Simon Kloiber, BSc.

Creating Subdivision Meshes from Scan Data
A curvature adaptive surface reconstruction algorithm

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Ursula Augsdörfer, Ph.D M.Sc.

Institute of Computer Graphics and Knowledge Visualisation
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter W. Fellner

Graz, May 2018

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online:
https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

Subdivision surfaces, typically employed in the entertainment industry, are becoming more and more relevant in the development of products. Creating subdivision control meshes can either be done from scratch by a designer or can be derived from existing models through re-meshing. A curvature sensitive mesh density is desired to accurately model the features of a shape, while having a sparse set of control points. This thesis aims to construct a Catmull-Clark subdivision control mesh from a triangle reference mesh. Quad re-meshing is the most important step for this approach and a well-researched topic. The quad-dominant re-meshing method developed in Jakob et al., 2015 is modified so that it produces curvature sensitive quad meshes which are then transformed into subdivision control meshes.

Contents

List of Figures

1. Introduction

Many applications require 3-dimensional representations of smooth surfaces. Examples for these representations are 3D-models without underlying mathematical properties such as triangle or quad meshes, combinations of parametric surface patches, or surfaces based on iterative refinement, like subdivision surfaces. Subdivision surfaces are defined through a control polygon which is refined iteratively. Their limit surface is the resulting surface after infinite iterations of refinement. This limit surface is are able to approximate complex shapes with a coarse set of control points. There are many refinement schemes for subdivision surfaces with different mathematical properties. All future references to subdivision surfaces will refer to the Catmull-Clark subdivision scheme (Catmull and Clark, 1978) because it has become the de facto standard in the CAD and entertainment industry for quadrilateral meshes. The properties of subdivision schemes are:

Topology: Since subdivision surfaces are created through iterative refinement of an arbitrary mesh, they can be used to model surfaces of arbitrary topology. This sets them apart from parametric surfaces which are bound to a rectangular domain.

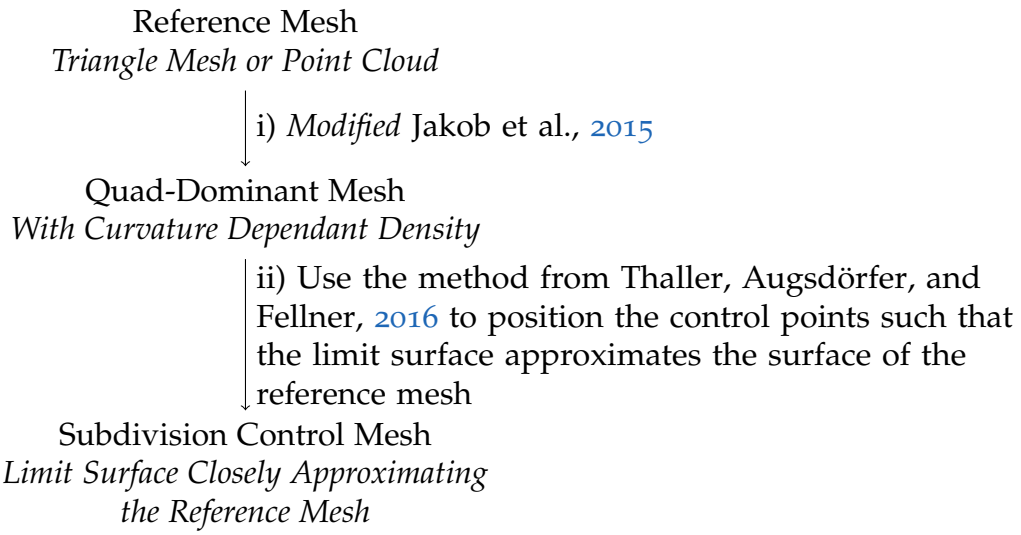
Scalability: Due to their iterative nature, their level of detail (LOD) can be scaled. Applications for this include distance based density when rendering or controlling the processing speed when performing calculations on the subdivision control mesh.

Numerical Qualities: The mathematical properties and positions of the limit surfaces of subdivision meshes are known and can be calculated without subdivision even in non-regular regions, as shown by Stam, 1998. Because the limit surface of Catmull-Clark subdivision control meshes is equal to uniform bi-cubic B-spline patches in regular regions, subdivision surfaces are curvature continuous everywhere except at extraordinary vertices (i.e. vertices which do not have a valence of four).

Ease of Use: Through above-mentioned points, subdivision surfaces are favourable compared to parametric representations in many applications. They are easy to implement and no stitching of boundaries is needed to satisfy continuity constraints in regular regions.

How to get the initial control meshes? Often, designers create real-life models of what they are trying to design. The models are then scanned and their shape is captured in the form of meshes. Because they closely capture the real-life models, these kinds of meshes will be called *reference meshes*. To continue the design with subdivision surfaces, they have to be transformed into subdivision control meshes.

Automatic conversion of a reference mesh to a quad-dominant Catmull-Clark subdivision control mesh is the goal of this thesis. As will be discussed in Chapter 2, a lot of research was done in the field of quad re-meshing. A new development in quad-dominant re-meshing by Jakob et al., 2015, creates quad-dominant meshes through quick local optimization from triangle or point cloud reference meshes. These meshes can then be transformed into subdivision control meshes with limit surfaces modelling the surfaces the scan data are describing. The method of Jakob et al. is based on creating quads of equal size. This is not favourable for designers trying to model objects with features because control meshes need to be dense enough to accurately represent the finest detail and creating meshes with quad faces of equal size may lead to unnecessary high overall densities. For this thesis, a pipeline is created that is able to transform reference meshes into subdivision control meshes with densities based on the curvature information of the referenced surface. The pipeline will contain the following steps:



i) A reference mesh is transformed to a quad-dominant mesh by the method developed in Jakob et al., 2015 with the changes outlined in this thesis, to ensure the density of the mesh is curvature-dependent.
ii) A subdivision control mesh is created from the quad-dominant mesh by moving its points along the normal of the reference meshes surface until their corresponding limit points closely approximate the surface.

The resulting subdivision control meshes can then be used for further modelling by designers to fine-tune their creations. To compare and evaluate the results of the pipeline, the following aspects are considered (adapted from Bommes, Zimmer, and Kobbelt, 2009):

Individual Element Quality: Quads of the mesh should have 90-degree corners, be planar and opposite edges should have same edge lengths

Orientation: Mesh edges should represent the principal curvature directions

Alignment: The mesh should represent the sharp features of the reference.

Global Structures: Careful positioning of as few extraordinary vertices as necessary is important.

Semantics: Some requirements on the resulting subdivision mesh depend on its later usage. On example for this is the ability to produce meshes suitable for design. These meshes must have few vertices while still accurately representing a given shape. One way to achieve this is adaptiveness of quad sizes to principal curvature.

For this thesis, the quality, orientation and alignment of quads are considered to create meshes suitable for design. Because the density of the control mesh adapts to the features, the global structure suffers far more extraordinary vertices when compared to meshes with an isotropic distribution. Consequently, the focus in this thesis does not lie on positioning the extraordinary vertices.

This thesis is structured into the following parts:
Chapter 2 summarises previous work on the topics of mesh reconstruction, re-meshing and mesh evaluation.
Chapter 3 explains the steps needed to create a quad-dominant mesh at a density adapted to the curvature of the surface described by the reference mesh. To find optimal methods for each step, different ones have been tried.
Chapter 4 performs a comparison of these methods, their combinations and the quality of the computed quad-dominant meshes and their corresponding subdivision control meshes.
Chapter 5 summarises the results of the work in this thesis and Chapter 6 gives an outlook for future work.

2. Related Work

The field of surface (re-)meshing is of interest to a great number of industries and their respective fields of research. Due to their natural alignment with the principal curvatures and patch structure, quad-based meshes are preferred for modelling tasks which focus on alignment and texture mapping (Bommes, Bruno, et al., 2012). Tensor product surfaces (e.g. B-splines, NURBS) and quad-based formulations of subdivision surfaces are all based on quadrangular meshes. Section 2.1 will discuss subdivision surfaces while Section 2.2 will give an overview of the methods for quad-based meshing.

2.1. Subdivision Surfaces

Subdivision surfaces describe surfaces derived by iteratively dividing a control mesh by following a subdivision scheme. The subdivision scheme used for all subdivision in this thesis is described by Catmull and Clark, 1978. They have created a subdivision scheme that will resolve to a bi-cubic patch in the limit surface for regular regions of the control mesh. This property is one of the most important features of Catmull-Clark subdivision surfaces. It means that in the regular regions of a mesh, the limit surface has curvature continuity. This property of subdivision surfaces allows the modelling of complex surfaces with great flexibility without having to worry about the mathematical properties of the limit surface in regular regions. In contrast, when modelling with parametric bi-cubic spline patches, care must be taken when stitching patches together to make sure the seams remain continuous (be it positional, directional or curvature continuity). Since extraordinary vertices locally disturb the regularity and slow down the convergence, careful placement of extraordinary vertices(EVs) is important.

The desire to model with subdivision surfaces demands for ways to construct them. The approach for the work in this thesis is by converting reference meshes into pure quad or quad-dominant meshes and then converting them into subdivision control meshes. While a quad mesh may define the topology of the control mesh, it does not yet define which positions the control vertices should have such that the limit surface lies on the reference mesh. To do this, the limit surface must be known. As mentioned before, the limit of regular regions is known per definition. On and around extraordinary vertices, this becomes more difficult.

Halstead, Kass, and DeRose, 1993 create a subdivision control mesh with limit points that interpolate the vertices of a reference mesh. They compute the position of the control points by creating a linear system which combines the limit point calculation with interpolation constraints. While the limit surface for regular vertices is known, Halstead et al. derive an expression for the limit positions of all vertices in the control mesh through an Eigen decomposition. The resulting surface, however, is not satisfactory and an additional fairing step is needed. They subdivide once, adding more degrees of freedom and improve the surface by minimising a smoothness norm based on a linear combination of thin-plate and membrane energies. Their evaluation of limit positions of control vertices is used in this thesis to create the subdivision control mesh. Their condition for limit points to interpolate their corresponding reference vertices is very restricting when trying to create a control mesh that models the referenced surface. This restrictive approach and the need for additional fairing, which increases the number of control points, make Halstead et al.'s method for creating a subdivision surface not satisfactory for this thesis.

In Shen et al., 2014, this method of evaluation is used to convert NURBS surfaces to subdivision control meshes. After analysing a NURBS surface and extracting a quad mesh with help of the method of Bommes, Zimmer, and Kobbelt, 2009, the positions of the control vertices are computed similarly to the approach of Halstead et al.

The method used to position the control points in this thesis was developed in Thaller, Augsdörfer, and Fellner, 2016. Thaller et al. have created graph grammars that change the topology of a subdivision control mesh based on rules. When adding new control points, they have to be positioned such that the limit surface remains the same before moving some of them to model the feature. Their approach is to move the control points parallel to the normals in the surface until their limit positions lie on the original surface.

As a note of interest, Stam, 1998 has derived the calculation of limit positions for arbitrary points on the limit surface. Before his work, a direct evaluation of the limit positions of points around extraordinary vertices (i.e. in non-regular areas) was not known. The only solution was to decrease the non-regular area around an EV with iterative subdivision until the point of interest would lie inside of the regular area.

2.2. Quadrangulation of Surfaces

The quadrangulation of surfaces can be split into two main categories: Pure quad and quad-dominant meshes. Pure quad meshes have more constraints to ensure no non-quads are created. Quad-dominant meshes are free to improve upon surface properties by allowing non-quads. There is less research on creating meshes with

edge lengths that adapt to curvature because global sizing parameters facilitate high regularity in the results. While some research bases edge length on curvature, often the focus on isometry is such that the differences in edge length are very little or the minimization of the number of singularities does not allow enough flexibility.

2.2.1. Pure Quad Mesh Extraction

This section will discuss methods on how pure quad meshes can be extracted from triangles meshes. The discussion will part them into the following categories: Parametrizations, curve-based approaches and a divide-and-conquer approach. Parametrizations seem to have the biggest appeal because most research revolves around them. They flatten the mesh and bring it into a disc-shaped domain, ideal for quad meshing. Curve-based approaches create orthogonal sets of curves on the reference mesh which form a quadrangular grid. Lastly, dividing-and-conquering classifies parts of the mesh into areas where the quadrangulation is already known.

Parametrization

Parametrization based techniques try to flatten the reference mesh to a two-dimensional domain. This transformation needs a parametrization of the mesh. It maps vertices from the three-dimensional space of a mesh into a two-dimensional domain. Orthogonal lines can then be used to extract a quad mesh from the parametrised surface. The main difficulties when creating a parametrization are designing and optimising the parametrization and finding orthogonal lines for quad mesh extraction.

The most important properties of a parametrization are (summarised by Bommes, Bruno, et al., 2012 and Bommes, Zimmer, and Kobbelt, 2009):

Non-overlapping: Not connected areas of the reference mesh must be parametrised such that they do not have the same coordinates in the parametrised space. Handling of these areas is difficult and they would not create a flat parametrization.

Singularities on Boundaries: The inside of a parametrization is a grid of orthogonal lines where intersections will be vertices of valence four. If a singularity were to lie on the inside, it would distort the grid which would make it harder to find proper solutions. When singularities are on the boundary of the two-dimensional parametrization, pure quad quadrangulation is facilitated.

Continuity of Neighbours: When creating the parametrization, it is vital to ensure that lines can be followed over neighbouring faces. This means that edges must either have the same coordinates for both adjacent faces or have meaningful transitions between them that do not destroy the grid lines. Generally, these translations are an integer multiple of the grid size.

There are two ways of creating a parametrization: Chart-based and global parametrizations. Chart-based approaches have better topology but finding good solutions is harder. Global approaches are more straight-forward and do not need to create auxiliary structures.

Chart-Based: Chart-based approaches create a sub-graph spanning over the reference mesh which consists of quadrangular patches called charts or patches. All singularities of the extracted quad mesh lie on the corners of the charts but not all corners have to be singularities. This way, a pure quad mesh can be created where all singularities lie on the boundaries of the charts and the insides of charts contain only quads. The sub-graph is constructed such that, after finding the chart corners, the edges of the charts are superimposed on the mesh and consist of a path of edges of the original mesh. The number of charts is kept low compared to the size of the reference mesh since only singularities prevent regular quadrangulation and their numbers are limited. This reduces the solution space but the quality of the result depends on the singularities' placement. The created meshes are semi-regular quad meshes (c.f. the categorization of Bommes, Bruno, et al., 2012) which are favourable because of the low number of singularities.

Tong et al., 2006 use this method to create what they call a meta-graph. They find singularities in a harmonic form and create edges by growing patches from the singularities until boundaries are formed. Since the meta-graph of charts may have a genus greater zero, it may need to be cut open. After solving a linear system to generate two-dimensional coordinates for all vertices of the meta-graph, the tiling of the parametrization is calculated via linear constraints enforcing the important properties of the parametrization mentioned above.

Another variant for creating these chart graphs is using Morse-Smale complexes. What seems to be the first use of Morse-Smales complexes for quad meshing is by S. Dong, Bremer, et al., 2006. They define them as:

> [...] a cellular decomposition of a scalar (Morse-Smale) function over a manifold [...] computed more practically by tracing lines of steepest ascent/descent.

Vertices are placed at minima, maxima or saddles of the Morse-Smale function and together with the traced lines, they form a quadrangular chart structure. Dong et al. computed the function via an eigenvalue decomposition of the matrix defined by discrete harmonic weights of a Laplacian operator on piece-wise linear functions over a triangle mesh. It is a spectral surface analysis where the eigenvalues form the spectrum and the eigenvectors define piece-wise linear functions of progressively higher frequencies. Sometimes it is better to use the quasi-dual complex which is computed by exchanging edges and diagonals such that saddle vertices are removed, reducing the number of vertices in the complex. After an additional simplification of the complex, a parametrization is needed to optimise the layout of the patches and their boundaries because it can be misshaped and produce degenerate patches.

The parametrization is initialized and iteratively refined by repeating three steps: Solving a linear system to compute a parametrization, swapping vertices of the original mesh between chart boundaries, and relocating vertices of the complex. The mesh is generated by using a global parameter that divides the charts into an equal number of quads. Positions can be retrieved from the parametrization. The quality of the mesh depends on the choice of the eigenvector. Dong et al. can choose a set of eigenvectors based on the desired number of singularities and use the one with the lowest distortion. Huang et al., 2008 have added further control to this method through control over size and alignment. They still create an isotropic mesh. Zhang et al., 2010 add an anisotropic formulation to increase the manual control over the sizing of the quads.

Chart-based quadrangulation is not favourable when trying to create meshes with varying densities. The main issues are the boundary conditions between charts. To ensure a correct quadrangulation, the number of segments on the boundary fixes the number of quads in the respective direction of neighbouring charts. While these methods are possible, the set of results is too restricted for the goals of this thesis.

Global: The second way of designing the parametrization is a global view. Instead of first creating a sub-graph on the entire mesh, every triangle of the reference mesh is mapped into the parametrised space. This means that there are more elements to be parametrised, but an optimised parametrization is created on the mesh itself. Finding a planar embedding for the entire mesh is harder since there are more elements to consider. As before, it is important to cut the mesh at singularities. They should be fixed to the boundary of the two-dimensional domain.

A simple form of a global parametrization was created by Hormann and Greiner, 2000. They assume a mesh with boundaries on which they produce a regular quad mesh. First, they choose four fitting vertices to sit on the corners of the rectangle of the parametrization. Since the mesh has boundaries, the boundary of the rectangle will equate to the boundary of the reference mesh. The parametrization is optimised iteratively such that the new vertex positions are calculated via a weighted sum of their neighbours. Since they assume a mesh with boundaries, the number of applications for this algorithm is limited.

A more involved method is done by Kälberer, Nieser, and Polthier, 2007. They derive their parametrization with the use of branched covering spaces. They use an input vector field to compute branch points (i.e. singularities) and parameters for edges to ensure a globally continuous parametrization. Their derivation of constraints ensures that branch points lie either on one of the grid's vertices or in the centre of a grid tile. If a branch point lies in the centre of a grid tile, it will produce a triangle on extraction. To ensure a pure quad mesh, the size of the grid must then be halved. After computing a cut graph (Kälberer, Nieser, and Polthier, 2007 claim that its computation type does not influence the result significantly), the parametrization is computed via a Hodge decomposition. The grid used for extraction is an integer based one. Integer translations for all edges on the boundaries induced by the cut graph must

be added because parametrised positions of the edges of both adjacent faces may lie on opposing sides. This method does not use density parameters. Because density changes induce more singularities, which induce the need for subdivision, it is not viable for the goals of this thesis.

For their parametrization, Bommes, Zimmer, and Kobbelt, 2009 have created a mixed integer solver that allows them to specify linear constraints for the parametrization in a straight-forward way. The solver is able to compute an optimal solution for a problem with real and integer-valued parameters via computing a real-valued solution and then performing a series of rounding and solving iterations which progressively fix all integers. All fields and parametrizations are computed with this solver. They first compute a smooth cross field which is optimised such that an energy based on the angular difference of neighbours is minimised. The angles are defined over orientations for each face, together with integers for each edge which define the number of 90-degree jumps between both faces. An initial set of fixed orientations which guides the optimization is computed from areas of the mesh where its orientations are well defined (e.g. feature lines).
The mixed integer solver now has to find angles for each face and integer jump numbers for each edge which minimise the energy. Computing the optimal parametrization then follows a similar approach. First, the mesh is cut open via a cut graph. The cut graph is calculated by generating a dual spanning tree having a random vertex as the root and ensuring the singularities lie on the boundary of the cut. Now, the constraints for the solver can be defined. Inner edges must have equal parametrization up to integer translation and rotational jumps (which are fixed by the cross field) while edges on the cut must have integer positions at both ends and integer translations between the parametrization of both adjacent faces. The energy that is minimised is a local orientation energy for each triangle. They add an anisotropic norm to favour different edge lengths over bad orientations, add feature alignment constraints, relocate singularities and add local stiffening to improve results. Extracting a pure quad mesh is done through an integer grid. Their comparisons with Huang et al., 2008 and Kälberer, Nieser, and Polthier, 2007 show that their approach can produce similar if not better results.
They improve upon this method in Bommes, Campen, et al., 2013 where, after computing a globally smooth parametrization, they perform additional refinement steps before creating the quad mesh. This mixed integer based solution is then refined before a mesh is extracted. The refinement allows them to change the resolution of the final mesh and improving the results.

