf{x}) - \frac{\sqrt{(l+m)(1+l+m)}}{2(l+1)}(\partial_x \mathbf{f}_{m-1}^l + i\partial_y \mathbf{f}_{m-1}^l)(\mathbf{x})$
    end if
  end for
end for
Return: $\mathbf{f}^{l+1}(\mathbf{x})$



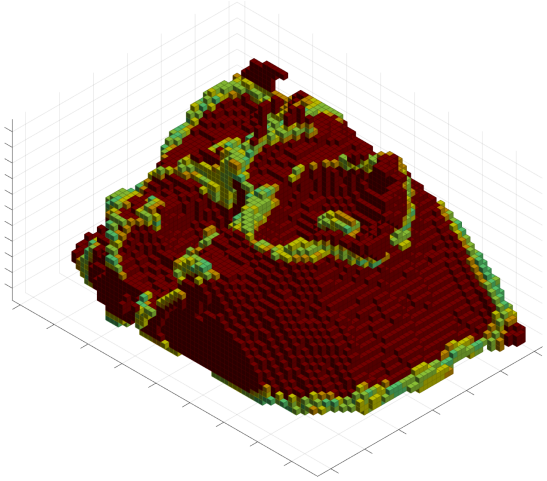
(a) *The source voxel shape, in this case a locally flat surface.*

(b) *Target voxelization, color coded corresponding to the degree of matching.*

Figure 3.3: *The degree of matching between the source voxel (and its local neighborhood) and the target voxelization. Green represents a good match and red represents a worse match. The turquoise colored voxels in the source voxelization represent the voxels which have contributed to the descriptor for the selected source voxel (the magenta colored voxel).*

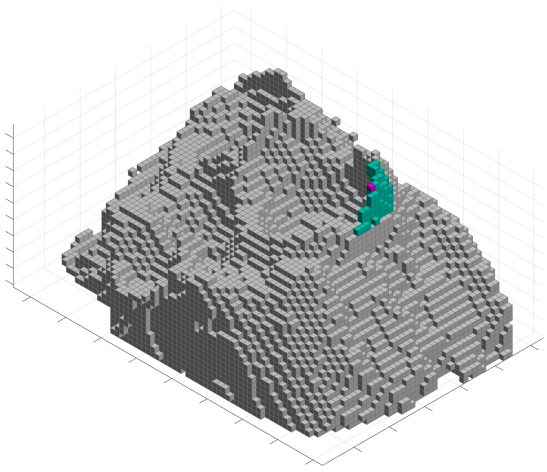


(a) *The source voxel shape, in this case an edge shape.*

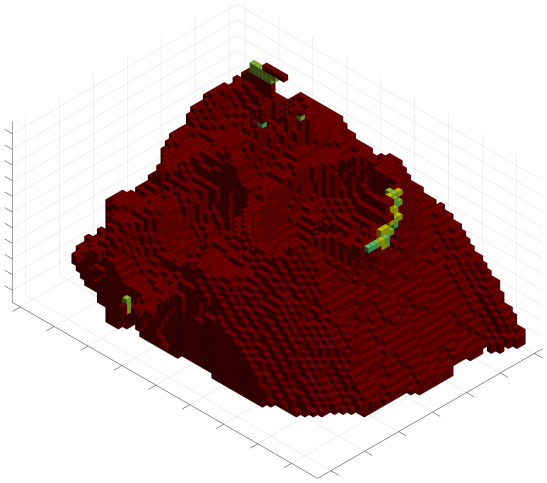


(b) *Target voxelization, color coded corresponding to the degree of matching.*

Figure 3.4: *The degree of matching between the source voxel (and its local neighborhood) and the target voxelization. Green represents a good match and red represents a worse match. The turquoise colored voxels in the source voxelization represent the voxels which have contributed to the descriptor for the selected source voxel (the magenta colored voxel).*

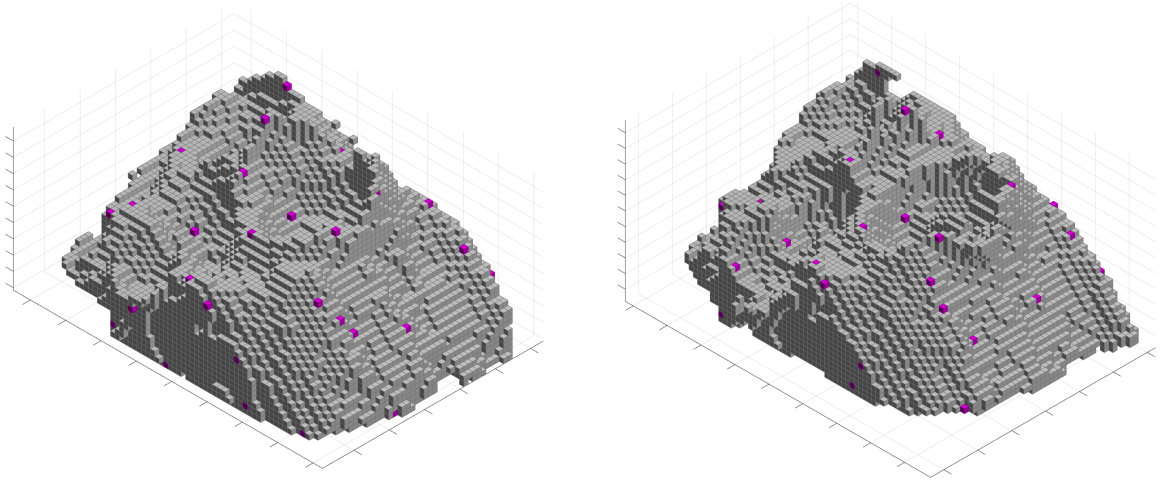


(a) *The source voxel shape, in this case quite unique shape.*



(b) *Target voxelization, color coded corresponding to the degree of matching.*

Figure 3.5: *The degree of matching between the source voxel (and its local neighborhood) and the target voxelization. Green represents a good match and red represents a worse match. The turquoise colored voxels in the source voxelization represent the voxels which have contributed to the descriptor for the selected source voxel (the magenta colored voxel).*



(a) Seed points on source voxelization. *(b) Seed points on target voxelization.*

Figure 3.6: *A set of seed point voxel pairs for the source and target voxelizations.*

3.3 Extract seed points

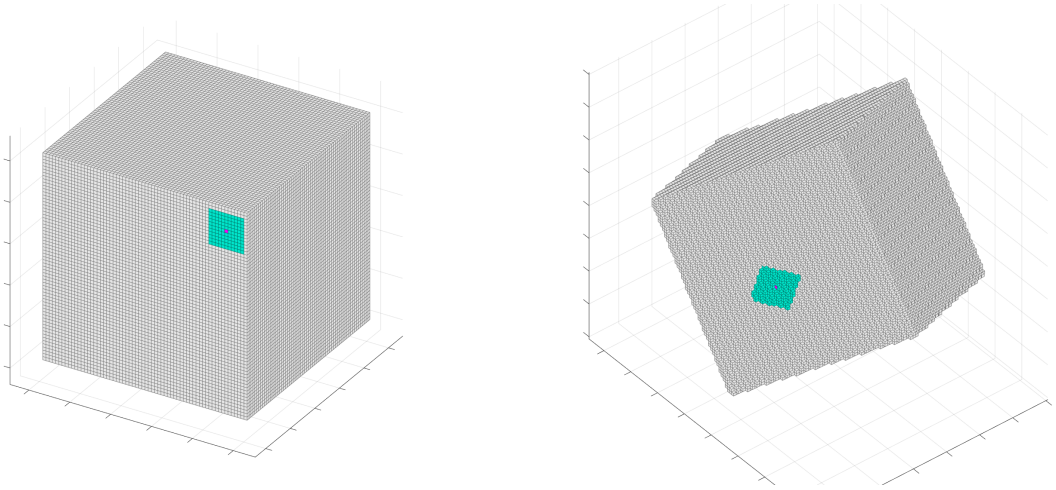
In order to make the conditions for matching a patch of a mesh to a patch of another mesh the best set of paired seed points is first extracted. A seed point pair is in itself a set of two corresponding voxels that within a local region of the voxelizations are the best matching voxels in terms of Gabor descriptors. The corresponding seed point vertices are set to be the vertices that are the closest to the seed point voxels. By locating seed points in this way the odds of the local neighborhood around the vertices matching each other is much greater than if just randomly trying to find similar patches for each mesh. A set of seed points for two voxelizations can be seen in Figure 3.6.

In order to ensure that the area around the seed points describe matching local patches that align well with adjacent locally matched patches a criterion for the curvature around a seed point has to be met. If entirely flat surfaces are compared it is very hard to extract seed point that have some form of unique properties. Take the extraction of seed point for a voxelized cube for example. If seed points were extracted without the minimum curvature criterion, as seen in Figure 3.7, they could potentially represent very bad starting points around which to extract a pair of patches since two flat surfaces could fit anywhere on each other. Such a match could in itself be considered a good match and yield a low error but would probably result in a bad global matching of several connected patches since the alignment could be completely wrong. If on the other hand seed point must meet the curvature criterion they generally represent a unique feature much better, as seen in Figure 3.8.