For a more architectural approach, Liu et al., 2011 try to create a pure quad mesh that where the quads are made planar in a post-processing step. Planarity is important for construction because it is much easier to create such tiles. Their approach uses a conjugate direction field to guide the parametrization. The parametrization is computed by creating a mixed integer formulation and adopting Bommes et al.'s

mixed integer solver. Singularities cannot be placed while creating the direction field, but they can be moved at later stages. The final step is a planarity optimization which has better results combined with their quad mesh when compared to other quad re-meshing methods.

To change these isometric parametrization techniques into an anisotropic formulation which adaptive edge lengths, Kovacs, Myles, and Zorin, 2011 have created an anisotropic metric that may be used for feature-aligned parametrizations and harmonic maps. The shown results are different depending on the parametrization. Their results seem to struggle with alignment to features and principal curvature directions sometimes and need manual constraints for alignment.

Curve Building

Using curves on the mesh to produce a pure quad mesh seems to be difficult. Most approaches using curves, produce quad-dominant meshes. This is because creating perfect layouts of curves, which do not get cut off by two neighbouring curves by getting too close - thus creating a non-quad - cannot be avoided entirely.

Campen, Bommes, and Kobbelt, 2012 have an interesting approach to using curves. Instead of creating a quad mesh from the curves, they use the curves as intermediary information. From a graph of curves, the dual graph creates a chart layout like discussed earlier. The most important observation for this method is that the loops must separate singularities of a curvature induced cross field. As discussed, singularities must be represented by vertices in a chart layout. Since every patch in the graph of loops will be a vertex in the dual graph, no two singularities may lie within one patch. To construct the loops, all pairs of singularities need to be separated with loops. Campen et al. use a greedy selection of loops where the largest of all minimal loops for pairs of singularities is selected. This selection is chosen because it leads to better, less locally oriented results. The created chart layout is optimised by moving the vertices. Extensions to this method include an additional introduction of feature curves, and curves improving the layout to increase alignment to mesh features. Because this approach uses a chart layout, the same problems concerning curvature adaptive edge lengths will arise as with the previously discussed methods which do the same.

Divide-and-Conquer

For another variation of chart-based approaches, Zhang et al., 2013 create a chart layout by segmenting the mesh into classified and unclassified patches. Classified patches are patches where an optimal quadrangulation scheme is already known and no further computation for the interior is necessary. They are built such that their

densities can be changed through subdivision. Unclassified patches are quadrangulated with a wave-based quadrangulation method and their result can be reused for similar patches. To ensure a proper quad mesh, stitching constraints have to be added that make sure the number of quad meshes is equal on both sides of chart boundaries. After solving the stitching parameters with the mixed integer solver of Bommes, Zimmer, and Kobbelt, 2009, the mesh can be extracted via stitching together all sub-meshes of the charts. The results depend on the quality of the classification. When much of the mesh can be classified, their results are fast and have good symmetry. For bad / no classification, the results are similar to related methods. Since this method is based on chart layouts, it has similar problems when trying to create curvature adapted edge lengths.

2.2.2. Quad-Dominant Mesh Extraction

Quad-dominant mesh extraction methods share many properties of pure quad methods, but often have decisions and techniques which allow or induce the creation of non-quads. The research is split into three main concepts: Parametrised and curve-based approaches - similar to the pure quad-based ones, and a triangle mesh decimation approach. There seems to be less research being conducted on quad-dominant re-meshing than on pure quad re-meshing. As mentioned before, quad meshes are very favourable for a variety of tasks. For similar tasks, quad-dominant meshes need special cases for non-quads or have a higher density and more singularities when transformed to pure quad meshes via one step of subdivision. The following discussion will lay out quad-dominant methods and how they differ to similar pure quad mesh methods.

Parametrization

Quad-dominant parametrizations are very similar to pure quad-based ones. The idea is the same but some techniques differ because they are more prone to create non-quads.

Chart-Based: Marinov and Kobbelt, 2006 have created a re-meshing scheme based on charts. The charts are computed via a slight variation of Cohen-Steiner, Alliez, and Desbrun, 2004. Each chart is parametrised and their edge paths are approximated as cubic curves. The difference to other chart-based approaches is that Marinov et al. do not quadrangulate the local parametrizations for each patch but instead create a network of orthogonal cubic curves. For this, the cubic curves of the chart boundaries are mapped to the parametrization and are segmented. Boundary constraints are introduced, ensuring an equal segmentation for adjacent charts. The segments are seeds to create a curve network. Its curves are created from an initial setup based on the segments that is then improved by swapping curves. Additionally, edges with

sharp intersections are removed.
The result depends on an energy formulation. It can be modified to favour squares or rectangles because the shapes of the resulting faces are defined by the intersections of the curves. In some rare cases, T-joints or non-quads might be allowed to an extent that can be defined in the energy definition. After re-meshing and smoothing of the quadrangulated charts in parameter space, the positions are mapped to 3-dimensional space, creating a quad-dominant mesh. While this method does not create a pure quad mesh, it faces similar problems concerning curvature adapted edge lengths as the pure ones. An alteration might allow for curves to be created in the centres of charts to change the number of vertices on opposite boundaries of a chart. This would add more flexibility to adapt edge lengths to curvature because densities would not have to stay the same in one dimension of a chart. This change introduces a lot of new degrees of freedom during optimization and would be difficult to achieve.

Global: What seems to be a cornerstone for parametrised approaches to quad re-meshing is the research in Ray, Li, et al., 2006. Contrary to the approaches for global parametrization discussed earlier, Ray et al. do not cut open the mesh via a cut graph and use a periodic notion to define orthogonal lines in the parametrization instead of integer lines. Like others, they use a principal direction based cross field which is created by propagating directions from highly anisotropic regions. The parametrization is done globally for each triangle with a constraint formulation for neighbouring triangles allowing 90-degree rotations and integer multiples of 2π as translations (as opposed to integers in other approaches). Similar to other methods, these constraints ensure continuous grid lines on the parameter domain. The parametrization is computed via minimising Ray et al.'s quadratic error formulation with a penalty against degeneracies. Mesh extraction is then done via tracing grid lines on integer multiples of 2π on the parametrised domain.
Special handling of singularities via the splitting of triangles must be done because the optimization for the parametrization does not incorporate them. Non-quads might be constructed since singularities in the centre of a grid tile will produce triangles. It is a good example why most other approaches take special care to position singularities on the boundary of the cut graph and on grid intersections of the parametrization to ensure a pure quad mesh. Like other approaches which compute a global parametrization that relies on a regular grid for mesh extraction, curvature adapted edge lengths cannot be enforced easily. Because this method allows for a more flexible placement of singularities however, further research on a grid with a density adapted to curvature information might be viable.

Curve Building

Building curves on the reference mesh itself is a straight-forward way of meshing quads. Enforcing the orthogonality such that only quads are created, however, is difficult. Hence, as noted before, more quad-dominant approaches to curve building were found than pure quad ones.

A very early example for curve building methods was created in Alliez et al., 2003. They assume a genus-0 surface patch (i.e. with boundaries) as input. On this surface, they compute a graph for tagged features (for curvature alignment) and a curvature tensor as direction field. For faster processing, the curvature is then flattened to a 2-dimensional parametrization domain. Umbilical points on the surface are seeds for a Runge-Cutta curve interpolation on the parametrization. Depending on the desired distances, new seeds along the traced curves are placed in a priority queue for further processing. Curve tracing may stop when it is too close to another curve - based on the desired density. Around umbilical points, where density of the curves may not be enough, additional point sampling is performed. After cleaning the data (e.g. dangling lines, vertices with none or two attached segments, ...) the resulting mesh is created by intersecting all curves to form quads and triangulating the remaining points. This creates a quad-dominant mesh that explicitly allows anisotropy of the resulting rectangles.

Further extending the approach of Alliez et al., Marinov and Kobbelt, 2004 add methods to allow for inputs to be closed meshes with arbitrary genus. They do this by not depending on a global parametrization and dealing with umbilical regions differently. As before, a curvature tensor field is created. However, it is created so that anisotropic regions propagate their directions into isotropic regions. This method is found in similar approaches because it deals with the problem of not knowing principal directions in umbilical regions. Seeds for lines are placed in anisotropic regions for well-defined directions and may switch minimal/maximal principal directions when integrating if needed. The rest is similar to Alliez et al.'s approach, except for smaller differences in used data structures. This creates quad-dominant meshes where non-quads are triangles or n-gons.

A different approach for creating direction fields for the curves was done in S. Dong, Kircher, and Garland, 2005. The steps are similar to Marinov et al.'s work. First, a cross field is created and then curves are traced on the surface with respects to density and the desired amount of isotropy vs. anisotropy. The difference lies in creating the cross field. Dong et al. do not use the principal curvature of the surface but compute a constrained harmonic scalar field on the mesh as a foundation for the directions. From this scalar field, a cross field can be computed by taking the gradient as one direction and a 90-degree rotation around the normal as the second direction. As an additional degree of freedom, a user may define vertices to be constant in the scalar field which introduces the ability to place extrema. This is important because

maxima/minima of the scalar field are where curves will converge. Similar to the other approaches, curves are then integrated and sampled on the cross field. This results in a method for quad-dominant mesh extraction which provides a user with great control over the placement of singularities.

Curve-based approaches seem to incorporate curvature adaptive edges length well. However, the distances between curves do not vary much. This means that these curve-based approaches are less flexible in their adaptation to changes in curvature.

Triangle Mesh Decimation

Methods based on triangle mesh decimation do not try to re-mesh the surface as a quad mesh but try to change the topology of a triangle mesh such that it becomes a quad-dominant mesh. An example of such an approach is the work of Lai, Kobbelt, and Hu, 2010. They create a quad-dominant mesh via iterative relaxation such that edges of a triangle mesh align with the principal directions of a surface. The principal directions are calculated via a curvature analysis which yields a direction field. After smoothing the curvature tensor, the triangle mesh is re-meshed as a second triangle mesh with a density that adapts to surface features. The base for the final quad mesh is constructed via iteratively moving vertices to optimised positions and changing the mesh topology where needed. Optimised positions for vertices are calculated in a parametrization of the 1-ring neighbourhood of each vertex. A quad-dominant mesh can be created by deleting all edges that are not aligned with the direction field because they can be seen as the diagonals in feature-aligned quads. This approach allows T-vertices and needs a post-process is to merge T-vertices to reduce their numbers. The result is a quad-dominant mesh with non-quads in areas of singularities of the principal direction field or because only a local minimum was found in the optimization. While this approach uses curvature information to adapt the density of the mesh, the differences in edge lengths are small and the authors have not presented results which reduce the number of vertices by the amount needed for the goals of this thesis.

2.2.3. Instant Meshes

Jakob et al., 2015 have created a method for meshing a surface into an isotropic triangular or quad-dominant mesh. They achieve this by using methods similar to existing techniques but use local approaches which do not need complex global optimizations. Besides being quick, these local approaches also work for non-manifold input which many other approaches are not able to process. Their method comprises two types of optimization and an extraction step. Both optimizations are initialized randomly. The following steps are performed:

Orientation Field. The first step is the computation of an orientation field on the reference mesh. The orientation field is based on the N-RoSy field from Ray, Vallet, et al., 2008. It is defined through N evenly spaced tangential vectors for every vertex which will serve as directions for further steps. Different values for N are possible, but $N = 4$ is most important for this thesis since the resulting cross field is favourable for creating a quadrilateral mesh.

The orientations of vertices are optimised by minimising an energy through local iterations that resemble the Gauss-Seidl method. Orientations are represented by one representative direction for each vertex and a set of integers for each pair of neighbours to encode the ambiguity added by the rotational symmetry of the orientation field. While optimising, the algorithm alternates between doing one Gauss-Seidel iteration and finding the set of rotational integers which minimises the energy via brute-force. While others use constraints to align their orientation fields to anisotropic regions of the mesh, Jakob et al. have found an extrinsic smoothness formulation to measure the energy during the minimization. Unlike the intrinsic formulation, they do not rotate the orientation information of one vertex into the tangent plane of its neighbour to compute an energy to be minimised. This leads to a behaviour which naturally snaps to edges of the reference mesh and propagates the orientations of these features to isotropic regions.

While their formulation does not need constraints for proper alignment, user interaction is still possible. A user may add constraints that force orientations and may add strokes to guide the orientations. When adding the integer rotations of a circular path around a vertex, singularities in the orientation field, which will lead to extraordinary vertices in the extracted mesh can be found. These singularities can be moved by a user and may cancel each other when moved close together and if they are valence-3 and valence-5, respectively. While these interactions are favourable when trying to extract a mesh manually, they are not used in this thesis because the focus lies on automated extraction.

Position Field. The next step is creating a local, discontinuous parametrization for each vertex. The parametrization is locally oriented along the computed orientation field. Its grid size (i.e. the length of a translation by one) is defined through a global scaling factor, which is determined by the desired amount of vertices on the extracted mesh. Every vertex minimises its local smoothness energy by finding optimal integer translations and an optimal representative position in the parametrization based on its neighbours. Because the parametrization is discontinuous over edges, no time consuming global optimization is necessary.

Similar to the orientation field, a representative position for each vertex and integer translations for neighbouring pairs are optimised via local Gauss-Seidel iterations and a brute-force search. Like before, an extrinsic smoothness energy is used that need not rotate tangent planes and naturally snaps to surface features. After each iteration, the representative positions of all vertices are rounded to the nearest position of the integer grid of the parametrization.

Since the computations of orientation and position field are very similar, manual interaction is possible here as well. Singularities in the parametrization, which may cause non-quads, can be found by summing integer jumps around a vertex. They can be moved and opposing types can cancel each other. Like before, this process is manual and automatic movement of these singularities does not warrant the increased computation time.

The flexibility of not performing global optimizations and from allowing non-quads is conductive to adapt the integer grid sizes according to curvature. A computation of the position field with scaling of integer translations based on curvature is the main point of change to this approach throughout this thesis. The different ways of changing the grid size are discussed in Section 3.4.

Extract Mesh. Lastly, a mesh based on the computed parametrization is extracted. Approaches which draw grid lines on the parametrization do not work here because the parametrization is discontinuous over edges (Bommes, Zimmer, and Kobbelt, 2009 create a continuous parametrization, for example). Instead, the optimised integer translations between neighbours are used to find the structure of the mesh to be extracted. Neighbouring vertices with integer translations opposite in sign, lie on the same integer position of the parametrizations' integer grid. Taking the average of the representative positions of a cluster of vertices on the same integer grid position yields one vertex of the extracted mesh. Similarly, an integer jump of one in the parametrization between neighbouring vertices indicates an edge in the extracted mesh.

In their research, Jakob. et. al. have found that the Gauss-Seidel optimization will get stuck in local minima easily. To combat this, they use a multi-resolution hierarchy which merges vertices into connected components. The ordering based on which vertices are merged is based on a weighted normal vector similarity measure. Other approaches for weighing were made to influence the convergence of the position field optimization which are discussed in Subsection 3.1.4. The optimization will start at the coarsest level of the hierarchy and will use the results of each level to initialize the parameters of the next level. This improves the results and avoids local minima. Figure 2.1 shows an overview of the results of the whole process and a comparison which highlights the improvements to the convergence when using a multilevel-hierarchy.

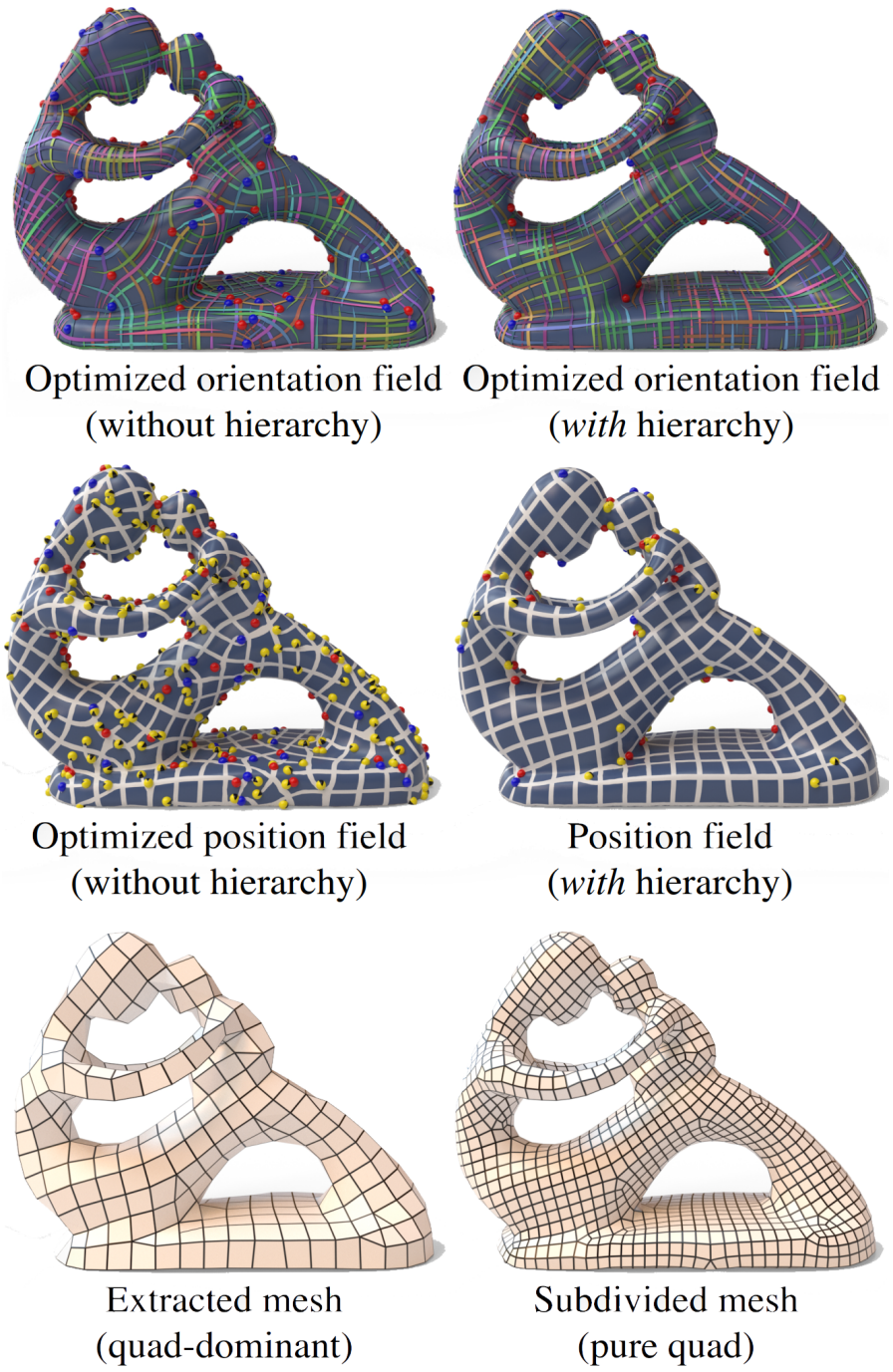


Figure 2.1.: An overview of the different steps in the Instant Meshes method and a comparison with and without multi-resolution hierarchy (Jakob et al., 2015).

3. Implementation

The goal of this thesis is a modified algorithm of Jakob et al., 2015 which creates quad-dominant meshes where the edges are aligned with features and the edge lengths adapt to curvature. To create such a mesh instead of an isotropic mesh, several changes have to be made to the work of Jakob et al. Curvature must be transformed into meaningful edge lengths and the algorithms by Jakob et al. must be adapted to use these differing scales. To learn the intricacies of the position field (mentioned in Section 2.2.3), many techniques to use curvature information in the calculations have been introduced in this thesis. Every technique was evaluated, their behaviour was observed and improvements were made.