There are several methods and papers published on discrete curvature estimation [GS03, KMS12, MKY01]. The curvature for the vertices corresponding to the extracted seed point voxels is calculated according to the method of fitting an extended quadric function of the form

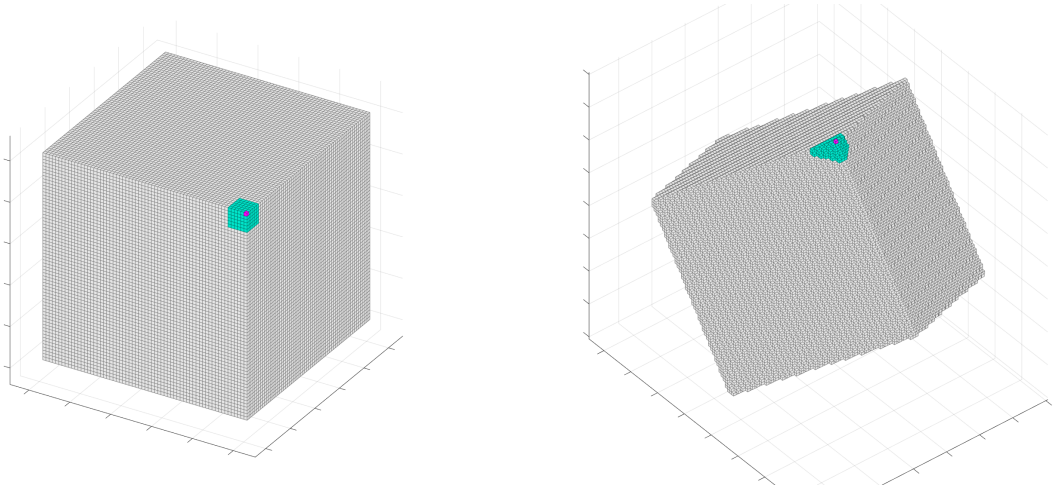
$$z = ax^2 + bxy + cy^2 + dx + ey + f \tag{3.18}$$

for the parameters a, b, c, d, e and f to the local neighborhood of vertices around a certain vertex. The method is described in [GS03] and is used due to its simplistic approach and ease of implementation. The function is fitted in a local coordinate system around the selected vertex



(a) *Source voxelization of a cube.* ***(b)*** *Target voxelization of a rotated cube.*

Figure 3.7: *An example of a seed point pair on the voxelizations of a cube when the curvature criterion is inactive. As clearly visible the local neighborhood around the seed points match each other very well but in terms of global alignment the seed points are a poor choice.*



(a) *Source voxelization of a cube.* ***(b)*** *Target voxelization of a rotated cube.*

Figure 3.8: *An example of a seed point pair on the voxelizations of a cube when the curvature criterion is active.*

where the local coordinate system, $\{\hat{\mathbf{e}}_i\}$, is defined by

$$\hat{\mathbf{e}}_1 = \frac{(\mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^T)\hat{\mathbf{i}}}{|\mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^T|\hat{\mathbf{i}}}, \qquad \hat{\mathbf{e}}_3 = \hat{\mathbf{n}}, \qquad \hat{\mathbf{e}}_2 = \hat{\mathbf{e}}_3 \times \hat{\mathbf{e}}_1 \tag{3.19}$$

where $\hat{\mathbf{n}}$ is the vertex normal for the selected vertex and $\hat{\mathbf{i}}$ is the global x-axis for the entire mesh. When a set of m vertices, $\{(x_j, y_j, z_j)\}$ $j \in [1, m]$,including the selected vertex are extracted and transformed from the global coordinate system to $\{\hat{\mathbf{e}}_i\}$ the function parameters a, b, c, d, e and f are acquired by solving the over determined system $\mathbf{Ax} = \mathbf{b}$ in a least square manner like

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{3.20}$$

where

$$\mathbf{A} = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m^2 & x_my_m & y_m^2 & x_m & y_m & 1 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix} \tag{3.21}$$

and $m \geq 6$.

A beautiful property of the quadric function is that the curvature can be calculated analytically. The parameters a, b, c, d and e are the essential parameters that are needed in order to extract the curvature for a specific fitting. For each seed point vertex the absolute value of the Gaussian curvature is the entity required to have a value higher than a certain threshold value where the Gaussian curvature, K, is calculated according to

$$K = \frac{4ac - b^2}{(1 + d^2 + e^2)^2}. \tag{3.22}$$

Chapter 4

Matching mesh patches

At some point in the algorithm the step back from voxelization to an actual mesh has to be made since the result shall describe matching features on the meshes, not the voxelizations. This section describes the methods used for extracting a patch on the source mesh and the corresponding patch on the target mesh as well as the mathematics behind the bijective mapping of any point on the source patch to its corresponding point on the target patch.

4.1 Patch extraction and ICP

The extraction of the mesh patches is done by geodesically expanding the local neighborhood around the seed point vertices until a certain patch size is found. However, saying that the corresponding vertex for a seed point voxel is the closest one can in some cases be problematic. If the mesh is very unequally spaced in terms of vertex placement the closest vertex for a voxel might be far away depending on the resolution of the voxelization compared to the edge lengths in between vertices. In those cases it would be preferable to get the actual closest point to the voxel on the mesh and represent it in terms of triangle index and barycentric coordinates. When eventually extracting a patch around the closest point on the mesh the barycentric coordinate information could be used to get a more precise patch extraction rather than just expanding around a certain vertex. This technique is however not implemented in this algorithm since the iterative closest point (ICP) matching technique will solve much of that problem if the target patch is simply expanded a bit further, i.e. made a bit larger than the source patch.

After the extraction the patches are validated to make sure that both of them fulfill the manifold criteria described in Section 2.2. This validation is done by going through the connectivity data for the patches and making sure that for each patch the following is fulfilled:

1. No edge is shared between more than two triangles.
2. All triangles shares at least one edge with another triangle.
3. All triangles sharing a vertex must be connectable[1] with each other by only the triangles sharing the vertex.

If all these conditions are met the patch is valid. When two valid patches, one extracted from each mesh, are acquired they must be relocated in space in order to best align with each other.

[1]The meaning of two triangles, T_1 and T_2, being *connectable* by a set of triangles, $S = \{T_i\}$, is that it is possible to traverse the set from T_1 to T_2 by only moving orthogonally across triangle edges.

This is achieved by using ICP matching [CM92] and specifically the software package provided by [KW10]. The ICP matching basically consists of the steps described in Algorithm 3.

Algorithm 3 *Outline of the ICP algorithm, where v_n represents a vertex in the first set of vertices $V_1 \in \mathbb{R}^{3\times N}$, ϵ is the error and ϵ_{max} is the maximum allowed error.*

1: Function: $\mathbf{F} : \mathbb{R}^{3\times N} \rightarrow \mathbb{R}^{3\times N}$
2: Input: $\mathbf{A} \in \mathbb{R}^{3\times N}$
3: **while** $\epsilon > \epsilon_{max}$ **do**
4: **for** $v_i \in V_1$ **do**
5: Find the closest vertex in the second set of vertices V_2.
6: **end for**
7: Estimate the six degrees of freedom, i.e. the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$.
8: Evaluate the error ϵ.
9: Transform all vertices in the set V_1.
10: **end while**
11: Return: $\mathbf{A}' \in \mathbb{R}^{3\times n}$

A more detailed description involving additional steps, e.g. weighting, edge rejection and error minimization can be found in [KW10]. A modified version of the standard ICP matching algorithm is used to fit a point cloud to a discrete surface instead of fitting a point cloud to a point cloud since that might give a bad result if the patches are unequally spaced in terms of vertex placement. In the case of fitting a point cloud to a discrete surface the error metric is changed from

$$\epsilon = \sum_{i=1}^{N} ||\mathbf{R}\mathbf{v}_i + \mathbf{t} - \mathbf{w}_i||^2, \tag{4.1}$$

to

$$\epsilon = \sum_{i=1}^{N} \left[(\mathbf{R}\mathbf{v}_i + \mathbf{t} - \mathbf{w}_i) \cdot \mathbf{n}_i\right]^2, \tag{4.2}$$

where $\mathbf{v}_i$ is a vertex in the set V_1, $\mathbf{w}_i$ is its corresponding closest vertex in V_2, $\mathbf{R}$ is the transformation rotation matrix, $\mathbf{t}$ is the transformation translation vector and $\mathbf{n}_i$ is the surface normal for the corresponding closest face in the set V_2. Note that the metric described in (4.1) and (4.2) is not a metric in a strict sens since the error scales quadratically with increased distance. This metric is however in accordance with the method suggested in [KW10]. The algorithm is also accelerated for better performance by using the K-D tree algorithm described in [Zha94].

4.2 Parametrization and bijective mapping

When a pair of patches has been extracted the actual functionality for bijectively mapping an arbitrary point on the source patch to exactly one point on the target patch is created by parameterizing the patches and aligning those parametrizations. The parametrization of a surface is the process of creating an atlas for a surface, i.e. finding a set of complete charts with both the patches U_i and the homeomorphisms φ_i.

Parametrization of surfaces is a widely used technique in computer graphics rendering, used for example when creating texture mappings of the global texture for a 3D-object. In all cases of parametrization of surfaces the general wish is to minimize the distortion of lengths and angles for a certain object, especially for texture mappings. Extensive research has been done in this field in order to achieve low distortion parametrizations [LM98, SCOGL02, JSS+14]. The theory of manifolds and how to correctly construct a parametrization of surfaces in general and discrete surfaces in particular is very well described in [CdGDS13].

However, building a complete system for parametrizations of discrete surfaces is in itself a large subject of research since there are no trivial and reasonably short ways of doing this. Building such a system is considered beyond the scope of this thesis and an in depth analysis of the various methods for surface parametrization will not be discussed here. Instead the very rigorous Matlab toolbox provided by [Pey09] is used for the parametrization of the patches.

When the patches have been relocated with the ICP algorithm and parametrized the parametrizations must be aligned in order to describe a map from the points on the source patch to the points on the target patch. The alignment of the parametric triangulation structure determines the exact mapping for an arbitrary point on the source patch to its corresponding mapped point on the target patch. Initiating the alignment is done by defining that the vertices on the boundary on the source patch should match to their closest point on the target patch. Note that this closest point can be any point on the target patch, i.e. it need not be a vertex, after relocation by the ICP matching algorithm. In order to achieve this the source patch parametrization triangulation is translated and distorted so that the points in $\mathbb{R}^2$ for the parametrization corresponding to the boundary vertices align with the coordinates in $\mathbb{R}^2$ on the target patch parametrization for the corresponding closest points on the target patch. This operation is a non-linear transformation of the set of vertices in the source patch parametrization, with corresponding connectivity to each other.

This problem can be modeled as an energy minimization problem and solved with various methods used to solve optimization problems in general. The suggested solution to this problem is implemented according to [SCOL+04] utilizing so called Laplacian mesh editing techniques, a technique specially developed for deforming meshes in an optimal way. A short version of the mathematics behind the Laplacian mesh editing is presented here but for the full version see [SCOL+04, ATLF06].