The following steps are performed to extract a quad-dominant subdivision control mesh:

Initialization: Build the multi-resolution hierarchy. For each level of the hierarchy: Estimate the curvature (Section 3.1), compute the curvature adaptive grid scale for every vertex (Section 3.2), and classify sharp feature vertices (Section 3.3) like ridges or creases.

Orientation Field: Optimise the orientation field with respect to the constraints of the classified feature vertices (Section 3.3).

Position Field: Optimise the position field (Section 3.4).

Mesh Extraction: Extract the mesh from the parametrization defined through the optimised position field (Section 3.5).

Ensuring Manifoldness: Find and remove all non-manifold edges and vertices (Section 3.6).

Mesh Post-Processing: Perform various post-processing steps to improve the topology and geometry of the extracted mesh (Section 3.7).

Subdivision Control Mesh: Create a subdivision control mesh from the quad-dominant mesh (Section 3.8).

The software developed by Jakob et al. has been adapted to incorporate all changes and additional settings. Section 3.9 gives an overview of the changed interface and the location of different sets of options.

3.1. Curvature

Subdivision control meshes should be sparse while still being able to model all necessary surface details of the object. This is not an easy task, as changes in detail of modelled objects may happen quickly and the subdivision control mesh must adapt its density accordingly. These changes in surface detail are measured by looking at the curvature. Areas with high curvature represent high-detail features with more creases and ridges, and a finer control mesh is necessary to model the level of detail. Areas with low curvature can be represented accurately with few control points. Changes in curvature need to be taken into consideration when adjusting the control mesh's density. These changes in the density of Catmull-Clark subdivision control meshes are difficult to achieve because of their quad-based nature. Transitions from low to high densities will introduce additional extraordinary vertices. The following will introduce the extraction of curvature from the reference mesh and the changes to the work of Jakob et al.

3.1.1. Curvature Estimation

There are two ways to estimate curvature on a piece-wise linear mesh: Fitting parametric surfaces locally around each vertex or estimating curvature in a discrete sense based on the neighbourhood of a vertex. Petitjean, 2002 and Gatzke and Grimm, 2006 have surveyed many methods and give an overview. Parametric surface fitting generally produces better results while having a greater time complexity. Most discrete methods are linear in time because they compute sums over neighbours, but are more susceptible to noise. A 1-ring neighbourhood is the standard case while a 2-ring neighbourhood may combat the effects of noise. 3-ring neighbourhoods or higher do not warrant the increased computational cost (Gatzke and Grimm, 2006). Methods from both types were tried, but the principal curvatures were more precise and uniform when fitting a parametric surface. The improved precision warranted the increased time complexity (a factor of three for an input mesh with 50000 vertices and 100000 faces).

Estimation via Discrete Sums: C. Dong and Wang, 2005 compares several curvature estimation algorithms, and create a simple, yet effective one. They base their work on the method of Chen and Schmitt, 1992 and change the matrix-based formulation to a simpler summation over all neighbours. The curvature of a point on a two-dimensional curve can be viewed as the inverse of the radius of the circle with the same slope as the curve in the point. For three-dimensional surfaces, this formulation becomes more complex because there is an infinite number of tangential curves on the surface going through a point. The normal curvature of a point along a curve is defined by the length of the projection of the curve's second derivative vector onto the normal plane of the surface in the point. All curves with the same tangent line at this point have the same normal curvature. The expression to formulate the

curvature κ_n of a point on the surface in any tangential direction t (defined via its angle Θ to the principal directions) can be done with the principal directions (e_1, e_2) and principal curvatures (κ_1, κ_2):

$$t = e_1 cos(\Theta) + e_2 sin(\Theta) \tag{3.1}$$
$$k_n = k_1 cos^2(\Theta) + k_2 sin^2(\Theta) \tag{3.2}$$

The method in C. Dong and Wang, 2005 uses a reformulated version of Equation 3.2 where a least-squares method over all neighbours can find an approximation of the principal curvatures:

$$\kappa_n(t_i) = a cos^2(\Phi_i) + b cos(\Phi_i) sin(\Phi_i) + c sin^2(\Phi_i) \tag{3.3}$$

Where $\kappa_n(t_i)$ are estimated normal curvatures toward each neighbour and Φ_i is the angle between the tangent direction t_i to neighbour i and a reference tangent direction. Normal curvatures toward the neighbours are computed via normal differences and the reference tangent direction points toward the neighbour with the largest estimated curvature. This creates a reference coordinate system like the one the principal directions are spanning (they are not known yet). Inserting these parameters into Equation 3.3 defines a system of equations over all neighbours. Through least-squares approximation the parameters a, b, and c are estimated and from them, the principal directions and their curvatures can be calculated. While the estimated curvature computed through this method yields acceptable results, the principal curvature directions are not satisfying. Hence, a parametric surface fitting method was tried out. Figure 3.1 compares the principal curvatures of both methods and shows why the directions computed in C. Dong and Wang's approach are not precise enough when compared to the parametric surface fitting.

Cubic Order Parametric Surface Fitting: The parametric surface fitting method of cubic order developed in Goldfeather and Interrante, 2004 estimates principal curvature directions well and computes satisfying results. To estimate the principal directions and curvature, a cubic order B-spline patch is fitted onto the neighbourhood of a vertex. The B-spline patch is of order three because lesser orders produce less accurate results (because they do not use the normals of the neighbourhood). Higher orders do not improve the results considerably while needing a bigger neighbourhood and more time. For estimation, a cubic surface is computed by bringing neighbouring vertices into a local coordinate frame and creating a system of equations where the neighbours describe the shape of the parametric surface. Every neighbouring point introduces three equations with one equation for the position and two equations for the normals. This system of linear equations can be written in matrix form and its least-squares solution (computed via a QR decomposition) is the Weingarten curvature matrix of the cubic surface. The Eigenstructure of the Weingarten curvature matrix is such that its Eigenvalues are the principal curvatures and its Eigenvectors are the corresponding principal directions.

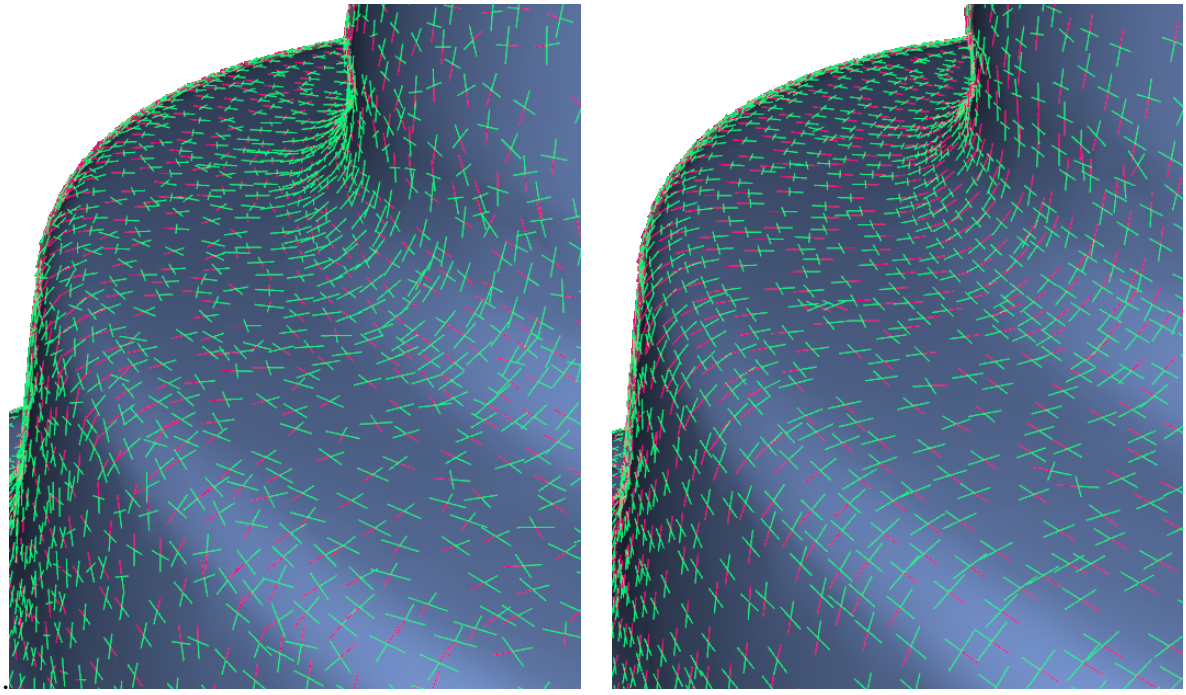
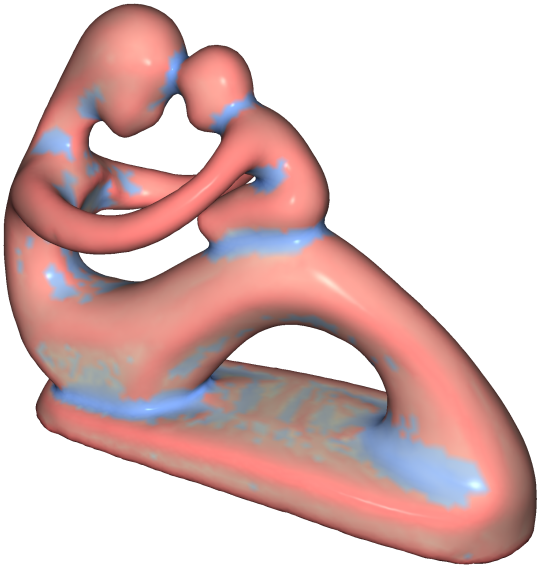


Figure 3.1.: This figure shows the resulting estimated principal curvatures for the approaches of C. Dong and Wang, 2005 (left) and Goldfeather and Interrante, 2004 (right). Goldfeather and Interrante's approach estimates the principal directions more accurately and more uniformly on feature lines. The most notable difference in accuracy is on the ridge where C. Dong and Wang's approach does not manage to estimate uniform principal directions.

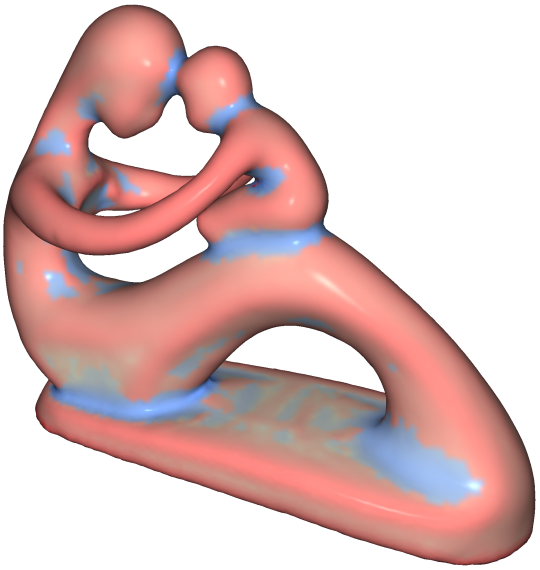
Both methods for curvature estimation allow a choice in the size of the neighbourhood of each vertex. The term *n-ring neighbourhood* is used to describe this choice. The parameter n defines how many additional neighbours of neighbours are considered being part of the n-ring neighbours. It is important that the neighbourhood is not too big because high-frequency detail might be lost. Conversely, a small neighbourhood might not convey enough information for curvature calculation when it is malformed (e.g. when all neighbours lie on one half of a circle in a 1-ring neighbourhood). Greater neighbourhoods can decrease the influence of noise on the estimation. Differences between the results of a curvature estimation with different neighbourhood sizes can be seen in Figure 3.2. Empirically, a 2-ring neighbourhood is a reasonable choice.

3.1.2. Curvature Selection

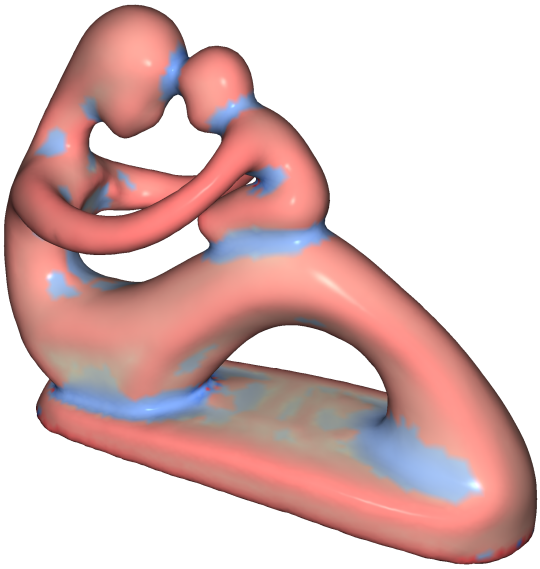
When designing the method to convert curvature to edge length, there are several values that result from the curvature estimation discussed in 3.1.1 which may be chosen as reference values. For Gaussian curvature, no conversion to a scaling factor exists because it characterises shape and its magnitude cannot be related to edge length. While mean curvature does have a similar magnitude as the principal curvatures, it is a shape characteristic as well and its value does not represent the normal curvature of the mesh accurately. Using both principal curvatures as edge length indicators (called directional curvature in this thesis) will lead to worse results. This stems from the fact that having different scales for both directions in the position field optimization of Jakob et al. introduces too much variation in the edge length such that a proper virtual grid structure cannot form. Taking the largest absolute value of both principal curvatures is the best choice ($\kappa' = max(|\kappa_1|, |\kappa_2|)$) because it ensures that no detail is lost. Figure 3.3 shows a comparison of these choices. The curvature is estimated with a neighbourhood of two, is not averaged (to better show the differences) and the edge length is calculated by the error based estimation method using the osculating circle discussed in Section 3.2.



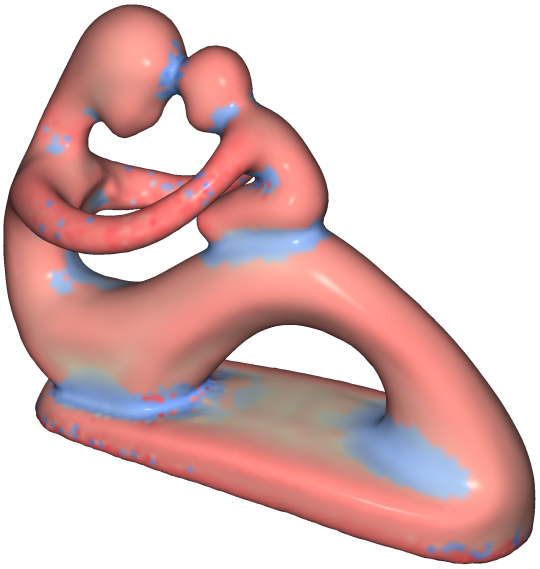
1-ring neighbourhood.



2-ring neighbourhood.

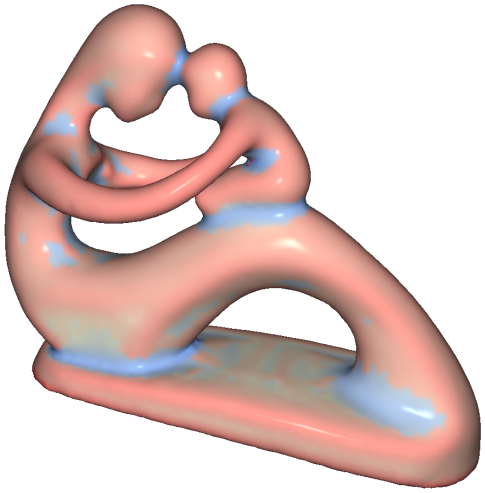


3-ring neighbourhood.

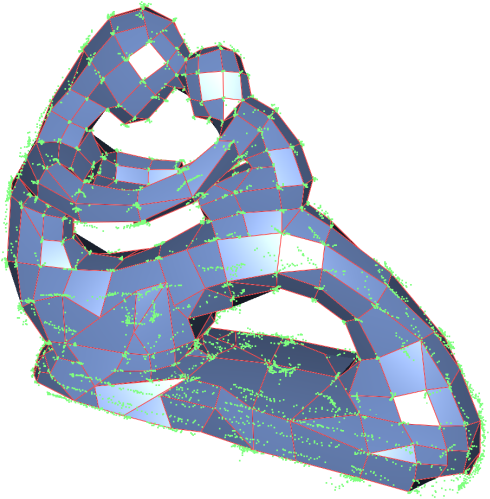


5-ring neighbourhood.

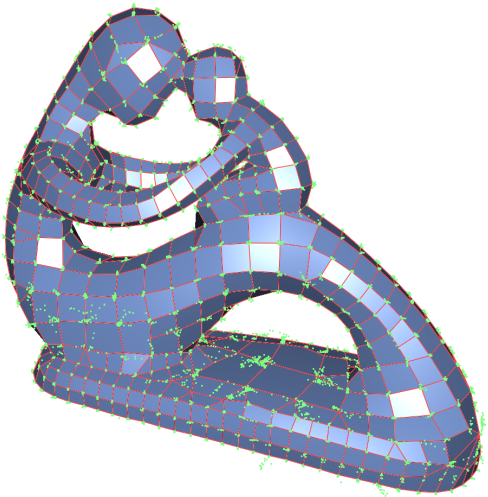
Figure 3.2.: This figure shows the largest principal curvature (blue for negative and red for positive curvature) estimated from differently sized neighbourhoods. Small neighbourhoods are prone to noise while bigger neighbourhoods lose high-frequency detail. This can be observed for 3-ring and 5-ring neighbourhoods on the neck. These neighbourhoods also have outliers with false signs in curvature (e.g. at the base) because the neighbourhood reaches too far into areas with different curvatures. Bigger neighbourhoods also take exponentially more time to estimate the curvature. On a PC with 16GB RAM, and a four core CPU with 3.5GHz, the run-times for 1-ring to 5-ring neighbourhoods on the shown reference mesh are 62ms, 166ms, 343ms, 899ms and 3305ms, respectively. The reference mesh has 14.000 vertices and 28.000 faces. These observations show that a 2-ring neighbourhood is the best choice for avoiding noise while not losing too much detail information.



Largest Principal Curvature.



Directional curvature.

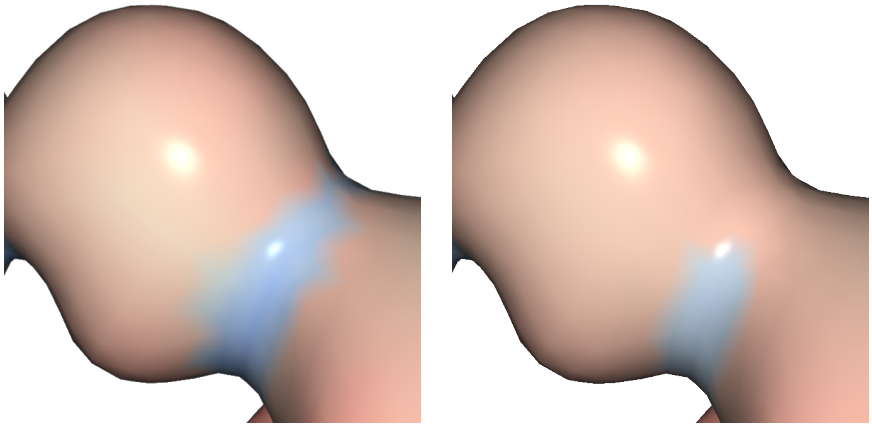


Largest absolute principal curvature.

Figure 3.3.: This figure shows results for using both principal curvatures (directional curvature) to compute the edge lengths and for using only the largest absolute principal curvature to compute the edge length. Directional curvature creates rectangles and the largest principal curvatures creates square quads. Both methods do not lose detail information by using the largest principal curvature as a guide for edge length. The directional curvature selection however, can represent the surface with less vertices. Unfortunately, directional curvature introduces more variety into the position field optimization which decreases the quality of the formed clusters for extraction (green dots) and often can not represent all sharp features (e.g. at the base). This means that the directional curvature should have the same accuracy in theory but the position field optimization does not always achieve this in practice.

3.1.3. Curvature Averaging

Observations have shown that a smooth distribution of curvature and little noise improves the formation of clusters in the position field optimization. Since the curvature estimation is discrete, the curvature information might not be as smooth as needed for good results. This may be due to noise in the reference mesh or multiple different features coming together in the mesh. To combat these issues, different averaging methods were developed. When averaging curvature, absolute curvature should be used because the magnitude of the curvature is important for defining the edge lengths, not the sign. This is visible in hyperbolic areas of the surface - where positive and negative curvature would negate each other through averaging. Figure 3.4 shows this issue when using the simple averaging method mentioned below.



Estimated Curvature Averaged Curvature Estimation

Figure 3.4.: These pictures show the issue when averaging the curvature without absolute curvature values. The sum of positive and negative curvature value when averaging makes the curvature vanish.

For all vertices of the reference mesh, the averaging methods described below compute a weighted average over all neighbours and the vertex itself. There are three user-controlled parameters. The number of *averaging iterations* n is a measure to control the amount of averaging. More iterations will make curvature information more uniform and curvature changes more gradual. A *damping factor* $\lambda \in [0, 1]$ controls the influence of the neighbours. Higher values for the damping factor will slow down the averaging. Its name was taken from Lai, Kobbelt, and Hu, 2010 who use it for the same purpose. The final parameter is the *size of the neighbourhood* (1-ring to 5-ring neighbourhood). Empirically, more than a 5-ring neighbourhood is too big because the curvature information becomes too uniform and iterations need much more time. Similar to the curvature estimation described in Subsection 3.1.1, a greater neighbourhood can compensate for noise in the reference mesh. The averaging methods define different weights w and perform the following weighted sum:

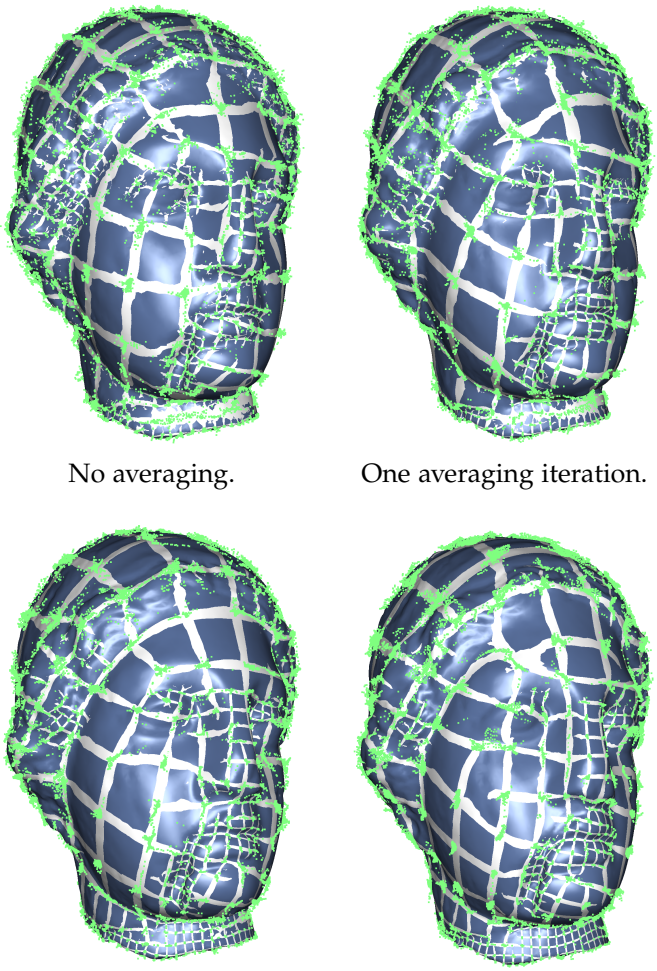
procedure AVERAGING
  for $i = 1$ to n **do**
    for all $v \in reference\_mesh$ **do**
      $W \leftarrow \sum_{x \in neighbours(v)} w_x(v)$
      $curv_{new}(v) \leftarrow \lambda \cdot curv_{old}(v) + \frac{1-\lambda}{W} \sum_{x \in neighbours(v)} w_x(v) \cdot curv_{old}(x)$
    end for
    $curv_{old} \leftarrow curv_{new}$
  end for
end procedure

Simple Average: The most simple method is an average of neighbouring vertices where all neighbours x of vertex v have a weight of $w_x(v) = 1$. Its goal is to smooth the mesh's curvature information. Through this, areas, where different curvatures meet are smoother, sudden changes in curvature are replaced by more gradual changes, there is less difference between the highest and lowest curvature, and small spots of high or low curvature become bigger - albeit less extreme. Figure 3.5 shows the differences in curvature and position field between multiple combinations of neighbourhood sizes and averaging iterations with a damping factor $\lambda = 0.5$.

Similarity-Based Average: To see whether increased uniformity in areas with similar curvature while not smoothing out gradual changes in curvature improves results, an averaging method with weights based on the curvature similarity between neighbouring points was created. Additionally, when similar curvature values have a higher weight than differing ones, high curvature values are not reduced as quickly when averaging.

The weights w for this averaging are defined as follows (v is the reference vertex and x is the neighbour):

$$w_x(v) = \frac{1}{(k * |curv(v) - curv(x)|)^2 + 0.01} \tag{3.4}$$



No averaging. One averaging iteration.

Three averaging iterations. Six averaging iterations.

Figure 3.5.: This figure compares different averaging iterations for the simple average where all neighbours have the same weight $w = 1$. The averaging shown here uses a 2-ring neighbourhood and a damping factor of $\lambda = 0.5$. A higher number of iterations will create tighter clusters in the parametrization (green dots) but will also lose detail in high-frequency areas and spread high curvature information into low curvature areas. These observations are most noticeable where low and high curvature areas meet. An example for the loss of detail is the nose and the eyelids where edges become larger for a higher number of iterations. Conversely, smaller edges than necessary can be seen at the back of the neck and around the nose especially at six iterations. This means that some averaging will improve the position field optimization but too much averaging will distort the curvature information. The resulting meshes are shown in Appendix A, Figure A.1.

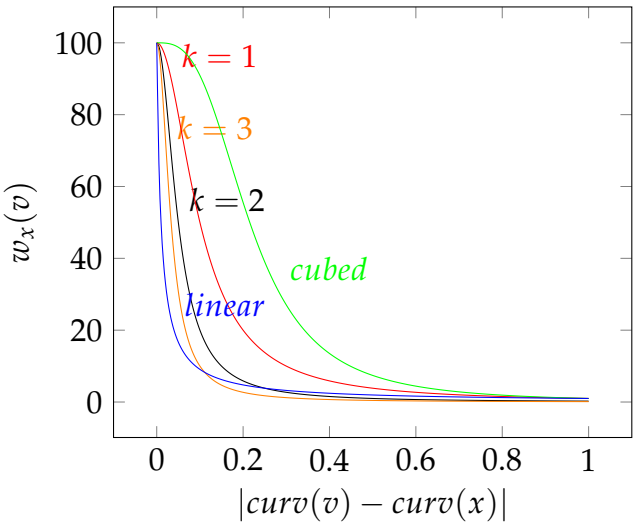
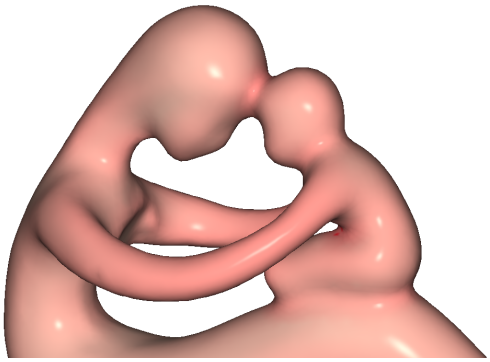


Figure 3.6.: This figure shows the plot for similarity-based weights of neighbouring vertices with different parameters. Equation 3.4 defines the shown weight $w_x(v)$ for vertex v with neighbour x and a normalized curvature $curv()$ for each vertex. The parameter k changes the factor in front of the curvature difference. It influences the speed of decline for the weights. *linear* and (cubed) show how the weighting would look like if the curvature difference was linear or cubed instead of squared. Cubed behaviour declines too slowly and linear behaviour declines too quickly.

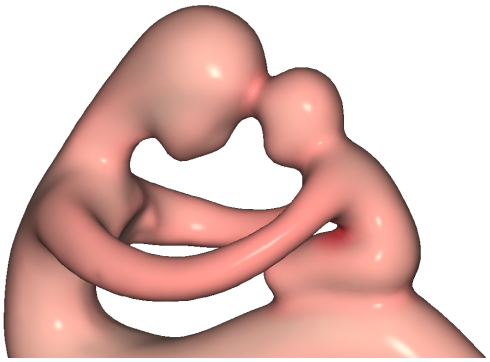
The influence of vertex x on vertex v is scaled by an inverse square of the difference of their normalised largest absolute principal curvatures. Because the limit approaches infinity for points with very similar curvatures, a normalisation factor of 0.01 is added which limits the value of the weight to 100 for differences close to zero. The scaling k changes the strictness of the weight calculation. $k = 2$ and the squared behaviour offer a balanced decrease of weights (otherwise most neighbours will have similar, low, weights). Figure 3.6 shows a plot of this equation, why the square behaviour was chosen and the influence of k. Figure 3.7 compares this averaging method to the simple average with uniform weights $w_x(v) = 1$.

An additional variation of above-mentioned averaging is one which increases the weight further if the normals of vertices are very similar. It did, however, not change much. The reason might be that vertices with very similar normals already have similar curvatures.

Expand High Curvature: Observations on the behaviour of the changed variations of the algorithm suggested that there are problems when areas of high curvature are small. They are reduced too quickly by the previous averaging methods. Thus, an averaging scheme which tries to expand high curvature areas was tried: Low curvature areas have a constant low weight and higher curvatures have a weight which increases with the curvature. The weight w is calculated based on a normalised curvature $c \in [0, 1]$ where the largest absolute principal curvature is normalised, a breakpoint $B \in [0, 1]$ where the constant influence changes to a squared behaviour,



Curvature for similarity-based average with $k = 2$.



Cruvature for simple averaging.

Resulting parametrization for similarity-based average with $k = 2$.

Resulting parametrization for simple averaging.

Figure 3.7.: This is a comparison between the similarity-based averaging and the simple averaging method. Both use a 2-ring neighbourhood, perform three iterations and have a damping factor of $\lambda = 0.5$. The tow row shows the difference between curvatures. They are very similar but the similarity-based average does not spread high curvature information as quickly (at the base of the feet). The bottom row shows the resulting parametrizations. Figure A.2 in Appendix A shows the meshes extracted from these parametrizations.

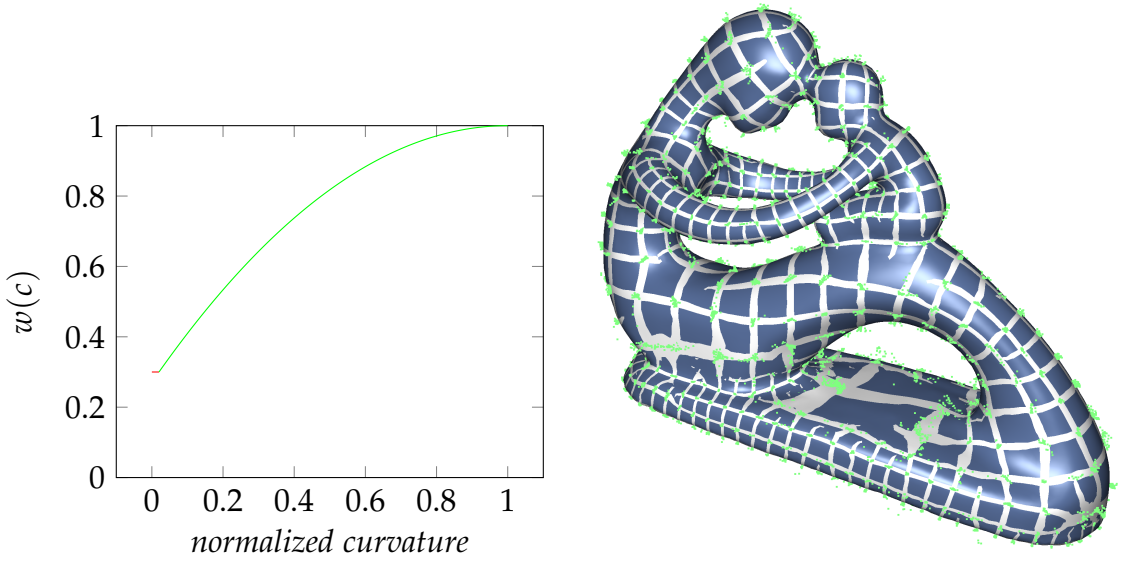


Figure 3.8.: This figure shows the plot of the weights defined in Equation 3.5 for the *expand high curvature* averaging (left) and the resulting parametrization (right). The parameters of the weights are $B = 0.005$ and $C = 0.3$, the damping factor is $\lambda = 0.5$. High curvature regions have more influence and the result has larger dense areas around creases. Unfortunately, because of its dependence on the distribution of the curvature, its effectiveness is not consistent. Figure A.3 in Appendix A shows the extracted mesh.

and the value $C \in [0,1]$ of the constant influence itself. A squared behaviour is favoured over a linear one because this way, rising curvature gains weight more quickly. The weight w is:

$$w(c) = \begin{cases} C & \text{if } c < B \\ 1 - (1 - \frac{c-B}{1-B})^2 * (1 - C) & \text{otherwise} \end{cases} \tag{3.5}$$

Since this weighting uses normalised curvature, it depends on a distribution of curvature which does not have extreme outliers. The curvature estimation of Goldfeather and Interrante, 2004 that is used tends to have bigger outliers which distort the distribution so that most curvature values are in the lower 1%. This means that a curvature of 1% cannot be considered completely flat. To ensure that only very flat curvature has a constant weight, the breakpoint is set to be 0.005. A constant weight of 0.3 works well for the tried meshes. Figure 3.8 shows the plot of the weights and the resulting parametrization.

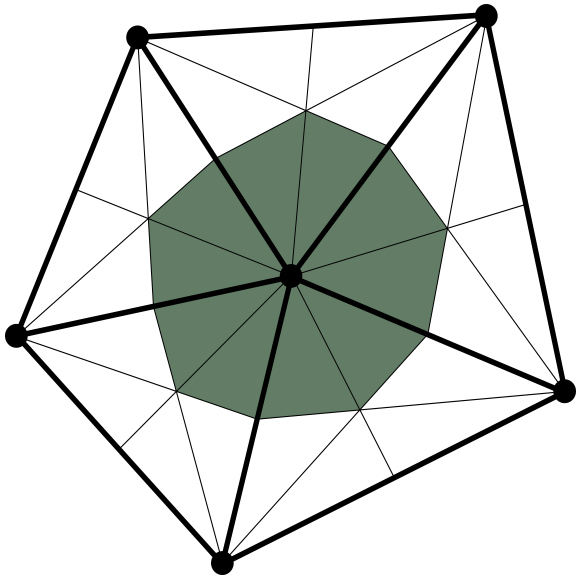


Figure 3.9.: The barycentric cell area for a vertex is defined over all adjacent triangles. It is the sum of areas between the mid-points of the adjacent edges, the centre of mass and the vertex itself, for each triangle.

3.1.4. Hierarchy and Curvature

The method in Jakob et al., 2015 creates a multi-resolution hierarchy of the reference mesh to improve stability and convergence of the orientation and position field estimation (see Section 2.2.3). The lowest level of the hierarchy contains all vertices of the reference mesh and higher levels of the hierarchy merge vertices of the previous level. It can be seen as a binary tree where the leaves represent the reference vertices and parents represent merged vertices. Optimization starts at the highest level of the hierarchy and computed solutions are the initial weights for the next lower level.

Hierarchy Creation: New levels of the hierarchy merge pairs of vertices with weights in descending order. In the work of Jakob et al., 2015 a *normal-based* hierarchy is created. The weights are calculated via the dot product of the normals of the vertices and scaled by the quotient ($\frac{max}{min}$) of their barycentric cells' areas. Figure 3.9 defines the barycentric cell and the areas of merged vertices are the sum of their children's areas. This weighting leads to a hierarchy that primarily merges vertices pointing in the same direction, but not always. Sometimes the hierarchy merges vertices which have less similar normals but a big difference in area to ensure that the hierarchy tree is balanced. An example of what would happen without the scale to balance the hierarchy is shown in Figure 3.10.

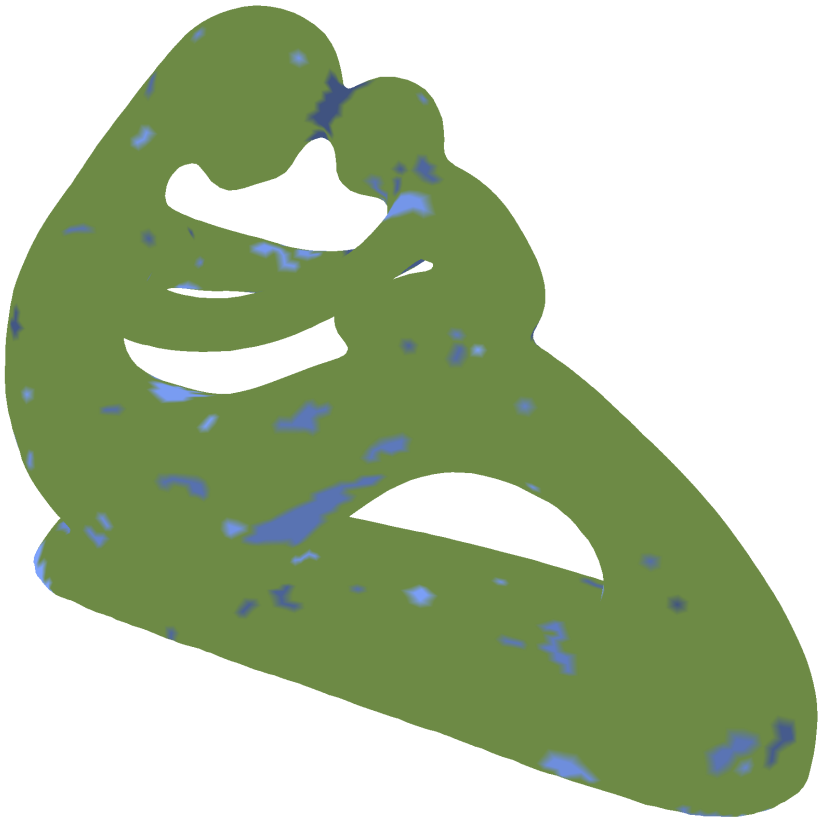


Figure 3.10.: This is a negative example of how the hierarchy would look like without the area-based scaling to balance the hierarchy. Since only similarity and not the size is factored in, the vertices at high levels (displayed here with differing colours) will be of different sizes. Because the hierarchy is used to initialize the next lower levels with the optimised values from higher levels to avoid local minima during optimization, an equal partition is favourable.

A second method for weighting was created, to see whether *curvature similarity-based* merging would be better. This *similarity-based hierarchy* defines the weight for vertex pair (v_i, v_k) as:

$$w(i,k) = -|curv(v_i) - curv(v_k)| \cdot \frac{max(area(v_i), area(v_k))}{min(area(v_i), area(v_k))} \tag{3.6}$$

Where $curv()$ is the curvature to be used (averaged or non-averaged largest absolute principal curvature) and $area()$ is the barycentric cell area of the vertices. As before, vertices can only be merged once per level and the combinations with the highest weight are merged first.

Figure 3.11 shows the main differences between both approaches. Both methods produce similar results because normals and curvature are intertwined and because of the influence of the additional barycentric cell scaling. The *normal-based* construction is able to separate flat areas from non-flat areas further up the hierarchy, however. The *normal-based* hierarchy will be used for all further meshing.