Let $P \subset \mathbb{R}^2$ denote a parametrization of a patch and be defined by a pair $(\mathbf{K}, \mathbf{V})$, where $\mathbf{K}$ represents the connectivity of the parametrization triangulation and $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ is the set of corresponding coordinates in $\mathbb{R}^2$ for the vertices in the parametrization. A key idea to this Laplacian mesh editing technique is to describe the patch P in terms of differentials, i.e. Laplacian coordinates $\mathbf{\Delta} = \{\delta_1, \delta_2, ..., \delta_n\}$, rather than absolute coordinates. A coordinate δ_i can be described by the difference in coordinates between the vertex $\mathbf{v}_i$ and the set of neighboring vertices, $\mathscr{N}_i = \{j | (i, j) \in \mathbf{K}\}$, as

$$\delta_i = \mathscr{L}(\mathbf{v}_i), \tag{4.3}$$

with $\mathscr{L}(\mathbf{v}_i)$ being defined as

$$\mathscr{L}(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in \mathscr{N}_i} \mathbf{v}_j, \tag{4.4}$$

where d_i is the number of neighbors in $\mathscr{N}_i$. The transformation from the absolute coordinates,

$\mathbf{V}$ to the laplacian coordinates $\mathbf{\Delta}$ can be realized by the operation

$$\mathbf{\Delta} = (\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})\mathbf{V}, \tag{4.5}$$

where $\mathbf{A}$ is the patch adjacency matrix and $\mathbf{D} = \text{diag}(d_1, d_2, ..., d_n)$. Let's say that the geometry should be changed from $\mathbf{V}$ to $\mathbf{V'}$. The energy in the system, $E(\mathbf{V'})$, as a function of the geometry V' can be described as

$$E(\mathbf{V'}) = \sum_{i=1}^{n} ||\mathbf{T}_i(\mathbf{V'})\delta_i - \mathscr{L}(\mathbf{v'}_i)||^2 + \sum_{i \in \mathscr{M}} ||\mathbf{v'}_i - \mathbf{u}_i||^2, \tag{4.6}$$

where $\mathscr{M}$ denotes the set of anchor vertices whose coordinates $\mathbf{u}$ should be fixed at some determined points and $\mathbf{T_i}$ is the transformation matrix for each individual vertex. Minimizing the energy can be done by finding $\mathbf{T_i}$ so that it satisfies

$$\min_{\mathbf{T}_i} \left(||\mathbf{T}_i\mathbf{v}_i - \mathbf{v'}_i||^2 + \sum_{j \in \mathscr{N}_i} ||\mathbf{T}_i\mathbf{v}_j - \mathbf{v'}_j||^2 \right). \tag{4.7}$$

If $\mathbf{T}_i$ in expression (4.7) is allowed to be unconstrained the solution is a so called membrane solution, i.e. all geometry detail is lost while $E(\mathbf{V'})$ is minimized. Therefore T_i must be constrained in some reasonable way. In [SCOL$^+$04] it is suggested that T_i is restricted to rotations, isotropic scaling and translation. With these constraints it can be shown that T_i (for a general 3D case of Laplacian mesh editing) can be represented by

$$T_i = s_i e^{\mathbf{H}_i} = s_i(\gamma_1 I + \gamma_2 \mathbf{H}_i + \gamma_3 \mathbf{h}_i^T \mathbf{h}_i), \tag{4.8}$$

where $\mathbf{H} \in \mathbb{R}^3$ is a skew-symmetric matrix that satisfies $\mathbf{Hx} = \mathbf{h} \times \mathbf{x}$. This is a quadratic expression in terms of $\mathbf{h}$ which can be approximated by its linear components. A general $\mathbf{T}$ with these constraints expressed only by its linear components can be represented by

$$\mathbf{T} = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & -h_3 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.9}$$

where for a specific $\mathbf{T}_i$ the vector of unknowns is $\mathbf{w}_i = (s_i, \mathbf{h}_i, \mathbf{t}_i)^T$. The minimization problem then reduces to

$$\min ||\mathbf{A}_i\mathbf{w}_i - \mathbf{b}_i||^2, \tag{4.10}$$

where $\mathbf{A}_i$ contains the positions of $\mathbf{v}_i$ and has the structure

$$\mathbf{A}_i = \begin{pmatrix} v_{k_x} & 0 & v_{k_z} & -v_{k_y} & 1 & 0 & 0 \\ v_{k_y} & -v_{k_z} & 0 & v_{k_x} & 0 & 1 & 0 \\ v_{k_z} & v_{k_y} & -v_{k_x} & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \end{pmatrix}, \tag{4.11}$$

with $k \in \{i\} \cup \mathcal{N}_i$ and $\mathbf{b}_i$ contains the positions of $\mathbf{v'}_i$ and has the structure

$$\mathbf{b}_i = \begin{pmatrix} v'_{k_x} \\ v'_{k_y} \\ v'_{k_z} \\ \vdots \end{pmatrix}, \tag{4.12}$$

with $k \in \{i\} \cup \mathcal{N}_i$. The vector of unknowns is then acquired by the standard linear least squares method

$$\mathbf{w}_i = \left(\mathbf{A}_i^T \mathbf{A}_i\right)^{-1} \mathbf{A}_i^T \mathbf{b}_i, \tag{4.13}$$

yielding the new geometry $\mathbf{V}'$ for all vertices by solving the system for the anchor vertices.

Linearly approximating $\mathbf{T}$ does not exactly capture perfect rotations, isotropic scaling and translation and therefore defects in the new geometry might occur. In [SCOL+04] several additional methods for improving the result of the Laplacian mesh editing algorithm are presented. These methods are however not adopted here since slight deformation of the parametrizations due to the linearization is not considered a large problem compared to the distortion generated by the mapping itself, i.e. when deciding what point on the source patch maps to what point on the target patch a much larger distortion will occur than when reshaping the parametrizations.

Using this method iteratively (until the error between the transformed vertices and the desired positions of said vertices are sufficiently small) ensures that the anchor vertices actually are transformed onto the required positions.

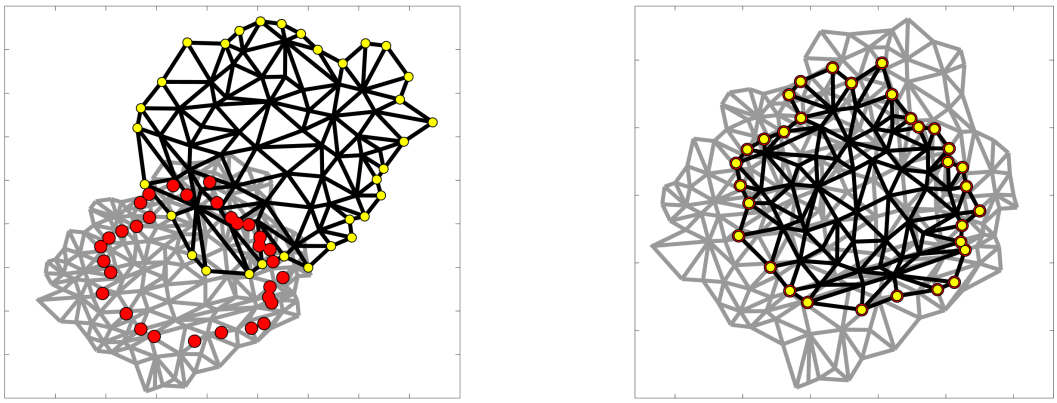
Aligning the parametrizations is now done by first defining the anchor vertices as the set of vertices in P that are found in the boundary of P. The desired positions for the parametrization coordinates in $\mathbb{R}^2$ for the vertices in P are the representation in the target parametrization of their closest points on the target patch in $\mathbb{R}^3$. The result of using the Laplacian deformation technique in order to achieve a proper alignment can be seen in Figure 4.1.

After the new geometry is applied to the source patch parametrization the parametrization pair is validated. The validation consists of following steps:

1. Check both patch parametrization triangulations for self intersections.
2. Check if the source patch parametrization triangulation is entirely located within the polygon created by the boundary of the target patch parametrization triangulation.
3. Check that the genus[2] of both the patches are equal to zero.
4. Find unnecessary parts of the target patch parametrization triangulation.

The check for self intersection is to ensure the that the bijective criterion is met. The second check is performed to make sure that there actually is a mapping for every point in the source patch. If some part of the source patch parametrization triangulation is located outside of the target patch parametrization triangulation that part has no mapping and the parametrization pair is invalid. The third check is not a requirement for the parametrization nor the mapping to

[2]The *genus* of a closed orientable manifold can be described as the number of cuts along closed curves on the manifold that can be done without rendering the resulting manifold disconnected. Open orientable manifolds are classified according to the number of boundaries they have along with the genus of their closed counterpart manifold. However a sloppy definition of genus for open oriantable manifolds can be thought of as the manifolds number of boundaries minus one.



(a) *Unaligned parametrizations, red dots indicate desired positions for the vertices marked with yellow.*

(b) *Aligned parametrizations.*

Figure 4.1: *The result of applying the Laplacian deformation technique to the source parametrization in order to achieve proper alignment. The black triangulation represent the parametrization of the source patch while the gray triangulation represent the parametrization of the target patch.*

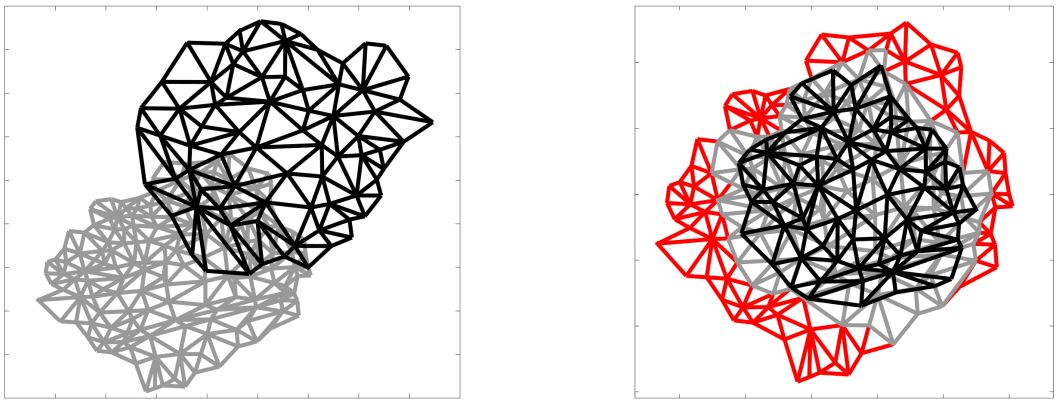
be valid *per se* but this thesis does not present a way of matching patches with any genus other than zero since patches of higher genus present a much more difficult matching problem in terms of analyzing the topology, orientation and geometry.