Curvature of Higher Levels: The changes to incorporate a custom scale for each point must now be propagated up the hierarchy as well. To do this, two ways have been explored.

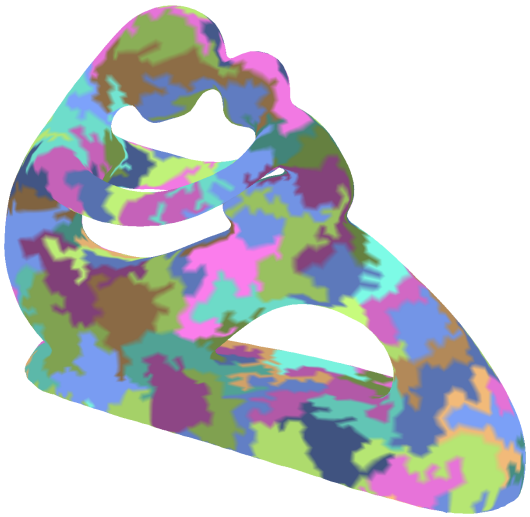
The first method, *curvature summation*, uses curvature information from the next lower level to compute the curvature information as a weighted sum of the merged vertices. The weighting is based on the area of the vertices' barycentric cells. They are the same weights as the ones in the original work to compute the 3-dimensional point of the merged vertices in the hierarchy.

The second method, *curvature estimation per level*, calculates the curvature on each level of the hierarchy based on the points in the 3-dimensional space and their adjacency. This way, the curvature information always represents the current state of the mesh. Since vertices are merged through averages, the higher level mesh might have areas with different curvatures than the reference mesh. Whether the curvature information of higher levels resembles the curvature of the original mesh depends on the weighting that decides which points are supposed to be merged at each level.

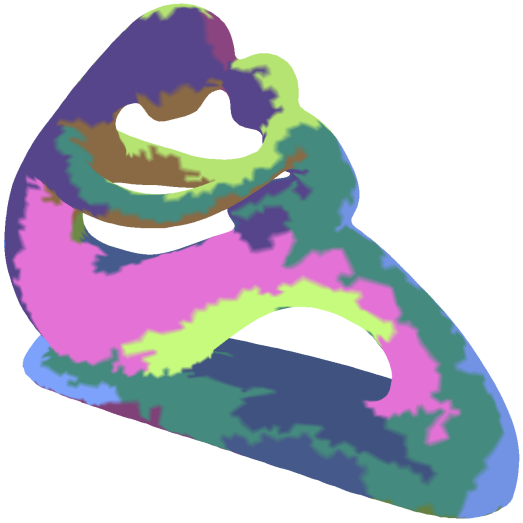
Figure 3.12 shows the different resulting parametrizations of these methods when using the original, *normal-based* construction of the hierarchy. The hierarchy constructions in Figure 3.11 use the second method while Figure 3.13 shows the results for a *curvature similarity-based* hierarchy using the first method The *normal-based* hierarchy construction is not affected by different higher level curvatures.

From the observations made, the original *normal-based* hierarchy creation and curvature estimation at each hierarchy level produce the best results. The additional time needed to estimate the curvature at each level is worth it for the improved precision.

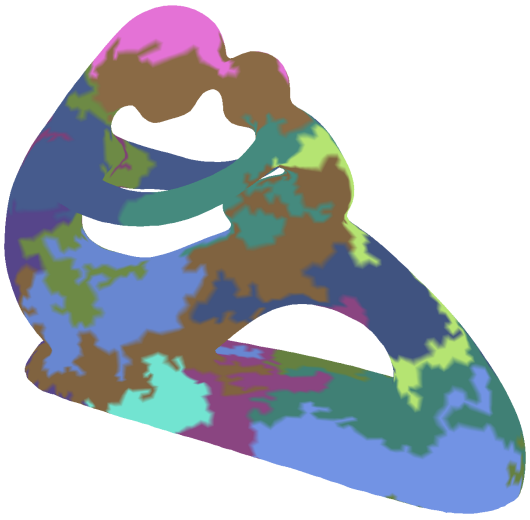
Normal-based hierarchy at level 8.



Curvature similarity-based hierarchy at level 8.

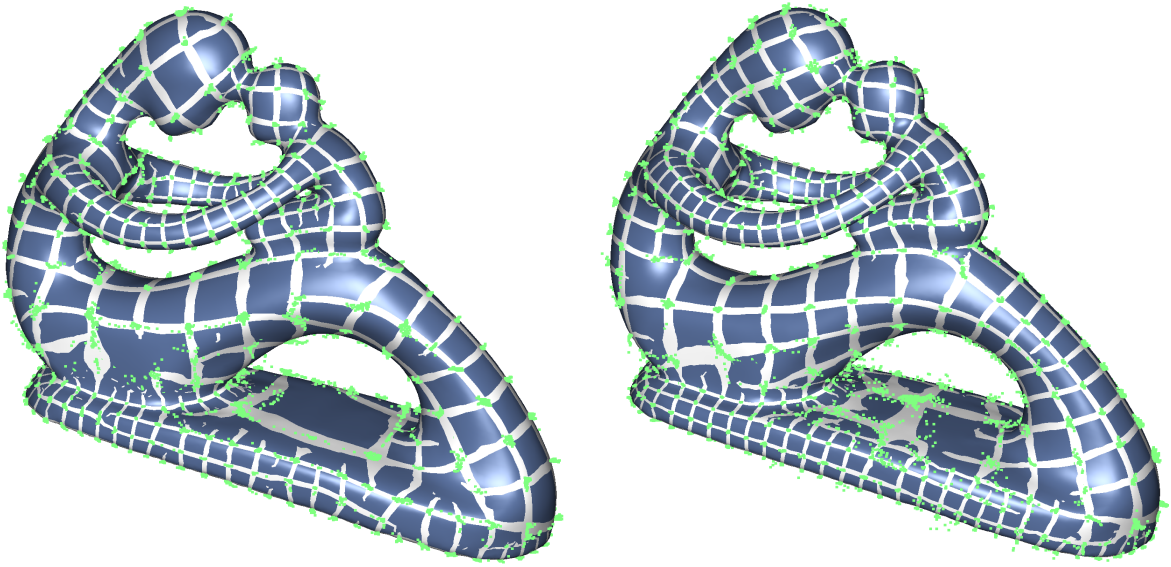


Normal-based hierarchy at level 11.



Curvature similarity-based hierarchy at level 11.

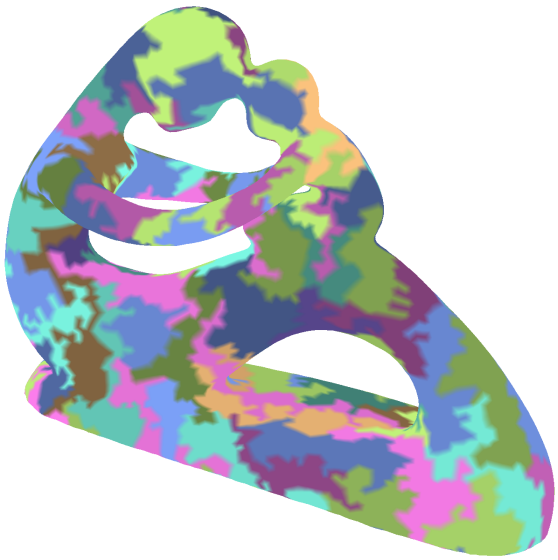
Figure 3.11.: This figure shows the differences between the *normal-based* and the *curvature similarity-based* construction of the hierarchy. The differently coloured patches represent single vertices in the hierarchy at that level. Due to the nature of normals and curvature, the results are very similar. However the normal-based hierarchy keeps flat areas split from non-flat areas until further up the hierarchy.



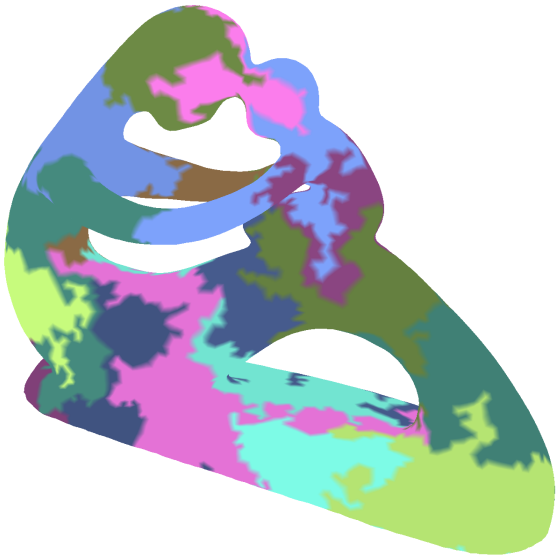
(a) Resulting parametrization for *normal-based* hierarchy and *curvature summation*.

(b) Resulting parametrization for *normal-based* hierarchy and *curvature estimation per level*.

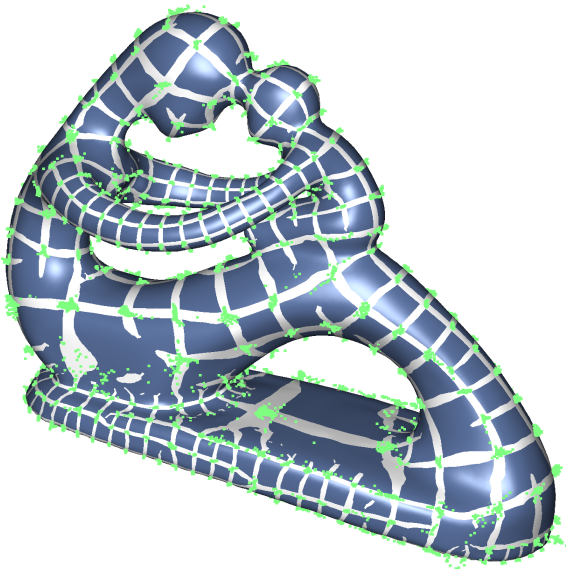
Figure 3.12.: This figure shows the resulting parametrizations for the different methods to define curvature for higher levels of the hierarchy. The differences between them are visible mostly around the base of the statue. When computing curvature as the sum of the merged vertices, the parametrization will create vertices at the edges of the base instead of in the middle. This is the same for the bottom of the statue. This behaviour creates a problem when extracting vertices as the big centre squares will have many vertices at their borders. The extracted meshes can be seen in Figure A.4 of Appendix A.



(a) *Curvature similarity-based* hierarchy with *curvature summation* at level 8.



(b) *Curvature similarity-based* hierarchy with *curvature summation* after 11 merges.



(c) Resulting parametrization for *curvature similarity-based* hierarchy and *curvature summation*.

Figure 3.13.: This shows the hierarchy and the resulting parametrization when using a *curvature similarity-based* hierarchy construction and *curvature summation*. The hierarchy has overlaps at feature lines and the base has a centred quad like in Figure 3.12. Figure A.5 in Appendix A shows the resulting mesh.

3.2. Edge Length from Curvature

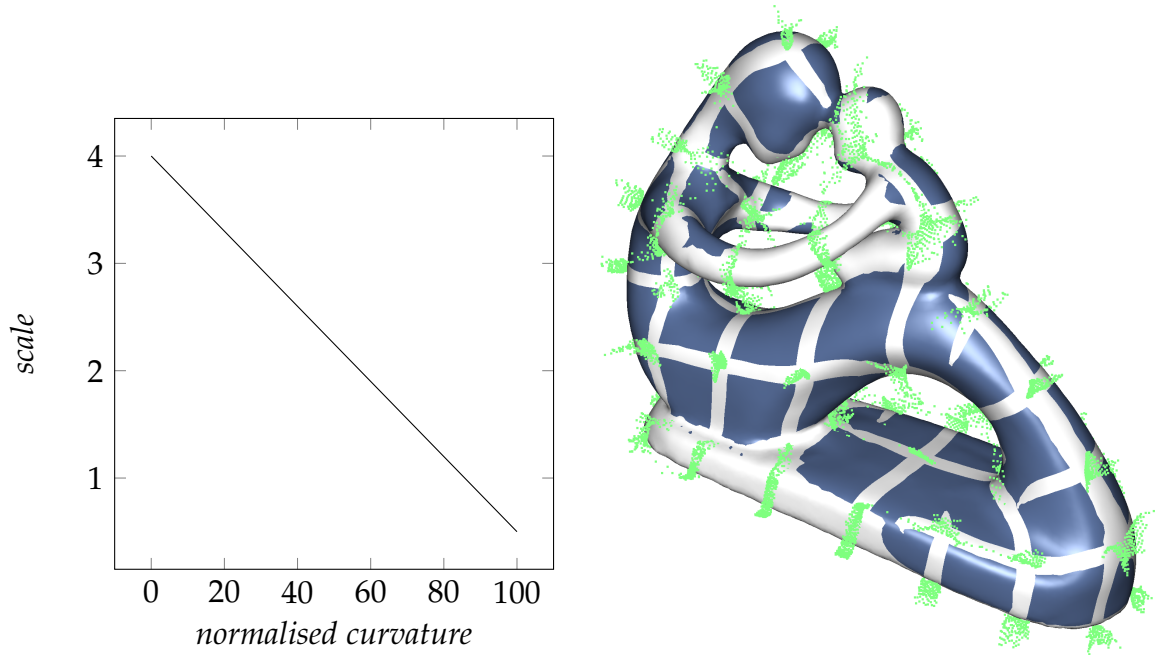
A scale parameter existing in the work of Jakob et al., 2015 is used to define a global desired edge length of the extracted mesh. This scale is influencing the position field calculation and the extraction itself. In the position field, the scale defines the positional invariance points of the field while in the extraction it is used to classify edges and to remove unnecessary edges.

To adjust scale based on curvature, the first step is unravelling the global scale by giving each point its own scale. While the goal of this thesis is an automated extraction of quad-dominant meshes, a manual map from curvature to edge length was developed to study the behaviour of Jakob et al.'s algorithm. Below, a manual map and a direct way to define the desired edge lengths are introduced.

Transfer Function: Transforming curvature to scale is a one-dimensional map and is very similar to a simple transfer function $T(c)$ which maps scalar input to scalar output. They take the normalised curvature c and map them to a scalar. This scalar is then used to scale Jakob et al., 2015's global desired edge length for each vertex. So, a constant transfer function $T(c) = 1$ maps to the uniform edge lengths in Jakob et al's work. Manually defined transfer functions $T(c)$ linearly interpolate between n user-defined two-dimensional value-pairs $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$, which are monotonically increasing in x_i. Their properties are:

$$
\begin{aligned}
& x_i \in [0, 100] \\
& x_0 = 0, x_{n-1} = 100 \\
& y_i \in \mathbb{R}_{\geq 0} \\
& x_i = x_{i+1} \Rightarrow x_j \neq x_i \text{ for } j \neq i, i+1 \\
T(c) &: [0, 100] \to \mathbb{R}_{\geq 0} \\
T(c) &= \begin{cases} y_{n-1} & \text{if } c = 100 \\ y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \cdot (c - x_i) & \text{else } x_i = \max_i \{x_i \leq c < x_{i+1}\} \end{cases}
\end{aligned}
$$

A domain of $[0, 100]$ is used for readability in the transfer function editor and has no influence on the resulting scales. Piece-wise linear transfer functions were chosen because they are very flexible and can approximate any other curve. Steps in the interpolation are explicitly allowed for more flexibility. A simple linear transfer function does not work well because the resulting mesh does not converge to small edge lengths fast enough for high curvatures. This can be seen in Figure 3.14. Better transfer functions, which approximate the edge length estimation through osculating circles (see page 42), produce similar results as the non-approximated version. This can be seen when comparing Figure 3.15 and Figure 3.19. Using steps is dangerous

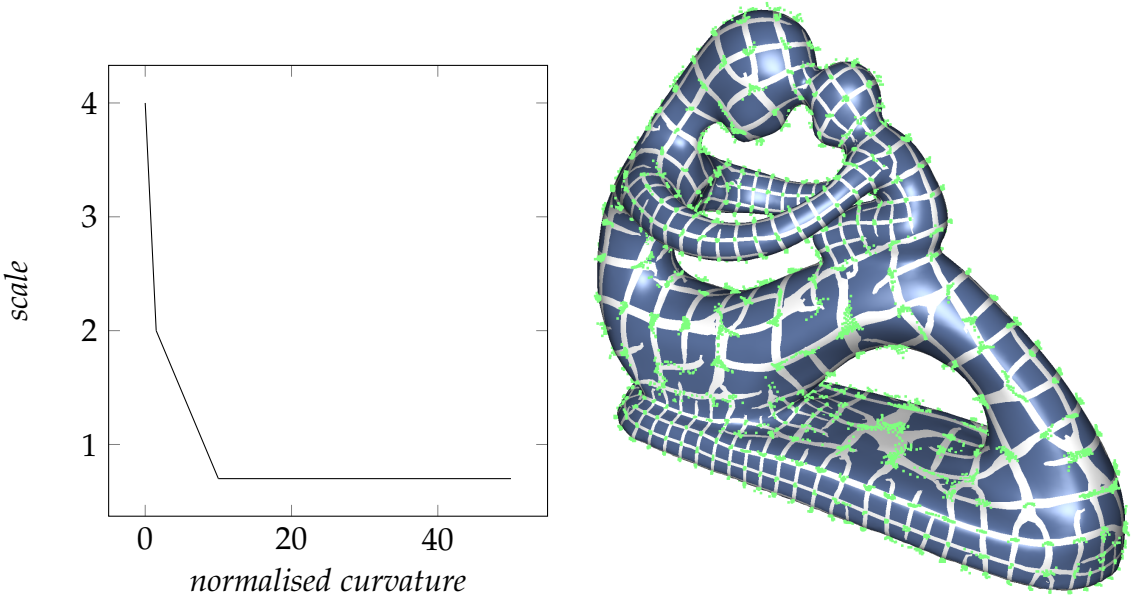


(a) A linear transfer function.

(b) Resulting Parametrization.

Figure 3.14.: This is the plot for a linear transfer function (a) and the resulting parametrization (b). As seen on the arms, edge lengths are too large for curved regions. This can be explained by looking at the estimation of edge length from curvature through osculating circles introduced on page 42, which is not linear. Additionally, the scaling is negatively affecting the position field optimization because the clusters of positions of vertices are more spread apart compared to Figure 3.15.

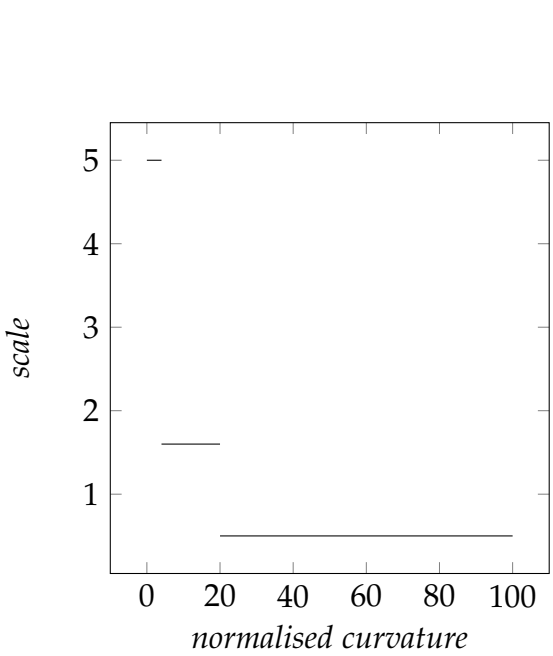
however, because they must be fitted to the reference mesh. An example of a bad transfer function with steps is shown in Figure 3.16. They must be fitted to each reference mesh because the normalisation of curvature depends on the largest estimated curvature.