The fourth check, i.e. locating the unnecessary parts of the target patch parametrization triangulation, is not needed in order for the parametrization pair to be valid but is performed for two reasons. The first is that it is plain unnecessary to include the triangles of the target patch parametrization that are not needed in order to describe the mapping of all points in the source patch. The second reason is that a parametrization might have been invalidated because of some topological or geometrical defect in the part that is unnecessary. If that part is removed and the validation performed again the parametrization pair might be valid. Figure 4.2 shows an example of the difference between two parametrizations when they are unaligned versus when they are aligned with unnecessary triangles marked as red.

If the parametrization pair returns invalid from the parametrization alignment the unnecessary parts are removed and the target patch is geodesically expanded. The expansion is performed since the source patch parametrization might cover areas where there are no target patch parametrization triangles. This iterative patch expansion, unnecessary parts removal and validation is done some number of times after which the error for the parametrization pair is evaluated or the matching is canceled due to an invalid parametrization pair.

4.3 Error metric

In order to measure the quality of a match between two patches an error metric has to be introduced. There are several ways of measuring errors, a very computational and memory efficient method of estimating the error between two triangulated surfaces is presented in [ASCE02] and relies on the Hausdorff distance. However since this thesis focuses on the quality of result



(a) *Unaligned parametrizations.*

(b) *The aligned parametrizations of two matching patches with unnecessary triangles marked as red.*

Figure 4.2: *The difference between two parametrizations when they are unaligned versus when they are aligned. The black triangulation represent the parametrization of the source patch while the gray triangulation represent the parametrization of the target patch. Unnecessary triangles in the target parametrization are marked as red.*

rather than the computational gain in this step the suggested approach in [ASCE02] for the error estimate is not adopted here.

The desired error metric shall represent how similar two patches are in terms of their shape and must capture even small anomalies in the mesh surfaces. The error calculation step is not in any way the most computationally heavy part of this algorithm, hence the lack of interest for very computationally efficient approaches. Let

$$\{\mathbf{t_i}\}_j \subset \mathbb{R}^3 \quad i = 1, 2, 3 \quad j = 1, 2 \tag{4.14}$$

be two sets of vectors in $\mathbb{R}^3$ that for each j represents the vertices in two separate triangles, on the source and target patch respectively, for which the triangle vertices are mapped to each other.

Consequently the points within the triangles are mapped to each other as well, where they simultaneously can be described by the same set of barycentric coordinates. Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be two points within the triangles ($\mathbf{p_1}$ is within the first triangle and $\mathbf{p_2}$ is within the second triangle). They can be described with a set of barycentric coordinates $\lambda_1, \lambda_2, \lambda_3$ according to

$$\mathbf{p_j} = \lambda_1 \mathbf{t_{1,j}} + \lambda_2 \mathbf{t_{2,j}} + \lambda_3 \mathbf{t_{3,j}}, \quad j = 1, 2. \tag{4.15}$$

Due to the properties of barycentric coordinates [Ung10] this can be reduced to

$$\mathbf{p_j} = \lambda_1 \mathbf{t_{1,j}} + \lambda_2 \mathbf{t_{2,j}} + (1 - \lambda_1 - \lambda_2) \mathbf{t_{3,j}}, \quad j = 1, 2 \quad , \tag{4.16}$$

since $\lambda_3 = (1 - \lambda_1 - \lambda_2)$. The distance $D(\lambda_1, \lambda_2)$ between two mapped points can therefore be described as

$$D(\lambda_1, \lambda_2)^2 = |\mathbf{D}(\lambda_1, \lambda_2)|^2 = |\mathbf{p_1}(\lambda_1, \lambda_2) - \mathbf{p_2}(\lambda_1, \lambda_2)|^2. \tag{4.17}$$

A reasonable way to measure how well two sets of points match each other is to integrate the square distance, $D(\lambda_1, \lambda_2)^2$, between the sets over the surface of the matched triangle. This can be described by the integral

$$E_t = \int_{Triangle} D(\lambda_1, \lambda_2)^2 \, \mathrm{d}\lambda_1 \mathrm{d}\lambda_2 = 2A \int_0^1 \int_0^{1-\lambda_2} D(\lambda_1, \lambda_2)^2 \, \mathrm{d}\lambda_1 \mathrm{d}\lambda_2, \tag{4.18}$$

where A is the area of the triangle and E_t is the total accumulated error for the triangle match. The integral E_t can be simplified and described in terms of the vectors given by (4.14). Doing so yields the expression

$$E_t = \frac{A}{6} \left(\mathbf{a} \cdot \mathbf{a} + \mathbf{b} \cdot \mathbf{b} + \mathbf{c} \cdot \mathbf{c} + \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{c} \right), \tag{4.19}$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are given by

$$\begin{aligned} \mathbf{a} &= \mathbf{t_{1,1}} - \mathbf{t_{1,2}} \\ \mathbf{b} &= \mathbf{t_{2,1}} - \mathbf{t_{2,2}} \\ \mathbf{c} &= \mathbf{t_{3,1}} - \mathbf{t_{3,2}}. \end{aligned} \tag{4.20}$$

See Appendix 9.1 for full derivation of the expression for the integral. This metric for measuring the error for a set of two arbitrary triangles that are considered to match each other can be used in order to get the error for larger sets of matched triangles, i.e. the error for entire patches. However, when two arbitrary patches are considered to match each other they do not (in general) align their triangulations. Therefore the surface integration cannot be applied straight up to the patches. The patches must first be transformed into new triangulations for which every vertex on the source patch has a corresponding unique matching vertex on the target patch. The total accumulated error for a patch matching is then given by the sum of all errors for each individual triangle matching.

It is possible to create a new polygon mesh from the overlapping parametrizations by merging the parametrizations. This is done by defining the new merged parametrization as the polygon mesh created by putting a vertex at every intersection between edges in the union of the two parametrizations. Since the original meshes are constructed of only triangles there exists an analytic expression for the surface integral over every polygon in the new merged parametrization if utilizing generalized barycentric coordinates [DBMoT06]. The expression for the error metric integral, given by (4.19), is however only defined for triangles, not polygons of arbitrary number of corners. This problem is solved by simply triangulating every polygon so that it consists of only triangles, after which (4.19) can be applied to every triangle in the new merged parametrization. The error for a matching between patches is defined as the sum of the errors for each triangle match

$$E = \sum_{t=1}^{N} E_t, \tag{4.21}$$

where E_t denotes the error of triangle t in the merged patch, evaluated with respect to (4.19), with a total number of N triangles.

However, since E is the error metric with respect to the integral over either the source patch area or the target patch area just integrating over one or the other might result in a misleading error evaluation for surfaces with very different total areas. Hence it is preferable to consider the

error metrics with respect to the source patch and the target patch as orthogonal metrics and let the total direction invariant (i.e. it does not matter if the patch p_1 is considered to match onto patch p_2 or vice versa) error, E_{tot}, for a patch matching be

$$E_{tot} = \sqrt{E_{source}^2 + E_{target}^2}, \tag{4.22}$$

where E_{source} and E_{target} are the error metric described in (4.21) integrated with respect to triangles in the source and target patch respectively.

Perhaps more interesting is the normalized error measure, $\tilde{E}_{tot}$, that represents the error per area of the matching. $\tilde{E}_{tot}$ is given by normalizing over the sum of the source and target patch areas, A_{source} and A_{target} respectively, yielding the expression

$$\tilde{E}_{tot} = \frac{E_{tot}}{A_{source} + A_{target}}. \tag{4.23}$$

Chapter 5

Global bijective matching

In Section 4.1 to 4.3 the algorithm described for matching two patches is non-adaptive, i.e. a matching of a patch has no concept of already matched patches . That means the matching does not know what part of the meshes a mapping of points are allowed to access in order to ensure that the mapping of all points within all matched patches are globally bijective.

An outline of the structure for the entire adaptive algorithm from raw input data to a bijective set of mapped patches in between the two meshes is shown in Algorithm 4. The algorithm describes the general idea for how to ensure that the mesh matching is globally bijective. Essentially this is an optimization problem for which there is no easy way of finding a global optimum. The quality of the result strongly depends in what order the matching of patches is performed. The algorithm presented in Algorithm 4 does not claim to find the global optimum but rather suggests a way of finding a valid solution to the problem.

Algorithm 4 *Outline of the algorithm for finding and creating a globally bijective set of matching patches from one mesh to another.*

1: Input: Raw data of two meshes.
2: **for** $i \in [1, 2]$ **do**
3: Preprocess data for mesh i.
4: Create voxelization for mesh i.
5: Calculate the Gabor descriptors for all voxels in the shell of voxelization i.
6: **end for**
7: Extract seed points.
8: Calculate the non-adaptive mappings for all seed points.
9: Sort the valid mappings according to the error metric, throw away the invalid ones.
10: Adaptively add and connect patches starting with the one with lowest error.
11: Adaptively add and connect additional patches, expanding from the boundaries of the already mapped domain, a number of times with reducing patch size.
12: Return: The set of matched patches for the two meshes.

Basically the only thing that differs from the way to extract and match a patch described in Section 4.1 to 4.3 is that there are certain constraints on what part of the mesh the extraction of the patch are allowed to access. This criterion is conceptually easy to satisfy by just removing triangles from the meshes as the matching progresses in order to ensure that a bijective set of patches is created. While such a solution would render a valid, globally bijective set of matching patches none of them would be connected to each other. The ability to connect patches to each

other is a must in order to cover as great part of the meshes as possible.

5.1 Connecting patches

Presume that there exists a bijective set of already matched patches in between a source mesh and a target mesh. When adding a new patch, P_s, extracted from the source mesh to that set the bijectivity must be preserved. Let's say P_s shares some boundary vertices (it cannot share any vertices other than boundary vertices since every triangle within the already matched patches is excluded from the domain from which the extraction of the new patch occurs) with an already existing patch, Q_s, on the source mesh. The adding of the new patch should ensure that all points along the curve in $\mathbb{R}^3$ spanned by that shared boundary segment on the source mesh maps to the exact same curve in $\mathbb{R}^3$ on the target mesh, no matter from which patch parametrization the mapping of the points along the curve is done with respect to. The parametrizations of the already present patch and the new addition must therefore be synchronized. At first glance it might seem like this problem is resolved by just ensuring that the shared vertices on the source mesh are mapped to the exact same location on the target mesh for both patch mappings, this is however not true in most cases.

Let $C_s^3 \in \mathbb{R}^3$ be the curve on the source mesh that is spanned by the shared vertices among patch P_s and Q_s and let $\mathscr{P}_A(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^2$ be the homeomorphism that transforms a set of points $\mathbf{x} \in \mathbb{R}^3$ to new points $\mathbf{y} \in \mathbb{R}^2$ according to the paramatrization for a patch A. It then follows that

$$\begin{aligned} \mathscr{P}_{P_s}(C_s^3) &= C_{s,P_s}^2 \in \mathbb{R}^2 \\ \mathscr{P}_{Q_s}(C_s^3) &= C_{s,Q_s}^2 \in \mathbb{R}^2 \end{aligned} \tag{5.1}$$

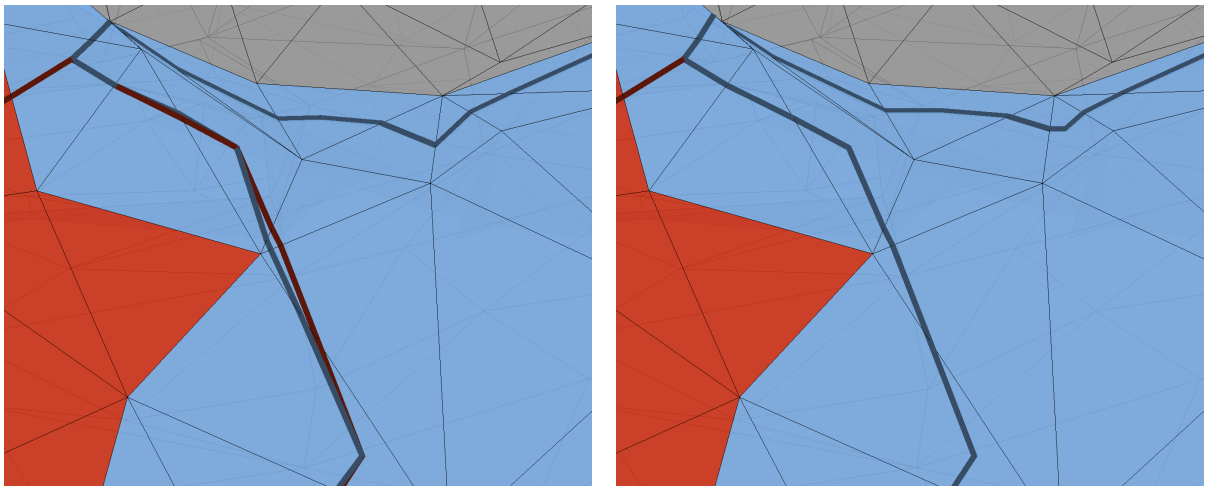
where in the general case $C_{s,P}^2 \neq C_{s,Q}^2$ since the parametrizations for patch P_s and Q_s typically are not the same. Even though $C_{s,P}^2 \neq C_{s,Q}^2$ it is still possible to ensure that

$$\begin{aligned} \mathscr{P}_{P_t}^{-1}(C_{s,P_s}^2) &= C_{t,P}^3 \in \mathbb{R}^3 \\ \mathscr{P}_{Q_t}^{-1}(C_{s,Q_s}^2) &= C_{t,Q}^3 \in \mathbb{R}^3 \\ C_{t,P}^3 &= C_{t,Q}^3, \end{aligned} \tag{5.2}$$

is fulfilled with P_t and Q_t representing the corresponding patches to P_s and Q_s on the target mesh and $\mathscr{P}_A^{-1}(\mathbf{y}) : \mathbb{R}^2 \to \mathbb{R}^3$ being the inverse homemorphism to $\mathscr{P}_A(\mathbf{x})$ that transforms a set of points $\mathbf{y} \in \mathbb{R}^2$ to new points $\mathbf{x} \in \mathbb{R}^3$ according to the paramatrization for patch A. The way $C_{t,P}^3 = C_{t,Q}^3$ is achieved is by changing the vertices and triangles of the parametrization for P_s and P_t, respectively, so that the curve C_{s,P_s}^2 intersects the triangles in the parametrization for patch P_t in the exact same way as the curve C_{s,Q_s}^2 intersects the triangles in the parametrization for patch Q_t.

Let V_{P_s} define the set of vertices in the parametrization of P_s that must be changed in order to acquire the right shape of C_{s,P_s}^2 and let V_{P_t} define the set of vertices in the parametrization of P_t that must be changed in order for C_{s,P_s}^2 to intersect the triangles at correct coordinates. Furthermore, let V_{Q_s} and V_{Q_t} be the corresponding sets of vertices in the parametrizations of the already existing patch pair Q_s and Q_t that describe the shape of the curve C_{s,Q_s}^2 and the vertices in the triangles the curve intersects in the parametrization of Q_t. Deforming the parametrization

for P_s and P_t into the correct shape is then preferably done with the Laplacian mesh editing technique (described in Section 4.2), where the anchor vertices are the vertices in V_{P_s} and V_{P_t} and the desired anchor positions are the positions of the vertices in the set V_{Q_s} and V_{Q_t}, respectively. The result of this and the difference when the technique adopted versus when it is not can be seen in Figure 5.1.



(a) *The case when the parametrization of the patches are not corrected with the technique described in Section 5.1. As clearly visible the matching is not bijective since the shared boundary segments between the shared vertices does not take the same path on the target mesh for both patches.*

(b) *The case when the parametrization of the patches are corrected with the technique described in Section 5.1. Here the mapping of points is completely bijective.*

Figure 5.1: *An illustration of the difference between when the patches parametrizations are not synchronized in order to have the same shape of their boundary on the target mesh versus when they are. The blue patch represents P_t and the red patch represents Q_t. Do not be misguided by the fact that the color of the triangulations overlap since this is the result of the patches having access to the shared triangles along the shared boundary segment. The thing that matters is the shared boundary curve and that it is the same for both patches after synchronization.*

Interestingly enough the situation does not become significantly more complicated if the new patch P_t shares boundary segments with more than one patch. The idea does not change that dramatically since the patches must be added one by one and for every additional patch the correction is made. Hence if by adding a new patch P_t makes a triangle be shared by three patches the triangle that now is shared among three patches was previously shared by two patches. When those two patches were connected to each other they synchronized their parametrizations so that the shape and intersections of that triangle are identical no matter from which parametrization they are extracted. Therefor the order of correcting boundary segments when correcting the parametrization of the third patch does not matter nor does the correction of shared boundary segment with the third and the first patch affect the correction of the shared boundary segment with the third and the second patch.

There is yet another situation which can occur, namely when a new patch is added and it shares several boundary segment with already existing patches that are not connected with each other. Since the parametrizations of the already existing patches need not be synchronized

there is nothing that guarantees that when adopting the correct parts of each parametrization the entire parametrization of the new patch will not break. The parametrization will most likely break, rendering the matching of that patch invalid, since the parametrization of each individual, non synchronized patch can have an arbitrary set of coordinates. This means that adopting the coordinates straight up when adding the new patch would most likely result in the parametrization of the new patch having self intersections or being heavily distorted.

This problem is solved by allowing the coordinates of the parametrization of the new patch to be freely translated, rotated and scaled to best fit the parametrizations of the already existing patches, as long as the translation, rotation and scaling is identically applied to all vertices in the parametrization of both P_s and P_t. Allowing translation, rotation and scaling if applied to the parametrization of both P_s and P_t does not affect $C^3_{t,P}$ since the mapping of points by $\mathscr{P}_A(\mathbf{x})$ is invariant to those kinds of transformations when applied simultaneously to both $\mathbf{x}$ and the parametrization of A itself. A schematic view of how this works can be seen in Algorithm 5.

Algorithm 5 *Outline of the algorithm for synchronizing a new patch, P_n, with an already existing set of patches.*

Input: Patch info for new patch P_n and the set of already existing patches $\{P_i\}$, $i \in [0, n-1]$.
for $i \in [0, n-1]$ **do**
    if $P_n \cap P_i \neq \emptyset$ **then**
        Calculate the translation, rotation and scaling between P_i and P_n.
        Apply translation, rotation and scaling to P_n.
        Use Laplacian deformation to adjust P_n so that the coordinates of the vertices in $P_n \cap P_i$ are identical with the coordinates of corresponding vertices in P_i.
    end if
end for
Return: Adjusted version of P_n.

This technique of synchronizing patches does not guarantee that the parametrization pair of the new patch will be valid since it can still break for various other reasons, e.g. manifold defects etc.

5.2 Potential compression ratio

When a set of globally bijective matched patches are acquired for a target and a source mesh the potential compression ratio in between their corresponding frames in the animation is somewhat proportional to the cumulative area of all the matched patches compared to the total area of the meshes, as well as the number of vertices in the matched region compared to the total amount of vertices.

However, the exact relationship is not so trivial to derive since neither the area nor the number of vertices in the matched region exactly describe what the resulting mesh after the remeshed sequence of frames is generated will look like. Let's say a large area on the target mesh can be replaced by an equally large area sprouted from the source mesh but that is resembled by a much lower number of vertices, while still describing the correct shape with a small enough error. The compression ratio in between the frames corresponding to those meshes will come from both the fact that a part of the first frame is used in the later frame as well as that the number of vertices is reduced. Hence, deriving an analytic expression for the compression ratio for a sequence of frames as a function of the matched patches in between the meshes is quite

non trivial since the compression ratio, apart from above mentioned facts, also depend on the particular algorithm used for remeshing the sequence. Despite this the matched area is still an interesting property that gives a hint of the level of compression one can expect.

Another interesting property of a series of mesh matchings constructed of consecutive meshes in an animation is how far an initially matched patch within the first mesh matching is persistent in the sequence. The nature of this property can be explained further by an example. Let's say there exists a matched patch between the first and second mesh, then that patch has a frame sequence length of at least one. However, if the target patch of that matching has some part of it that maps to a patch on the third mesh the patch has a frame sequence length of at least two. By analyzing the entire sequence in the same way it is possible to determine the longest frame sequence, or the *patch persistence*, for an initially matched patch.

This entity is worth measuring since it is desirable to have as long lived patches as possible when guiding the remeshing. A long lived patch means that the triangles within that patch can potentially be reused quite a lot, yielding an over all higher compression ratio throughout the sequence.