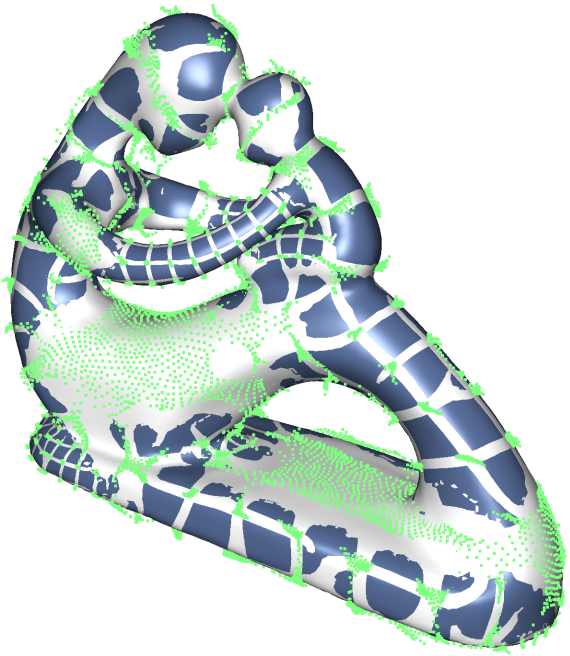
(a) Transfer function approximating osculating circle estimation.

(b) Resulting parametrization.

Figure 3.15.: Shown are the results (b) for a transfer function (a) which linearly approximates the scale estimation from osculating circles (see page 42). Values for the x-axis were set to fit the distribution of the curvature. The results are similar to the ones in Figure 3.19.



(a) A piece-wise constant transfer function.



(b) Resulting parametrization.

Figure 3.16.: Thresholds for edge length estimation are dangerous. While this transfer function may work well for other meshes, it does not work here (b). The flat areas have edge lengths which are too big, smooth transitions of curvature on the reference mesh are not represented in the parametrization and high-curvature areas (e.g. the arms and the base) have edge lengths which vary too much. Thresholding by using only steps in the transfer function is not enough. Steps have to be mixed with linear transitions. Fitting such a transfer function is specific and must be tailored to each reference mesh.

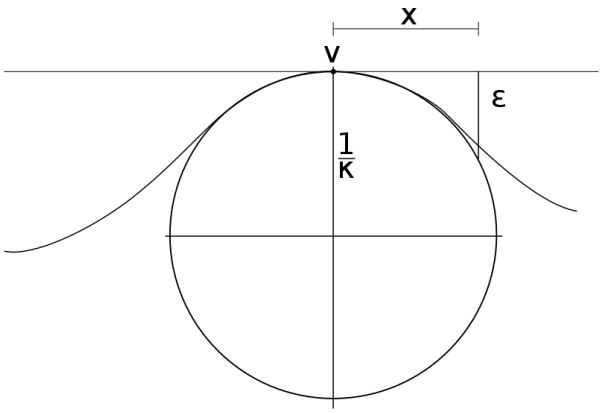


Figure 3.17.: Estimated error ε when radiating out from a vertex v with curvature κ by x.

Edge Length Estimation through the Osculating Circle: Transfer functions are very flexible but the effectiveness of a user-defined transfer function may depend on the meshes. All the information necessary to define a desired edge length for a given vertex which fits the complexity of the underlying surface, is provided by the curvature estimation. Normal curvature is the inverse radius of the osculating circle that approximates a curve lying on the surface at a point. Therefore, the osculating circle provides an error metric. See Figure 3.17 for a visual reference. While this error metric works for any tangential direction, the direction of the largest absolute normal curvature is the best choice. In this direction, the osculating circle is the smallest and will ensure that no detail is lost.

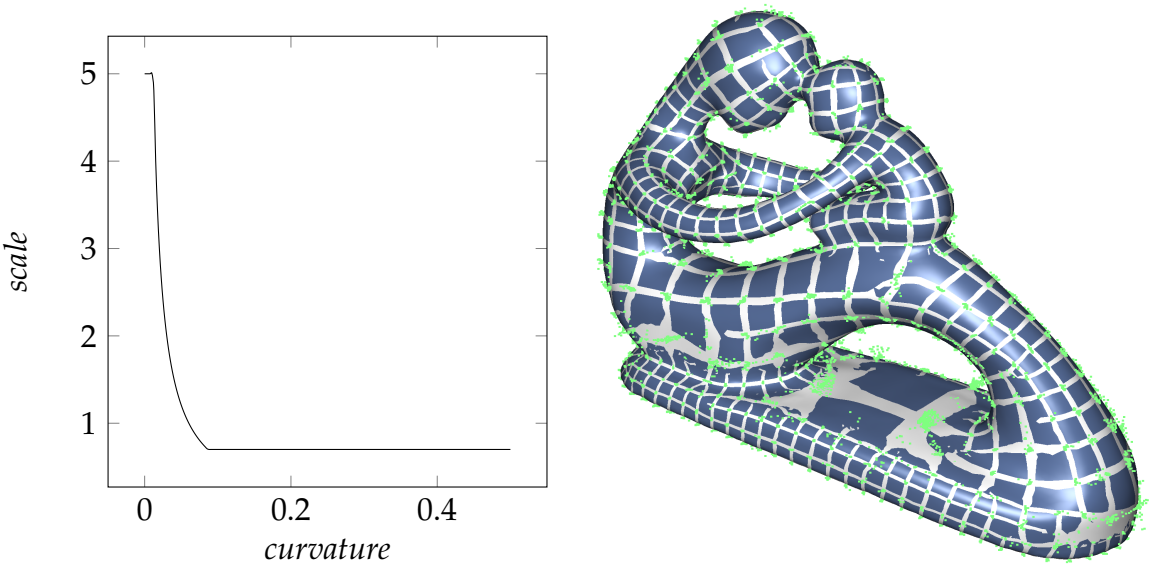
For a point v with curvature κ along a tangential curve on the surface, the osculating circle has radius $\frac{1}{|\kappa|}$. When moving away from the point along this curve by x, the estimated distance to the curve will be denoted as ε. These parameters have the following properties:

$$
\begin{aligned}
r &= \frac{1}{|\kappa|} \\
r^2 &= (r-\varepsilon)^2 + x^2 \\
x^2 &= r^2 - (r-\varepsilon)^2 \\
x^2 &= \frac{1}{|\kappa|^2} - (\frac{1}{|\kappa|} - \varepsilon)^2 \\
x^2 &= \frac{2}{|\kappa|}\varepsilon - \varepsilon^2 \\
\mathbf{x}(\boldsymbol{\varepsilon}) &= \sqrt{\frac{2}{|\kappa|}\varepsilon - \varepsilon^2} && 0 \leq \varepsilon \leq \frac{1}{|\kappa|}
\end{aligned}
$$

$$
\begin{aligned}
x^2 &= \frac{2}{|\kappa|}\varepsilon - \varepsilon^2 \\
0 &= \varepsilon^2 - \frac{2}{|\kappa|}\varepsilon + x^2 \\
\varepsilon(x) &= \frac{1}{|\kappa|} \pm \sqrt{\frac{1}{|\kappa|^2} - x^2} && -\frac{1}{|\kappa|} \leq x \leq \frac{1}{|\kappa|} \\
\boldsymbol{\varepsilon}(\mathbf{x}) &= \frac{1}{|\kappa|} - \sqrt{\frac{1}{|\kappa|^2} - x^2} && -\frac{1}{|\kappa|} \leq x \leq \frac{1}{|\kappa|}
\end{aligned}
$$

When a maximum error, ε, to a curve along a tangent direction on the surface with curvature κ is desired, the estimated edge length can be retrieved from $\mathbf{x}(\boldsymbol{\varepsilon})$ and for an edge length, the estimated distance to the surface is $\boldsymbol{\varepsilon}(\mathbf{x})$.

Two ways of using this derivation to transform curvature to edge length can be used. A simple way is taking an edge length of a (user-defined) factor of the radius of an osculating circle of a curve on the surface. As noted in Section 3.1.2, the best choice is the osculating circle of a curve going through the principal direction with the highest absolute curvature. For a factor of 0.8, this would be the transfer function and result



(a) Representing transfer function. (b) Resulting parametrization.

Figure 3.18.: This figure shows the transfer function (a) derived from an edge length estimation based on a fixed factor 0.8 of the radius of the osculating circle along the largest absolute principal curvature. The resulting parametrization (b) declines to smaller edge lengths more quickly than the method that is using the error metric instead of a fixed factor. Therefore, it produces meshes with a higher density than necessary. Figure A.6 in Appendix A shows the extracted mesh.

given in Figure 3.18 and would yield an estimated error of:

$$\varepsilon(\frac{4}{5|\kappa|}) = \frac{1}{|\kappa|} - \sqrt{\frac{1}{|\kappa|^2} - \frac{16}{25|\kappa|^2}}$$
$$\varepsilon(\frac{1}{2|\kappa|}) = \frac{1}{\kappa} - \sqrt{\frac{9}{25|\kappa|^2}}$$
$$\varepsilon(\frac{1}{2|\kappa|}) = \frac{1}{|\kappa|} - \frac{3}{5|\kappa|}$$
$$\varepsilon(\frac{1}{2|\kappa|}) = \frac{2}{5|\kappa|}$$

Which is a small error for big curvatures and a larger error for smaller curvatures. Because $\frac{1}{\kappa}$ can become large for flat surfaces, the calculated values are bound via an upper and lower bound. These bounds are based on the edge length calculated by the original algorithm, thus, the error does not become too large for flat areas and the edges do not become too small for curved areas. Since the values are bounded for flat areas, the greater error in those areas does not have too much significance.

Another way is to calculate the edge length with the formula for $\mathbf{x}(\varepsilon)$. The error ε is based on the size of the reference mesh. When assigned to an arbitrary value independent of the scale of the reference mesh, the error does not have much meaning and may induce overly big or small quads. So, ε is computed as a (user-defined) fraction of the largest axis of the reference meshes' bounding box. Figure 3.19 shows that this error must be chosen carefully to not lose too much detail.

Alliez et al., 2003 have used a similar estimation that places the constructed edge such that its endpoints are on opposing sides of the circle (i.e. the vertex is its centre). This way, the same error ε will allow an edge length of $2\mathbf{x}(\varepsilon)$. This factor of two is different to just increasing the allowed error (i.e. quadrupling the error in the square root) and, for medium curvature values, produces quads that are too big. Figure 3.20 compares the transfer function equivalents for the method with and without the factor of two, with adjusted errors, and shows exemplary resulting control meshes.

The computed edge length does not prevent the error from exceeding the user-defined error entirely: The edge lengths that are estimated based on the osculating circle are capped to be below five times the global edge length because they tend toward infinity as the curvature approaches zero. Conversely, the edge lengths are capped to be larger than 70% of the global edge length because the optimization of the position field relies on clusters of positions to extract vertices. In areas where the edge length is too small for the density of the reference mesh, the position field can either not form clusters or forms clusters that are too small to extract vertices. If not specified differently by a user, the global edge length is set such that the expected number of extracted vertices is 1/16th of the number of reference vertices (Jakob et al., 2015). This expectation has become less precise due to varying quad sizes but is still a good indicator to ensure that edge lengths do not become too small. Setting the minimal edge length to be between 70% and five times the global edge length has proven to be a good estimate.

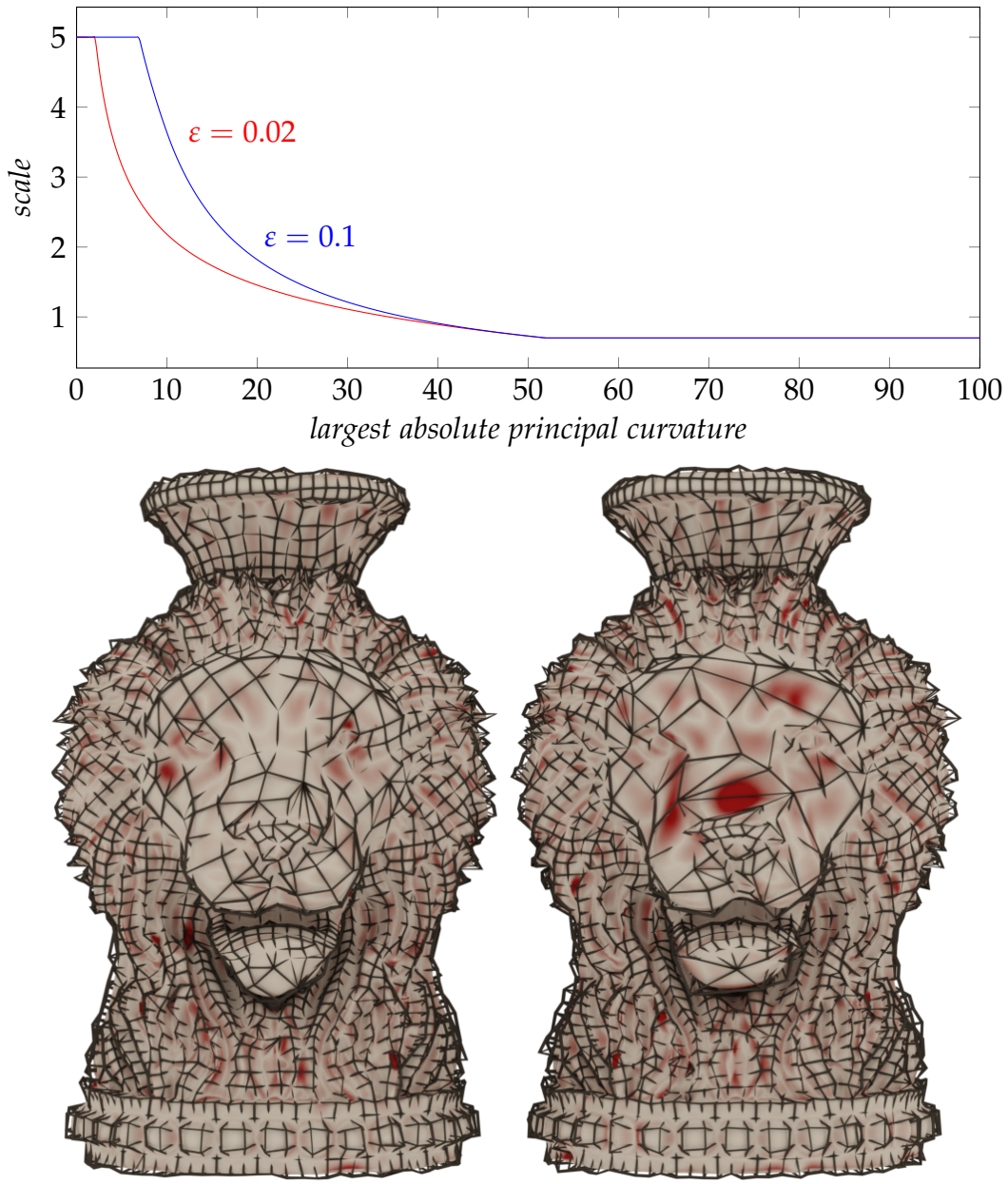


Figure 3.19.: When estimating edge length based on an error, it must be chosen carefully. Here, two different ε(as a percentage of the largest bounding box axis) are used, together with the corresponding subdivision control meshes (and the error for four steps of subdivision). The mesh to the left is computed with $\varepsilon = 0.02$ while the one on the right is computed with $\varepsilon = 0.1$. Both results are similar on the mane of the lion (because edge length is capped at 70% of the global edge length), but the face is better approximated with the smaller allowed error. An error between 0.02 and 0.04 is a good middle-ground between precision and face reduction.

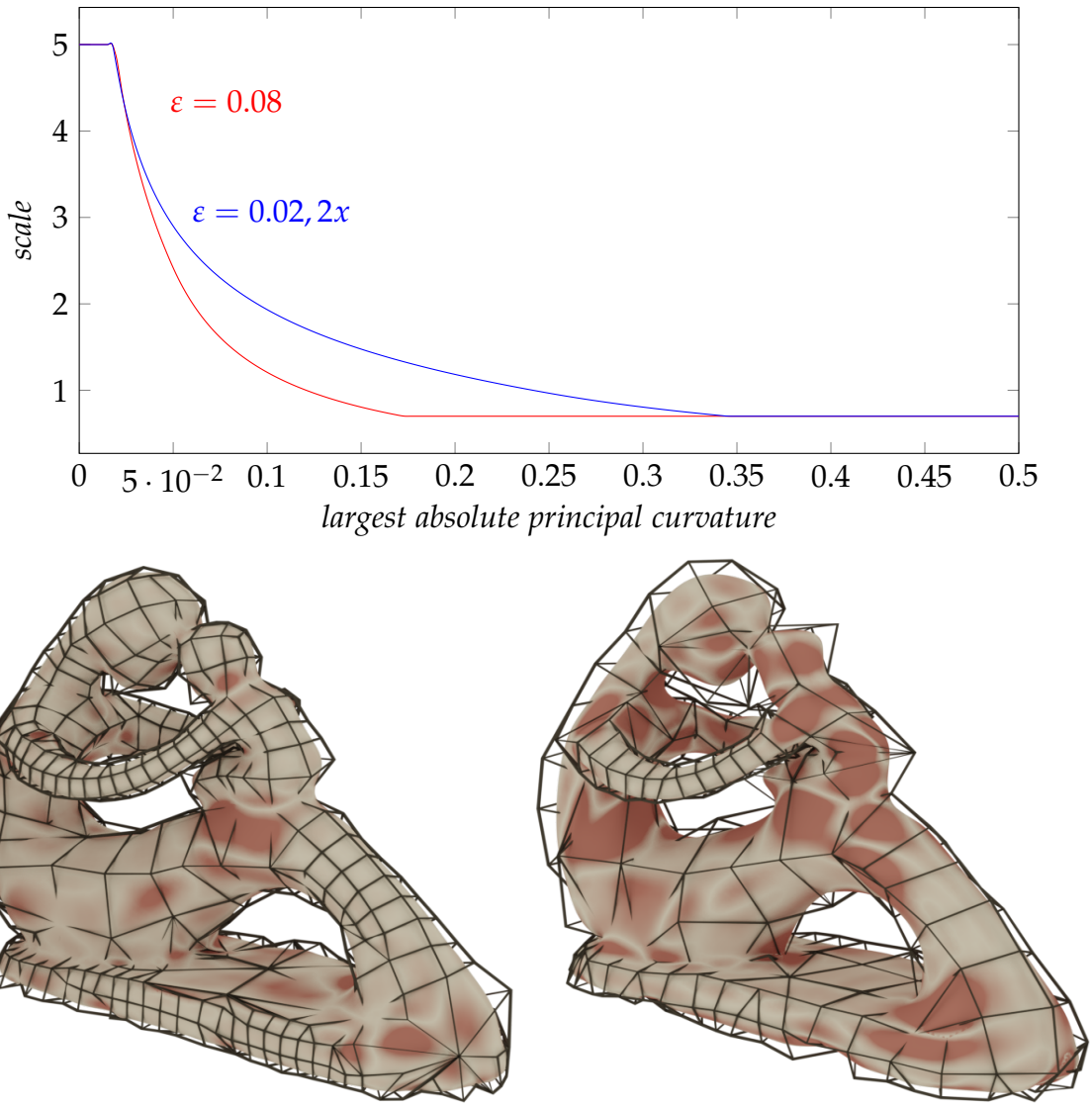


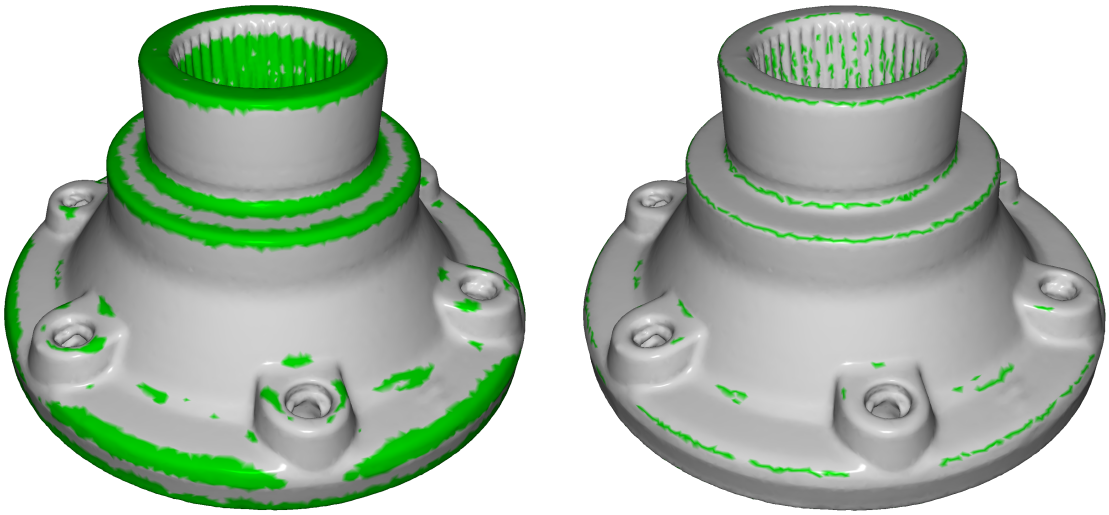
Figure 3.20.: The additional factor of two used in Alliez et al., 2003 (right, blue) decreases the convergence to smaller edge lengths significantly when compared to omitting this factor and increasing the allowed error instead (left, red). The error is increased by a factor of four to balance out the factor of two because the error is inside a square root in the edge length estimation based on the osculating circle. While the behaviour for low curvatures is similar, the slower descent in the middle produces larger quads that are not able to represent the reference mesh as accurately. Depicted are the subdivision control meshes and the error after four steps of subdivision.

3.3. Orientation Field Optimization

While the orientation field computed in Jakob et al., 2015 produces satisfying orientation fields, it is not without flaws. Especially for rotation symmetric objects, the algorithm cannot always direct the orientation field so that the flow lines align with feature lines (e.g. ridges, creases). To combat this, vertices lying on a feature line use their principal directions as a constraint for their orientation. This is what most approaches for quad re-meshing do for the orientation of their mesh. They use highly anisotropic regions of the input mesh as constraints for the following optimization. It is important to note that these constraints have to be used with care because they may disturb/constrain the orientation field optimization too much. Constraints from anisotropic regions of the mesh are used because the principal directions are well defined in these regions.

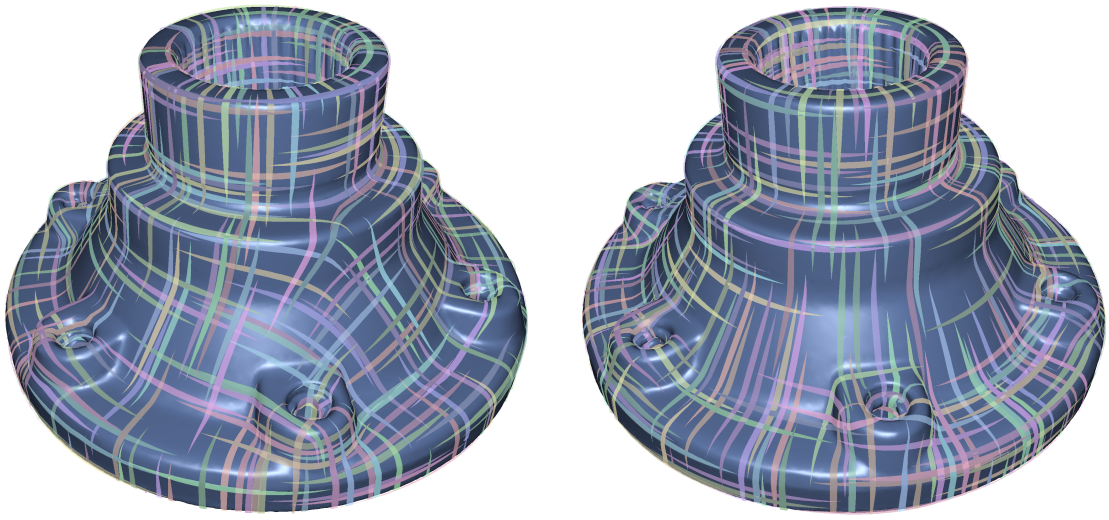
There are more precise methods for feature line detection, but they rely on third order derivatives of the surface. Their precision is not needed since not all vertices which lie on feature lines are needed to constrain the orientation field. To save computation time, a method which does not rely on derivatives to find feature vertices was conceived.

To classify highly anisotropic regions, a threshold for the difference between minimal and maximal curvature needs to be defined and the maximal curvature must exceed some extent to make sure the anisotropic region is on a feature line with high curvature in one direction. Because curvature is not scale-invariant, thresholds based on absolute values for all reference meshes are not possible. Additionally, using thresholds based on percentages of the curvature distribution is dangerous since outliers may disturb the distribution. To classify anisotropy, a scale-invariant quotient κ_3 is used. It is computed from the (smoothed) principal curvatures k_1 and k_2: $\kappa_3 = \frac{min(|\kappa_1|,|\kappa_2|)}{max(|\kappa_1|,|\kappa_2|)}$ (Rugis and Klette, 2006). A threshold for high curvature is defined by the median curvature of all principal curvatures for all vertices. The median of the curvature distribution is less affected by outliers but depends on the shape of the reference mesh and its distribution of curvature. It proved to be stable enough as a measure for high curvature. Candidate vertices for feature lines can now be defined to have $\kappa_3 < 0.1$ and $max(|\kappa_1|, |\kappa_2|) > median$. Only one path of vertices is needed to constrain the orientation along a feature line but the candidate vertices are not limited to a single path yet. To remedy this, only vertices v, which have $max(|\kappa_1^v|, |\kappa_2^v|) > max(|\kappa_1^n|, |\kappa_2^n|)$ for each of its neighbours n and have at least one neighbour which is a feature candidate are true feature vertices. True feature vertices require a feature candidate neighbour to avoid stray feature candidates coming from noise in the curvature estimation. A visualisation of the difference between candidate vertices and true feature vertices can be seen in Figure 3.21. This classification is not perfect, but it suffices to give a minimal set of constraints for the orientation field optimization. The merits of classifying feature vertices and adding constraints to the orientation field optimization can be seen in Figure 3.22.



Feature Candidates | True Feature Vertices

Figure 3.21.: This shows the difference between feature candidates (left) and true feature vertices (right). Since only a minimal set of constraints is desired, only one path of vertices per feature line is necessary.



Original Optimised Orientations | Optimised Orientations With Constraints

Figure 3.22.: Left is the optimised orientation field with no changes and right is the improved orientation field when adding principal directions as constraints for true feature vertices. The original optimization is not able to detect the rotational symmetry of the reference mesh and has a fault in the field where the quads on a ridge do not follow its orientation (notice the cross of flow lines on the ridge). The constrained version does not have the same issue.

3.4. Position Field Optimization

The position field optimization of the algorithm of Jakob et al., 2015 calculates a local parametrization for every vertex where each vertex 'votes' for a position on its grid. The size of the grid tiles for a vertex is defined by its computed scale. A positional vote for each vertex is calculated via a sum of positions of its neighbours. For each neighbour, an optimal position of the vertex and the neighbour is calculated based on the grids spanned by both. The vertex position for the next neighbour is now a weighted sum of both computed optimal position. In the end, the computed position - which may not lie on the grid lines of the vertex - is rounded to lie on the closest point of the vertex' grid. For a parametrization where edge length is not global, scales and grid sizes must differ from vertex to vertex. The following paragraphs will describe three ways of influencing the grid sizes according to the scales.

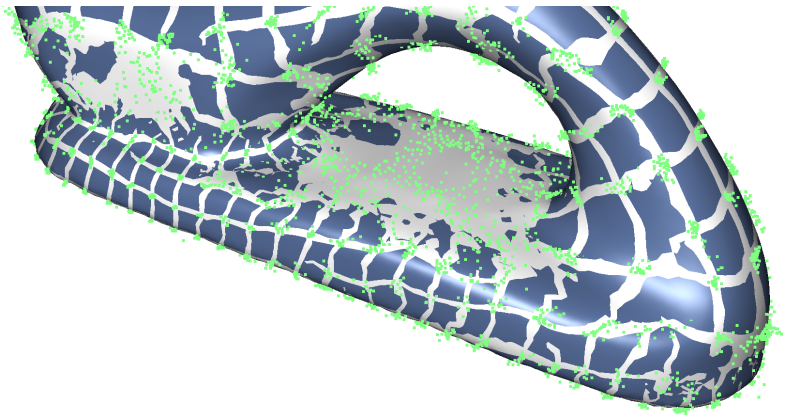
Average: The first approach was averaging the scale parameter to use the original scaling mechanism. For each neighbouring pair, the grid sizes are now a simple arithmetic mean of the two scales. In areas with similar curvatures, this mean is equal to the original algorithms' behaviour. When the curvature changes drastically, however, scales can differ greatly for different neighbours. This is not ideal because the optimization then tries to find an optimal position based on a grid with differently sized tiles for each neighbour. The result is that the local parametrizations of neighbouring vertices will not find concise clusters of 'votes' from which a vertex may be extracted.

Reference Point: As a second method, to keep the grid size more stable, only the grid size of the reference point is used when computing optimal positions with all neighbours. This way, neighbours will not contribute their curvature based scaling to the reference points' optimal position. This achieves much more concise clusters of 'voting' points on the grid. However, it may still be improved.

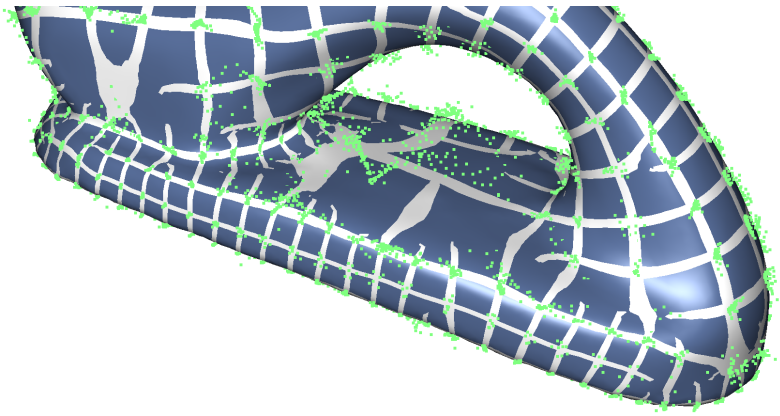
Per-Vertex: As a final version, the grids for neighbouring vertices may have different sizes when summing over all neighbours. Now, each point will always use its own grid when computing optimised points with neighbours. This means that the reference point and the neighbour use their differently sized grids when searching for optimal positions on the grid. All vertices now compute their optimal grid positions (be it as a reference point or as a neighbour) based on the same grid size.

Results of these variations can be seen in Figure 3.23. As mentioned, averaging the grid positions results in less concise clusters of 'voting' points on the virtual grid in comparison with the other approaches.

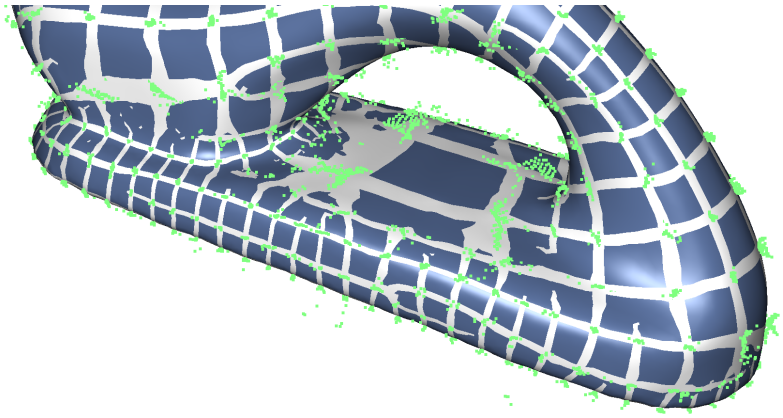
Number of Iterations: Due to the increased complexity of differing grid sizes, the number of iterations on each level of the hierarchy may be changed manually. Jakob et al. recommended six iterations per level in their work. Observations show that 10-15 iterations are better for consistent results.



(a) Arithmetic mean scaling.



(b) Reference point only scaling.



(c) Per-vertex scaling.

Figure 3.23.: This displays the behaviour of the different scaling methods for position field. It can be seen that clusters form more strongly for methods (b) and (c). The extracted meshes can be seen in Figure A.7 in Appendix A

3.5. Mesh Extraction

The extraction of the final mesh is done in two stages: extracting a graph of vertices and their neighbourhood and then extracting a set of quad-dominant faces from this neighbourhood. The following three changes were made to the mesh extraction mechanism.

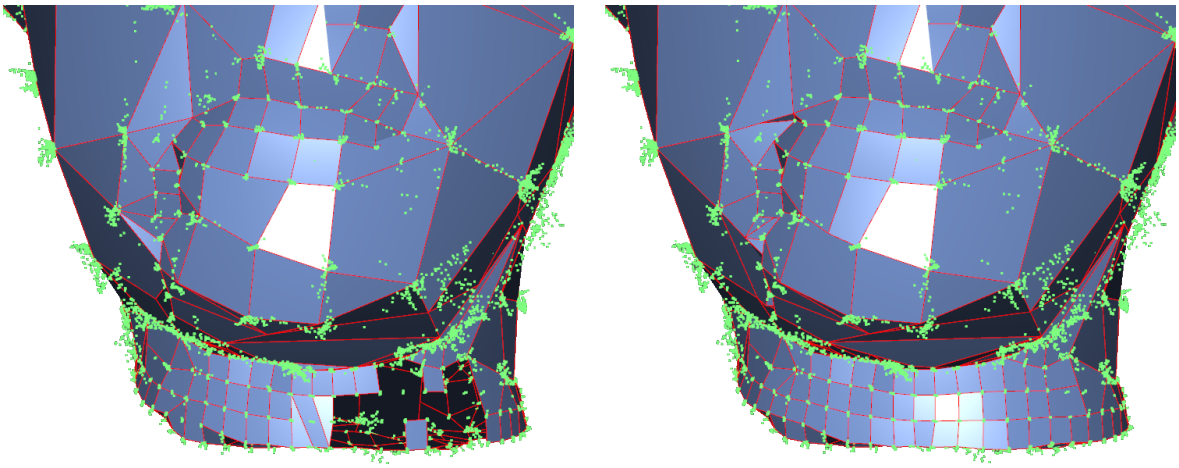
3.5.1. Minimal Vertex Cluster Size

In Jakob et al.'s work, extracted vertices are discarded when the size of their cluster in the parametrization is smaller than one-tenth of the average cluster of all extracted vertices. This is done to avoid stray extracted vertices. For most meshes, one-tenth is appropriate. However, when the extracted mesh has large faces, the clusters of their vertices may distort the average such that vertex clusters are discarded that would still be needed. To give control over this behaviour, an option was added to change the fraction of the cluster size below which vertices are discarded. An example for this behaviour is shown in Figure 3.24 where a minimal cluster size of one-tenth of the average is too large to keep small faces on high curvature areas of the reference mesh.

3.5.2. Snapping Thin Triangles

When extracting vertices, Jakob et al.'s method sometimes produces thin triangles. These triangles are addressed by changing their neighbourhood and positioning so they form a line instead of a triangle. In the original work, three neighbouring vertices are considered a thin triangle if any height of the triangle is below 30% of the global scale. This scale defines the global edge length. However, since the scale is not global anymore, this threshold does not work and each extracted vertex needs its own scale. The new scales for the extracted vertices are based on the scales of the vertices which contribute to their position in the parametrization. They are computed as a weighted sum over all scales of contributing vertices, using the same weights as the computation of the extracted vertex positions. A new threshold for thin triangles can now be defined to be the sum of 10% of each extracted vertex' scale. It is equivalent to the one used by Jakob et al., 2015 when all vertices have the same scale.

If a thin triangle is encountered, the vertex from which the height was calculated is moved. If this vertex is close to one of the other two vertices of the triangle (i.e. its distance is smaller than the height threshold) then these two are merged and their position is averaged. When the vertex to be moved lies between the other two



Minimal cluster size is too big. Better minimal cluster size.

Figure 3.24.: Vertices are not extracted when the size of their cluster is below a fraction of the average cluster size. When mesh densities differ greatly, the average cluster size is influenced by vertices with big clusters (i.e. the clusters of vertices next to big quads). When this happens, the minimal cluster size must be set carefully. Left shows the result of extraction with a minimal cluster size of 1/10 of the average cluster size. When set to 1/20 (right), the small quads at the bottom are not removed.

vertices (i.e. it is not close to any of them) it is moved so that all three are collinear. Unfortunately, this simple threshold is sometimes moving vertices away from feature lines of the reference mesh. This decreases the quality of the meshing because sharp features cannot be represented when no vertices lie on them.

The snapping is performed at a point in time where only the adjacency between vertices is known. This reduces the effort of maintaining mesh information (there are no edges or faces yet) but limits the available information to prevent bad snapping. Computing curvature is not advisable because the neighbourhoods may be non-manifold and the curvature may change while snapping. Since extracted vertices are created through clusters of reference vertices, the number of feature vertices (introduced in Section 3.3) and the average principal curvature of their clusters can be computed. Additionally, from the oriented adjacency information, normal vectors of potential faces can be computed by iterating over all adjacent pairs of neighbours. If a pair of neighbours forms a nearly straight line (less than 10 degrees), the normal is computed with the next edge of the potential face to increase accuracy. When any two normals of a vertex have more than 45 degrees, the vertex is considered to be a feature vertex.

To prevent bad snapping, the position of merged vertices is set to be the position of the vertex with the greater principal curvature (in any direction) instead of computing an averaged position. Collinear snapping is only performed if the vertex to be moved is not a feature vertex and its number of feature vertices is smaller than two or all three vertices are feature vertices with more than one feature vertex. Figure 3.25 shows a mesh with no snapping, with the original snapping and with the improved snapping.

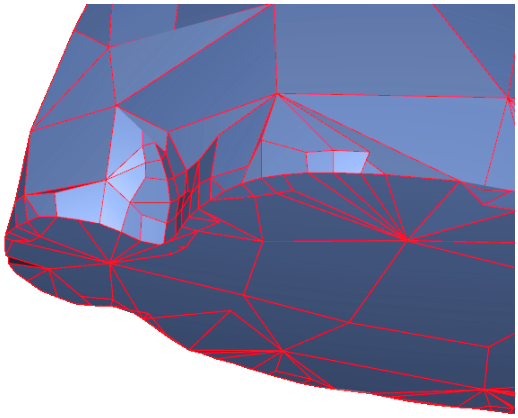
3.5.3. Creating Quads

When extracting faces, the algorithm searches for polygons in the adjacency graph of the extracted vertices. First, it searches for quads, then for triangles, then for polygons with up to 8 vertices, in increasing order. After this is done, the mesh might contain holes which are then filled. Creating faces is the same for the search of polygons and for filling holes: A face is created if the polygon is a triangle, a quad or a pentagon. n-gons for $n > 5$ are filled with quads until the remainder of the hole can be filled with a triangle, quad or pentagon. Jakob et al. search for and fill holes with up to 8 vertices. Because it is important for this thesis that no holes are left, the maximum number of vertices is increased to 30 (which was big enough while testing). The polygon search maximum remains at 8 vertices because missed polygons this size (which are very rare) are found by the hole-filling. Selecting quads for filling these n-gons follows a greedy scoring system. In the original paper, for an n-gon $v_0, v_1, ..., v_{n-1}$, the score of quad $v_i, v_{i+1 \pmod n}, v_{i+2 \pmod n}, v_{i+3 \pmod n}$ is calculated as a sum of the four next angles' differences to a right angle, starting from i:

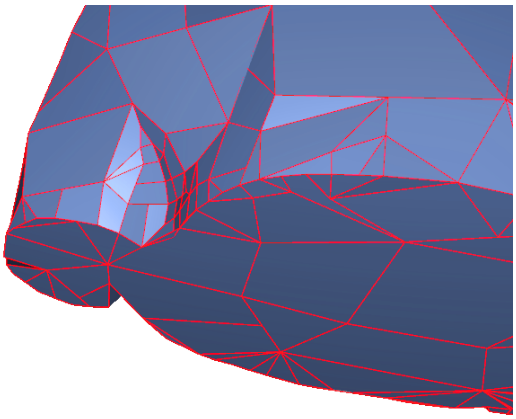
$$score_i = \sum_{j=0}^{3} \left| 90^\circ - \measuredangle \begin{matrix} v_{i+j \pmod n} \\ v_{i+j+1 \pmod n} \\ v_{i+j+2 \pmod n} \end{matrix} \right|$$

In Jakob et al., 2015, no reasoning for this scoring is given and it does not try to find perfect quads as only the first two angles (i.e. $j = 0, j = 1$) are part of the quad. Figure 3.26 gives a visual representation of this scoring system. To favour optimal quads, a new scoring system which does not use four consecutive angles but the four angles of the quad in question is used. The score for the same quad as before is defined as the following:

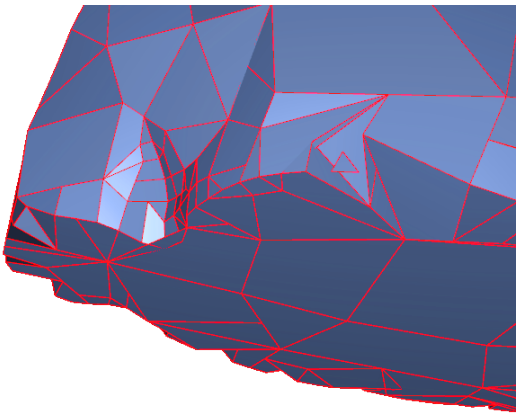
$$score_i = \sum_{j=0}^{3} \left| 90^\circ - \measuredangle \begin{matrix} v_{i+j \pmod n} \\ v_{i+(j+1 \pmod 4) \pmod n} \\ v_{i+(j+2 \pmod 4) \pmod n} \end{matrix} \right|$$



No snapping.