Chapter 6

Results

In this section various results concerning the quality of the matchings as well as properties that are important for the potential compression ratio are presented. For the voxelization the resolutions used are $\lambda_{fine} = 1/256$ and $\lambda = 1/16$ where λ_{fine} is considered to capture the finest structures of the meshes. For the calculation of the Gabor descriptors a local neighborhood size of $\beta = 11$, in each cartesian dimension, is used for information extraction. A larger β would give higher precision but the computational time for calculating the descriptors drastically increases with β. The radial maximum tensor rank k_{max} is in the calculations set to $k_{max} = 5$ and since $\beta = 11$ the maximum angular tensor rank is $l_{max} = 5$. Furthermore the width parameter, σ, of the gaussian part of the Gabor descriptor basis is set to $\sigma = \text{round}(\beta/1.5) = 7$ (empirically found to generate good compromise between how important local features are compared to features that are farther away).

For the seed point extraction the minimum allowed absolute value of the gaussian curvature K_{min} is set to $K_{min} = 1$. The local region of each voxelization in which the comparison of Gabor descriptors for seed point extraction is done is set to 1/4 of the entire voxelization size in each cartesian dimension.

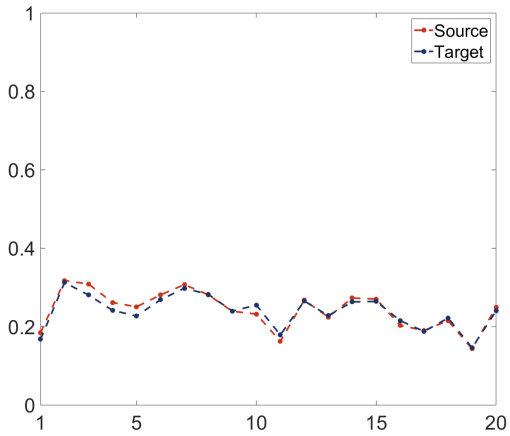
6.1 Area covered

As mentioned in Section 5.2 a hint of the compression ratio in between two frames can be given by analyzing the area of the matched area for that frame pair. Similarly can the compression ratio for a sequence of frames be given an indicative value by analyzing the cumulative matched area for a sequence compared to the cumulative area of the entire meshes for that sequence.

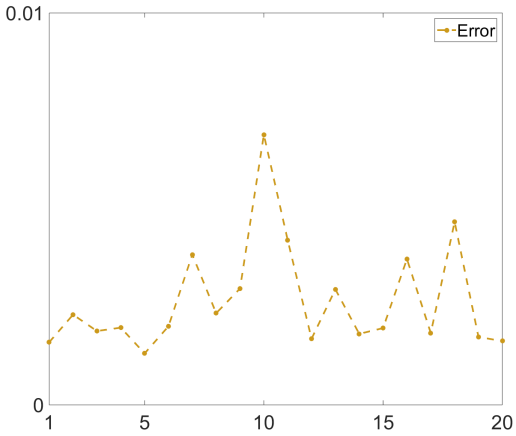
In a sequence of 21 frames, i.e. 20 subsequent mesh pairs, the average covered areas for each frame pair is roughly 25%. The exact result for all frame pairs is shown in Table 6.1 as well as in Figure 6.1. Note that the area for the matched set on the target mesh follows the curve of the area for the matched set on the source mesh quite well. This observation is also indicative of the quality of the matching, with low spread between the curves indicating good quality.

6.2 Patch persistence

Measuring the patch persistence as described in Section 5.2 of a frame sequence of 21 frames yielded a patch persistence of 20, i.e. some part of the patches that are present in the first frame are successively matched in such a way that their matching patches are present in the very



(a) Amount of area covered for matched patch sets in a sequence of 21 frames. The x-axis represent the frame pairs, numbered from 1 to 20, and the y-axis represents the amount of area covered by the matched patches where 0 represents no area and 1 represents complete coverage.



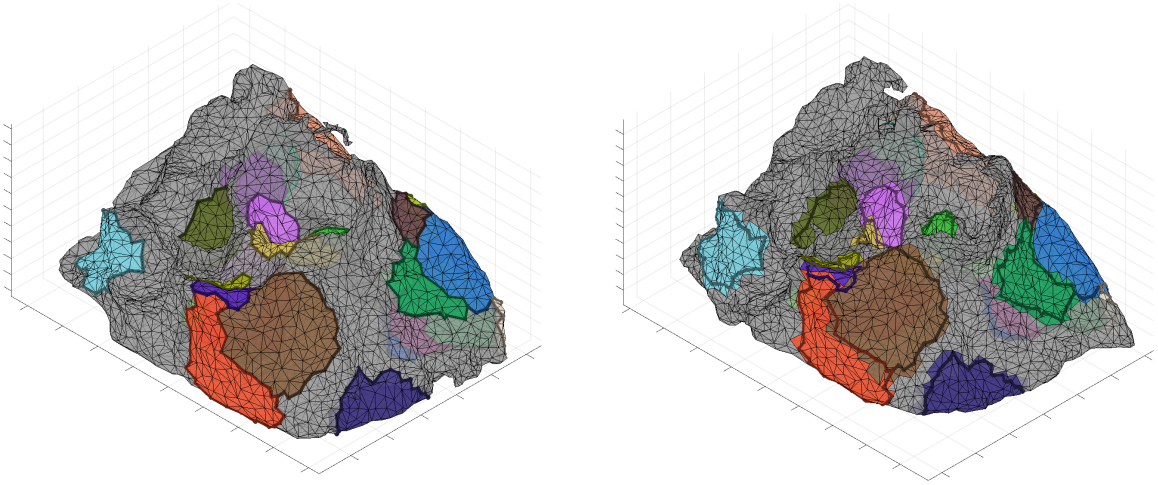
(b) Error for matched patch sets in a sequence of 21 frames. The x-axis represent the frame pairs, numbered from 1 to 20, and the y-axis represents the error.

Figure 6.1: *Covered area and error for matched patch sets in a sequence of 21 frames.*

last frame of the sequence. This measurement can be somewhat inaccurate for arbitrary mesh sequences since it highly depends on the quality of the first matching. If the first matching has a low of amount of matched patches, due to non similar meshes of other unforeseen obstacles while matching, it greatly affects the over all persistence for any patch if measured from that initial mesh matching. However, according to Table 6.1 the first matching is within the limits of what would be considered okay, even though the matching is slightly worse than the majority of the other matchings in terms of matched area, so the patch persistence measurement ought to be accurate.

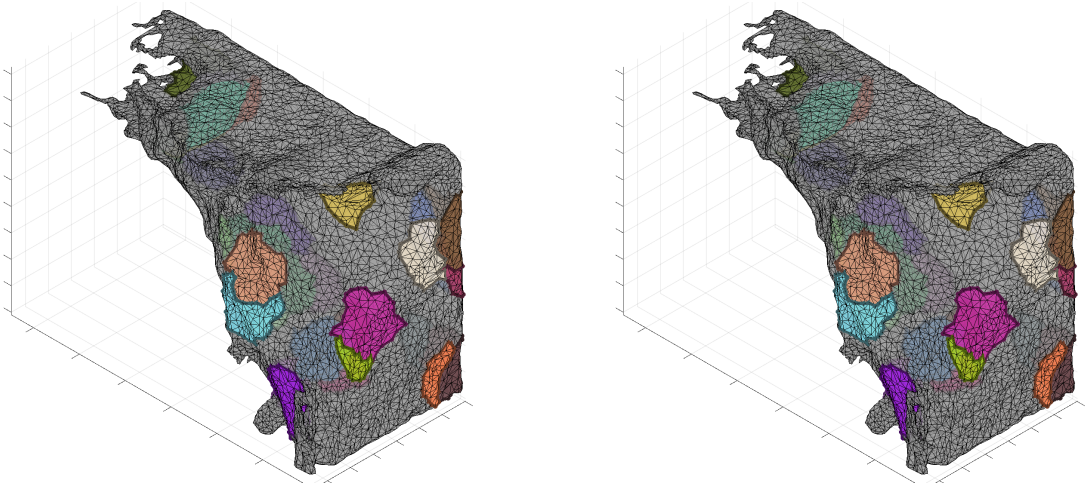
6.3 Different mesh shapes

The algorithm developed shows reasonably good performance for meshes of certain kinds. For meshes that have a structure consisting of somewhat unique features evenly scattered all over the mesh the amount of matched area is larger than for meshes with large chunks of non unique features. Figure 6.2 and 6.3 shows two mesh pairs with different structure in terms of how unique the meshes are on a local scale. The meshes in Figure 6.2 have a somewhat unique structure on a local scale as opposed to the large flat parts of the meshes in figure Figure 6.3 which are not that unique. The amount of covered area in Figure 6.3 is siqnificantly lower on the large flat parts of the mesh than on parts with higher amount of unique local structures.



(a) *Matched patches on the source mesh.* ***(b)*** *Matched patches on the target mesh.*

Figure 6.2: *The matched patches on a mesh pair. Do not be misguided by the fact that the color of the triangulations overlap on the target mesh since this is the result of the patches having access to the shared triangles along the shared boundary segment. The things that matter are the shared boundary curves which define the patch.*



(a) *Matched patches on the source mesh.* ***(b)*** *Matched patches on the target mesh.*

Figure 6.3: *The matched patches on a mesh pair of different character than the ones shown in Figure 6.2. Do not be misguided by the fact that the color of the triangulations overlap on the target mesh since this is the result of the patches having access to the shared triangles along the shared boundary segment. The things that matter are the shared boundary curves which define the patch.*

Table 6.1: *The amount of matched area for a sequence of consecutive frame pairs as well as the global error per area for the entire matching.*

Frames	Area source [%]	Area target [%]	Error [10^{-3}]
0 & 1	18.52	16.83	1.60
1 & 2	31.76	31.28	2.30
2 & 3	30.91	28.09	1.88
3 & 4	26.17	24.18	1.97
4 & 5	25.08	22.73	1.31
5 & 6	28.10	26.87	2.01
6 & 7	30.75	29.82	3.84
7 & 8	28.11	28.24	2.34
8 & 9	23.94	23.99	2.97
9 & 10	23.23	25.52	6.89
10 & 11	16.28	17.89	4.20
11 & 12	26.76	26.56	1.69
12 & 13	22.48	22.82	2.95
13 & 14	27.31	26.34	1.81
14 & 15	27.08	26.46	1.96
15 & 16	20.34	21.55	3.72
16 & 17	19.03	18.79	1.83
17 & 18	21.56	22.23	4.67
18 & 19	14.37	14.61	1.73
19 & 20	24.92	24.02	1.63
Mean	24.34	23.94	2.67

Chapter 7

Discussion

Results show that the proposed algorithm produces a matching set of patches in between two meshes well enough to cover roughly 25 % of the areas spanned by the meshes while fulfilling the constraints set for the matching. The error of a matching was successfully measured according to a direction invariant error metric that captures area distortion while still lets the triangulation structures for the patches be completely non similar. Furthermore the algorithm showed consistently sized errors for the matched sets within a sequence of meshes. This is good since it shows that this way of matching does not produce matches of radically different patches.

Another desirable property of the algorithm is that it produces matchings in a consistent way in terms of patch persistence through a sequence of patches. The patch life length reached numbers well above single digits which is good for the guidance of the remeshing. A high patch persistence may result in greater compression since parts of the meshes can be more frequently reused. Achieving a high patch persistence by using this algorithm does however not ensure that the same is true after the remeshing of all frames. The reason for this is that the process of successively remeshing the first mesh pair, then the second, then the third etc. might alter the meshes in such a way that the exact same patches as produced by this algorithm no longer exist. Still, a long patch life is better than a short one since it shows consistency of the matching algorithm itself.

7.1 Improvements

While the algorithm presented is successful in achieving a bijective set of matched patches between a pair of meshes there is still quite a lot of room for improvement. Many of the methods used are only scratching the surface of the ideas and techniques that could improve the algorithm. As the algorithm is constructed right now it does not perform any global optimization in terms of matching error and quality. Several of the steps described in Chapter 3 to 5 could be extended in terms of error optimization, and some ideas for improvements are presented here.

Better seed points extraction

Seed points extraction in the voxelizations is an important part of the algorithm since it basically constructs the foundation for the actual mesh matching. Currently the seed points are individually constructed, i.e. a seed point pair for a set of voxelizations is not aware of any other seed points. Seed points could be created in a manner where each seed point is extracted by relating the positions of the seed points to each other. This would produce a more consistent

spread of the seed points as well as increasing the overall coverage of the voxelizations. Having a seed point structure derived from distances to other seed points could perhaps also create better circumstances for connecting patches in the global bijective matching. Since all patches sprout from the seed points the global bijective matching could use that information to better determine the appropriate patch size as well as the patch position, achieving a higher accuracy for the matching and also possibly increasing the amount of covered area by the matched patches.

ICP modification

The algorithm does unfortunately in the case of meshes with large flat surfaces not produce as high degree of matching as hoped. This phenomenon of being unable to expand the matched part onto flat surfaces is thought to depend on the nature of the ICP matching in combination with the criterion that specific vertices should match to certain points. Assume there is a set of already matched patches and another patch that is almost completely flat is added to that set. The vertices on the boundary of the new patch that are shared with already existing patches should match in the exact same way (mechanic described in Section 5.1) while the rest of the boundary vertices should match to their closest point on the target patch after the ICP matching method is applied. However, the ICP matching has no concept of the vertices shared with already existing patches and does not know that these should be relocated with a weighting factor proportional to the positions of the already matched vertices. The ICP matching simply positions the extracted patch as good as possible from its local criteria, which for two entirely flat patches can result in a repositioning heavily rotated and translated from the original position. If the patches are heavily rotated from their original positions the risk of self intersecting parametrizations after alignment according to the positioning criteria for the boundary vertices is high ans can in many cases render the set of parametrizations invalid.

A solution to this problem could be to both weight the extraction of the closest points as well as the local error metric within the ICP algorithm while still preserving the global structure, i.e. adding a soft constraint for the placement in $\mathbb{R}^3$ in terms of the adjacent patches.

Optimize parametrization

The exact method for creating the parametrizations is not at all discussed or emphasized in this thesis. As mentioned there are several ways of creating a parametrization and a lot of different scenarios where some methods are preferable over others. Instead of using already existing toolboxes for creating the parametrization for a patch a method that can parametrize two patches simultaneously could be developed. Such a method could measure the error continuously while creating the parametrizations and try to minimize it. This would both increase the accuracy for the matching as well as render the initial parametrization alignment step redundant. A more advanced parametrization technique could in other words encapsulate all steps from the parametrization itself to the error evaluation along with increasing the quality of the matching.

Dynamic patch size

The global bijective matching described in Algorithm 4 successively reduces the patch size for the current matching, i.e. it first tries to add patches of large size a number of times and then reduces the patch size. This works quite well in terms of trying to cover large areas with bigger patches until that is not longer possible and then switching to smaller patch sizes. However, a completely dynamic patch size that is optimized with respect to the error metric would probably

solve this problem better. The risk of moving into a local minimum in terms of global error is smaller if the matching of each patch tries to minimize the local error for that patch rather than just accepting the error as it is. Implementing such a feature could perhaps be done with the use of conventional optimization methods but would at the same time significantly increase the computational complexity of the algorithm.

Higher patch genus

As of now the algorithm only supports matching of patches resembling open manifolds of genus zero. These are the simplest and most intuitive patches to match but support for patches of higher genus could increase the matched area even further. Supporting matching of open or closed patches of any genus presents a much more difficult problem in terms of aligning parametrizations and ensuring bijective matching. For example, an open patch of genus 2 could match onto any open patch of genus 2 or higher while the opposite scenario is not true. Many things can go wrong when trying to correctly align and connect patches of different genus to an already existing set of patches. While this is in no way impossible, it is definitely a hard problem.

Different resolution matching

An idea for globally optimizing the mesh matching is to successively do matchings of the Gabor descriptors for different voxelization resolutions. If the Gabor descriptors for some regions match each other at a low resolution voxelization the process is repeated for those regions but the resolution is increased. In this way the local neighborhood used for calculating the descriptors could be successively increased along with the resolution, making it possible to capture even finer structures in the voxelization without needing to compare the descriptor with the entire voxelization since some regions have been excluded at lower resolution matchings.

This idea could work but it presents some problems in terms of fine tuning the parameters that determine the rate of change for the resolution and local neighborhood size.

Non bijective approach

The idea of having a set of matched patches that bijectively maps one point to another could be revoked and replaced by a so called multivalued mapping, i.e. an element in the source can map to multiple elements in the target but no element in the target has multiple sources. Figure 7.1 shows a schematic view of the difference between a bijective mapping and a mapping based on a multivalued function.

Approaching the matching in this way may increase compression ratio even more but it complicates things in terms of how to handle adjacent patches to those patches in the source that map to multiple targets.

Parallelism

The implementation of the algorithm presented in this thesis has not been paid much attention. It was not the goal to create a high performance implementation at this point in the development, however the potential for performance optimization in terms of parallel implementation is quite good. When calculating matching sets of patches for an entire sequence the parallelism is obvious in terms of splitting the calculations at each frame pair, the hard part to figure out when doing this is how to split the sequence in order to get good load balancing for sequences where the meshes drastically change sizes during the sequence.

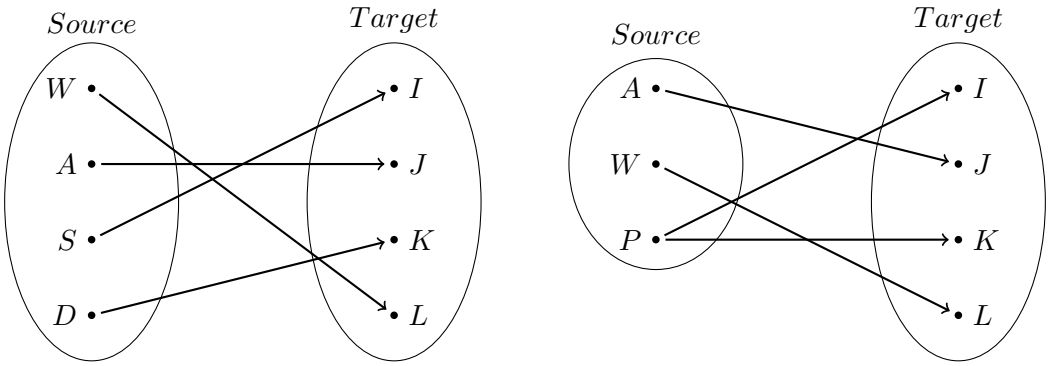


Figure 7.1: *Two different kinds of mappings. On the left side is a bijective mapping and on the right side is a non bijective multivalued function mapping. The letters symbolize patches and the arrows does essentially symbolize homeomorphisms. A letter together with an arrow can in some sense be interpreted as a chart.*

Large chunks of the algorithm described in Chapter 3 are also highly parallelizable. The calculation of Gabor descriptors for each voxel could potentially be implemented in a GPU environment since there is basically no branching or communication in between different data structures for those calculations. Another example is the comparing and matching of Gabor descriptors which could also be implemented in parallel.

When it comes to the adaptive adding of patches described in Chapter 5 it is a bit more cumbersome to implement the algorithm in a parallel way. The reason for that is that the addition of a new patch is completely dependent on the existing patches. However a new patch is only dependent on the patches it will be connected to so the problem could be partially solved by analyzing what patches are connected to certain regions and try to match patches of different regions, i.e. patches that certainly will not connect to the same parts of the mesh, in parallel.

Chapter 8

Conclusion

This thesis describes a method for producing a set of matching mesh patches between two meshes while ensuring bijective mapping of all points within the patches. The constructed error metric captures anomalies in shape similarity well and gives a quantitative way of measuring the quality of a mesh matching.

The mesh matching is bijective and the error is measured but algorithm is in no way flawless and there is room for improvements. Things like dynamic error optimization and custom designed parametrization methods are not incorporated in this algorithm. Future work could improve the algorithm in those areas and also pay more attention to computational complexity. Despite this the conclusion is that the goal for this thesis is achieved and the project can be considered a success.

Chapter 9

Appendix

9.1 Error integral derivation

Let $\mathbf{a}, \mathbf{b}, \mathbf{c}$ be defined according to (4.20). The integral E_t from (4.18) can then, with $D(\lambda_1, \lambda_2)^2$ equal to

$$\begin{aligned} D(\lambda_1, \lambda_2)^2 &= (\mathbf{a}\lambda_1 + \mathbf{b}\lambda_2 + \mathbf{c}(1 - \lambda_1 - \lambda_2))^2 \\ &\Longrightarrow \\ D(\lambda_1, \lambda_2)^2 &= \mathbf{a} \cdot \mathbf{a}\lambda_1^2 + \mathbf{b} \cdot \mathbf{b}\lambda_2^2 + 2\mathbf{a} \cdot \mathbf{b}\lambda_1\lambda_2 + 2\mathbf{a} \cdot \mathbf{c}\lambda_1(1 - \lambda_1 - \lambda_2) + \\ &\quad 2\mathbf{b} \cdot \mathbf{c}\lambda_2(1 - \lambda_1 - \lambda_2) + \mathbf{c} \cdot \mathbf{c}(1 - \lambda_1 - \lambda_2)^2 \end{aligned} \tag{9.1}$$

and with $\gamma_i,\ i = 1, 2, 3, 4, 5, 6$ defined as

$$\begin{aligned} \gamma_1 &= \mathbf{a} \cdot \mathbf{a} - 2\mathbf{a} \cdot \mathbf{c} + \mathbf{c} \cdot \mathbf{c} \\ \gamma_2 &= \mathbf{b} \cdot \mathbf{b} - 2\mathbf{b} \cdot \mathbf{c} + \mathbf{c} \cdot \mathbf{c} \\ \gamma_3 &= 2\mathbf{a} \cdot \mathbf{b} - 2\mathbf{a} \cdot \mathbf{c} - 2\mathbf{b} \cdot \mathbf{c} + 2\mathbf{c} \cdot \mathbf{c} \\ \gamma_4 &= 2\mathbf{a} \cdot \mathbf{c} - 2\mathbf{c} \cdot \mathbf{c} \\ \gamma_5 &= 2\mathbf{b} \cdot \mathbf{c} - 2\mathbf{c} \cdot \mathbf{c} \\ \gamma_6 &= 2\mathbf{c} \cdot \mathbf{c}, \end{aligned} \tag{9.2}$$

be rewritten as follows

$$
\begin{aligned}
& E_t = 2A \int_0^1 \int_0^{1-\lambda_2} \Big[\gamma_1\lambda_1^2 + \gamma_2\lambda_2^2 + \gamma_3\lambda_1\lambda_2 + \gamma_4\lambda_1 + \gamma_5\lambda_2 + \gamma_6\Big]\, \mathrm{d}\lambda_1\mathrm{d}\lambda_2 \\
\Longrightarrow \\
& E_t = 2A \int_0^1 \Big[\gamma_1\frac{\lambda_1^3}{3} + \gamma_2\lambda_2^2\lambda_1 + \gamma_3\frac{\lambda_1^2\lambda_2}{2} + \gamma_4\frac{\lambda_1^2}{2} + \gamma_5\lambda_1\lambda_2 + \gamma_6\lambda_1\Big]_0^{1-\lambda_2}\, \mathrm{d}\lambda_2 \\
\Longrightarrow \\
& E_t = 2A \int_0^1 \Big[\gamma_1\frac{(1-\lambda_2)^3}{3} + \gamma_2\lambda_2^2(1-\lambda_2) + \gamma_3\frac{(1-\lambda_2)^2\lambda_2}{2} + \\
& \quad \gamma_4\frac{(1-\lambda_2)^2}{2} + \gamma_5(1-\lambda_2)\lambda_2 + \gamma_6(1-\lambda_2)\Big]_0^{1-\lambda_2}\, \mathrm{d}\lambda_2 \\
\Longrightarrow \\
& E_t = 2A\Big(\frac{1}{12}\gamma_1 + \frac{1}{12}\gamma_2 + \frac{1}{24}\gamma_3 + \frac{1}{6}\gamma_4 + \frac{1}{6}\gamma_5 + \frac{1}{2}\gamma_6\Big),
\end{aligned}
\tag{9.3}
$$

which when expressed in terms of the triangle vectors looks like

$$
\begin{aligned}
& E_t = 2A\Big(\frac{1}{12}(\mathbf{a}\cdot\mathbf{a} - 2\mathbf{a}\cdot\mathbf{c} + \mathbf{c}\cdot\mathbf{c}) + \frac{1}{12}(\mathbf{b}\cdot\mathbf{b} - 2\mathbf{b}\cdot\mathbf{c} + \mathbf{c}\cdot\mathbf{c}) + \\
& \quad \frac{1}{24}(2\mathbf{a}\cdot\mathbf{b} - 2\mathbf{a}\cdot\mathbf{c} - 2\mathbf{b}\cdot\mathbf{c} + 2\mathbf{c}\cdot\mathbf{c}) + \frac{1}{6}(2\mathbf{a}\cdot\mathbf{c} - 2\mathbf{c}\cdot\mathbf{c}) + \frac{1}{6}(2\mathbf{b}\cdot\mathbf{c} - 2\mathbf{c}\cdot\mathbf{c}) + \frac{1}{2}(\mathbf{c}\cdot\mathbf{c})\Big) \\
\Longrightarrow \\
& E_t = \frac{A}{6}\Big(\mathbf{a}\cdot\mathbf{a} + \mathbf{b}\cdot\mathbf{b} + \mathbf{c}\cdot\mathbf{c} + \mathbf{a}\cdot\mathbf{b} + \mathbf{a}\cdot\mathbf{c} + \mathbf{b}\cdot\mathbf{c}\Big).
\end{aligned}
\tag{9.4}
$$

Bibliography

[AF09] Santiago Aja-Fernández. *Tensors in image processing and computer vision.* Springer, New York; London, 2009.

[ASCE02] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: measuring errors between surfaces using the hausdorff distance. volume 1, pages 705–708 vol.1, 2002.

[ATLF06] O. K. . Au, C. L. Tai, L. Liu, and H. Fu. Dual laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):386–395, 2006.

[BT82] 1923-2005 Bott, Raoul and Loring W. Tu. *Differential forms in algebraic topology*, volume 82. Springer, New York, 1982.

[CdGDS13] Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 courses*, SIGGRAPH '13, New York, NY, USA, 2013. ACM.

[CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[DBMoT06] Mathieu Desbrun, Alan H. Barr, Mark Meyer, and California Institute of Technology. Methods for computing barycentric coordinates generalized to irregular n-gons and applications of the same, 2006.

[DP13] Jonathan D. Denning and Fabio Pellacini. Meshgit: Diffing and merging meshes for polygonal modeling. *ACM Trans. Graph.*, 32(4):35:1–35:10, July 2013.

[EEG+08] David Eppstein, D. Eppstein, M. T. Goodrich, Michael T. Goodrich, E. Kim, Ethan Kim, Rasmus Tamstorf, and R. Tamstorf. Approximate topological matching of quadrilateral meshes. pages 83–92. IEEE, 2008.

[EGKT09] David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. Approximate topological matching of quad meshes. *The Visual Computer*, 25(8):771–783, 2009.

[GP74] 1937 Guillemin, Victor and Alan Pollack. *Differential topology.* Prentice-Hall, Englewood Cliffs, N.J, 1974.

[GS03] R. V. Garimella and B. K. Swartz. Curvature estimation for unstructured triangulations of surfaces. Technical Report LA-UR-03-8240, Los Alamos National Laboratory, November 2003.

[JSS+14] Yao Jin, Zeyun Shi, Jun Sun, Jin Huang, and Ruofeng Tong. Content-aware texture mapping. *Graphical Models*, 76(3):152 – 161, 2014. Computational Visual Media Conference 2013 Second Computational Visual Media Conference (CVM).

[KFR03] Misha Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Symposium on geometry processing. *Computers & Graphics*, 27(1):159 – 164, 2003.

[KMS12] Ullrich Köthe, Annick Montanvert, and Pierre Soille. Applications of discrete geometry and mathematical morphology: first international workshop, wadgmm 2010, istanbul, turkey, august 22, 2010 : revised selected papers. 7346., 2012.

[KW10] Hans Martin Kjer and Jakob Wilm. Evaluation of surface registration algorithms for pet motion correction. 2010. `http://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point`.

[LM98] Bruno Lévy and Jean-Laurent Mallet. Non-distorted texture mapping for sheared triangulated meshes. pages 343–352. ACM, 1998.

[LSS+11] Kun Liu, Henrik Skibbe, Thorsten Schmidt, Thomas Blein, Klaus Palme, and Olaf Ronneberger. 3d rotation-invariant description from tensor operation on spherical hog field. In *Proceedings of the British Machine Vision Conference*, pages 33.1–33.12. BMVA Press, 2011. `http://dx.doi.org/10.5244/C.25.33`.

[MKY01] F. Mokhtarian, N. Khalili, and P. Yuen. Curvature computation on free-form 3-d meshes at multiple scales. *Computer Vision and Image Understanding*, 83(2):118–139, 2001.

[MW97] 1931 Milnor, John W. and David W. Weaver. *Topology from the differentiable viewpoint*. Princeton Univ. Press, Chichester; Princeton, N.J, rev. edition, 1997.

[ONIT04] M. Okuda, K. Nagatomo, M. Ikehara, and S. Takahashi. Similarity detection of 3d meshes using 2d hierarchical regular grids. volume 1, pages 145–148 Vol.1, 2004.

[Pey09] Gabriel Peyre. *Toolbox Graph*. Gabriel Peyre, July 2009. `http://www.mathworks.com/matlabcentral/fileexchange/5355-toolbox-graph`.

[Ria08] Ricardo Riaza. *Differential-algebraic systems: analytical aspects and circuit applications*. World Scientific, New Jersey, 2008.

[Ros63] M. E. Rose. *Elementary theory of angular momentum*. New York, 1963.

[RRK+06] J. Rydell, Joakim Rydell, H. Knutsson, Hans Knutsson, M. Borga, and Magnus Borga. Rotational invariance in adaptive fmri data analysis. pages 2841–2844. IEEE, 2006.

[SCOGL02] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. pages 355–362. IEEE Computer Society, 2002.

[SCOL+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, 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. ACM Press, 2004.

[SHI96] Heung-Yeung Shum, M. Hebert, and K. Ikeuchi. On 3d shape similarity. pages 526–531, 1996.

[SR13] Henrik Skibbe and Marco Reisert. Rotation covariant image processing for biomedical applications. *Computational and mathematical methods in medicine*, 2013:931507, 2013.

[SRS+11] H. Skibbe, M. Reisert, T. Schmidt, T. Brox, O. Ronneberger, and H. Burkhardt. Fast rotation invariant 3d feature computation utilizing efficient local neighborhood operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1563–1575, 2012; 2011.

[Ung10] Abraham A. Ungar. *Barycentric Calculus in Euclidean and Hyperbolic Geometry: A Comparative Introduction*. World Scientific Publishing Company, Incorporated, Hackensack, 2010.

[Whi35] Hassler Whitney. Differentiable manifolds in euclidean space. *Proceedings of the National Academy of Sciences of the United States of America*, 21(7):462–464, 1935.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.