New snapping.



Original snapping.

Figure 3.25.: When no snapping (top) is performed, more thin triangles are left in the mesh. When large and small edge lengths come together, the snapping (bottom) also reduces the number of small faces on the mesh because the larger edge lengths increase the threshold for thin triangles. This is visible at the ridge in the picture because the snapping removes the small quads. The differences in snapping between the new method (bottom left) and the original (bottom right) are exaggerated in feature lines with flat sides. The original snapping moves vertices away from the shown ridge such that face diagonals 'cut' into the ridge. This is avoided by the new method. Quads that still have diagonals 'cutting' into the ridge will be aligned properly during post-processing (see Subsection 3.7.2).

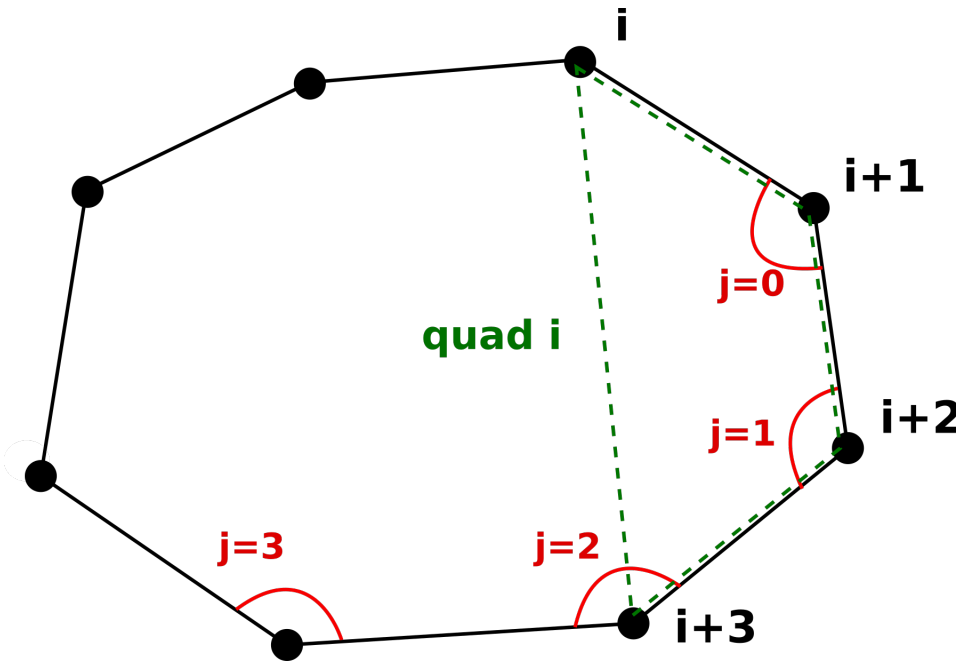


Figure 3.26.: This figure shows which angles are considered for the original scoring of quads when extracting the mesh.

This scoring now fills the space with the most rectangular quad first. Figure 3.27 gives a visual representation of this new scoring system. Observations on the behaviour also suggest that the best quads are also well aligned with the principal directions and that no additional consideration for this alignment is necessary. Figure 3.28 shows the different behaviours of these two methods.

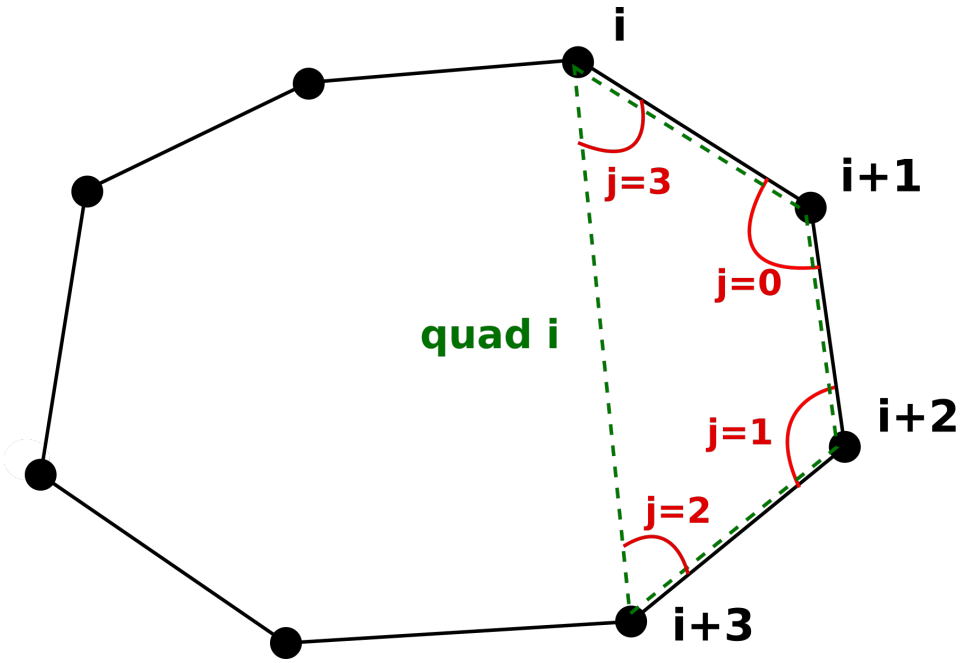


Figure 3.27.: This figure shows which angles are considered for the new scoring of quads when extracting the mesh.

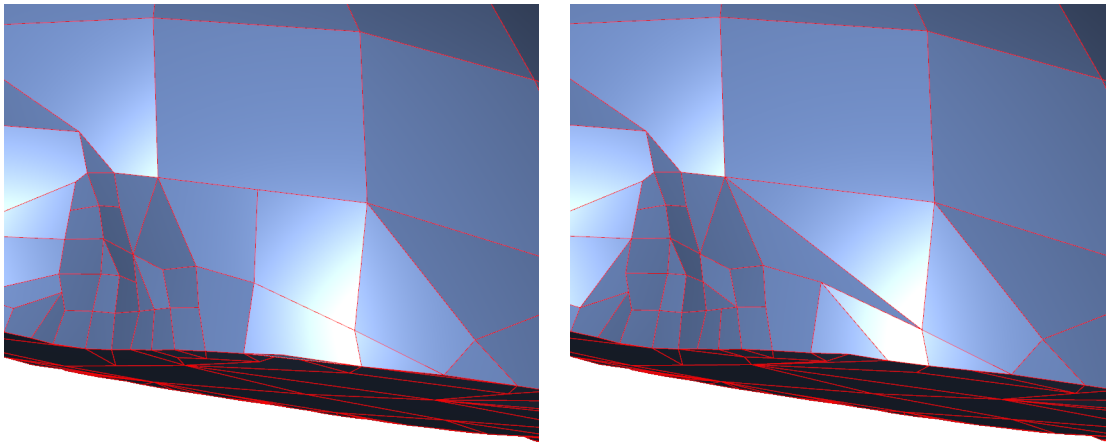


Figure 3.28.: This figure shows the different results stemming from the two quad scoring methods when creating the faces of the extracted mesh. Left is the new scoring and right is the scoring by Jakob et al., 2015. The main difference is visible on the big face in the centre where the original creates two distorted quads and the new scoring creates more regular quads. The new scoring has a better average quad quality (see 4 for more detail on this quality measure).

3.6. Non-Manifold Clean-Up

After the extraction is done, the mesh must be cleaned. Because of the local nature of the position field optimization and the greedy face creation, extracting a manifold is not always possible. To remove the non-manifold parts of the mesh, Jakob et al. have developed a simple algorithm. Their algorithm performs two passes where faces introducing non-manifold edges and then all faces surrounding non-manifold vertices are deleted. Since the goal of this thesis is a closed mesh, Jakob et al.'s approach is not enough as deleting affected faces results in holes. The method of Jakob et al. was changed to repair non-manifoldness instead of deleting it. Non-manifold normals (i.e. neighbouring faces with opposite normals) are not created during the extraction and do not need to be handled.

Non-Manifold Edges: Non-manifold edges can be detected in the half-edge data-structure used to represent the extracted mesh when the same directed edge is introduced by two faces. Removal of these edges is done by changing the affected faces or by removing one face (as a last resort). If a non-manifold edge involves more than two faces (very rare), the faces are detected iteratively such that only two faces have to be considered at a time. Manifoldness is ensured for arbitrary triangulations of the quad-dominant mesh and not just for the mesh itself. This means that all diagonals can be non-manifold as well. The method to repair a non-manifold edge is as follows: delete all vertices in both faces which are not part of the non-manifold edge and are adjacent to only one or two faces in total. If no vertex is deleted, delete one face as a last resort (if the non-manifold edge is a diagonal on one face, delete this face - otherwise delete a random face). When the deleted vertices leave faces with 2 or fewer vertices, they are deleted as well. Figure 3.29 and Figure 3.30 show the most common types of non-manifoldness and how the cleaned solutions look like.

Non-Manifold Vertices: Non-manifold vertices can be detected by creating a list of all faces adjacent to the vertex, starting at a random neighbouring face, traversing the neighbourhood of the vertex and removing all encountered faces off the list. If the vertex is non-manifold, it has more than one neighbouring face loop. By starting at a random face and traversing the local neighbourhood, only faces on the same face loop as the starting face are removed from the list. So, if the list is not empty after all faces were visited, the vertex is non-manifold. To find all loops, this loop-search is repeated until all neighbouring faces have been assigned to a face loop. For each additional face loop, a new vertex is created and all instances of the non-manifold vertex are positioned to be the average of all adjacent vertices in their respective face loop.

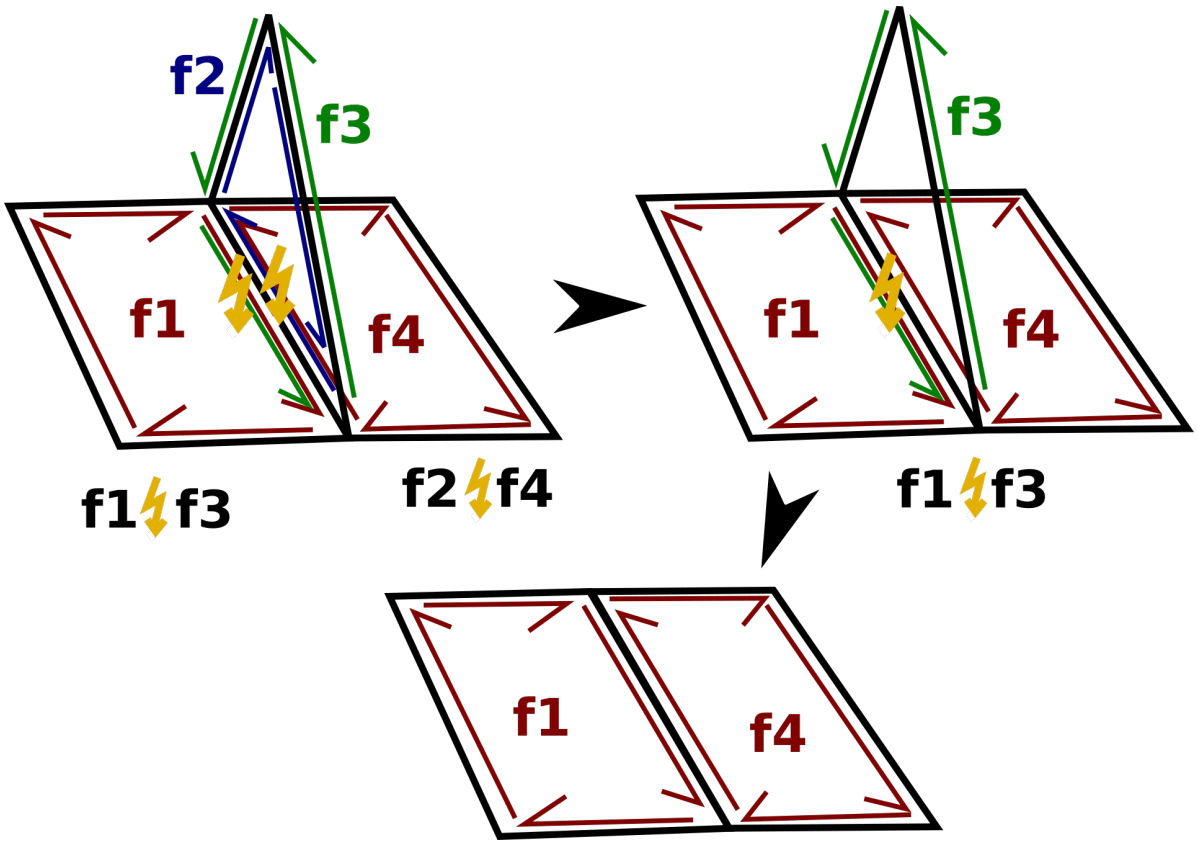


Figure 3.29.: In this case, two triangles ($f2$, $f3$) disturb the manifoldness of the surrounding mesh. Two half-edges are non-manifold here because both triangles affect different ones. Since the edges are handled in succession, two steps are needed to resolve the issue. The first pair of faces is $f2$ and $f1$. The vertex of $f2$ not on the edge has only two adjacent faces and is deleted, which prompts the deletion of $f2$ because only two vertices remain. Face $f1$ does not have vertices that need deletion. The same process is performed for $f3$ and $f4$, with the only difference being that the vertex to be deleted has only one adjacent face.

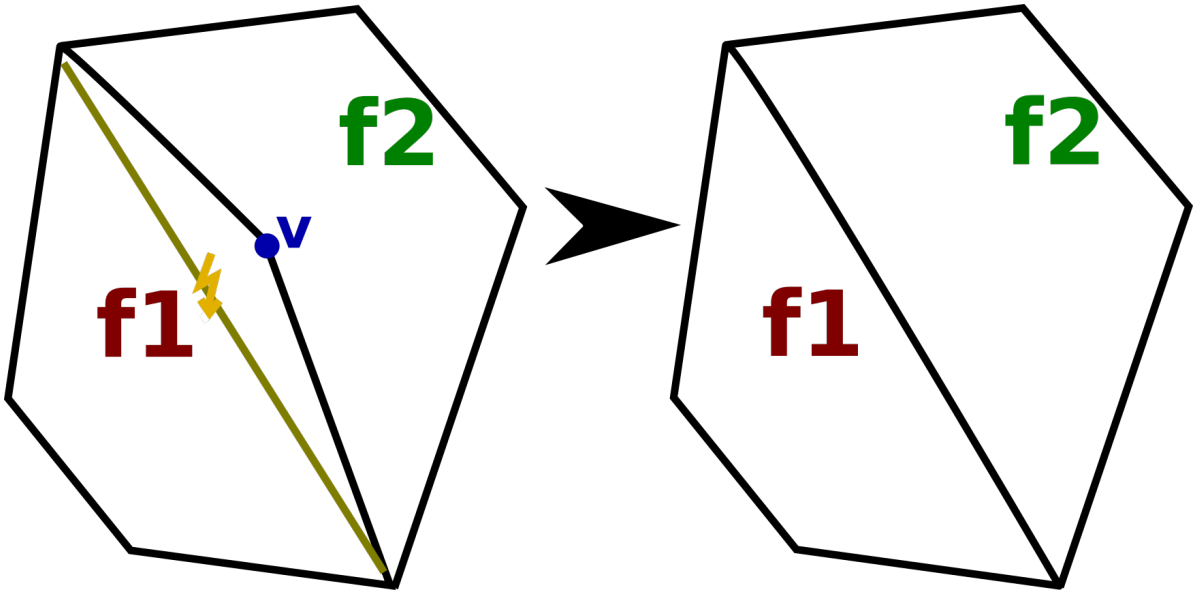


Figure 3.30.: Here, two 5-gons share vertex v which has adjacency to only these two faces. This leads to a non-manifold diagonal, which would cause non-manifold edges for some triangulations. After deleting these vertices, the non-manifoldness is resolved. When this non-manifoldness occurs, 5-gons are reduced to quads, which is favourable.

While this method removes all non-manifold elements, it cannot guarantee that no holes are created. This is because there are a few cases where faces cannot be changed and must be deleted to ensure manifoldness. Since a closed mesh is the goal, the hole-filling algorithm from the extraction (see Section 3.5) is used to find and patch potential holes. To ensure that the hole-filling does not introduce non-manifoldness, one more iteration of non-manifold clean-up is performed. Any holes that persist are considered too complex to fill and are not patched.

3.7. Mesh Post-Processing

After the non-manifold clean-up is performed, the extracted mesh still has potential for improvement. The goal when extracting the mesh is an accurate representation of the reference mesh while minimising the number of faces and vertices needed. All post-processing is performed directly on the data-structure representing the mesh. This means that half-edge information must be kept up to date for each step. This increases the effort for development but decreases the run-time of the algorithms. The following three additional passes for optimization are performed to improve both metrics.

3.7.1. Flat T-Vertices

All T-vertices in the extracted mesh induce extraordinary vertices in the subdivision control mesh. These T-vertices may be removed by changing two of the neighbouring faces and removing the T-vertex from the third one. To prevent the loss of detail, the two changed faces must lie on similar planes and their edges adjacent to the vertex must lie on similar lines. For the purposes of this clean-up, two faces lie on similar planes when their normals differ by less than five degrees. Two outgoing edges from the T-vertex lie on similar lines when their angles are between 175 and 185 degrees. These angles have shown to be a fitting threshold. Figure 3.31 shows the different combinations of faces and their resolved topology. The size of the third face does not matter as it is just reduced by one vertex (triangles reduced by one vertex are lines and can be removed). In all cases where a T-vertex is cleaned, one non-quad is removed and the reduced face may introduce a non-quad or become a quad. Most removed T-vertices have 5-gons on one side and two flat faces on the other side. If it was not a 5-gon it would be a very thin triangle or a distorted quad which the position field optimization and extraction try to avoid. Because of this, the clean-up tends to reduce the number of non-quads. Additionally, the number of extraordinary vertices in the subdivision control mesh is reduced by one.

3.7.2. Feature-Aligned Quad Edges

Mis-aligned quad edges occur when quads along a feature line are rotated by approximately 45 degrees. This means that one of its diagonals follows the feature line instead of one of its edges. To preserve edge paths along these lines, two different ways of dealing with the quads are used: remove them by merging both endpoints of the diagonal on the feature line and putting the merged point on the middle of the diagonal, or splitting them into two triangles. Removing quads, instead of splitting them, does not introduce two non-quads but it must be done with caution, because the quad may lie on a bent part of the feature line and removing it could increase

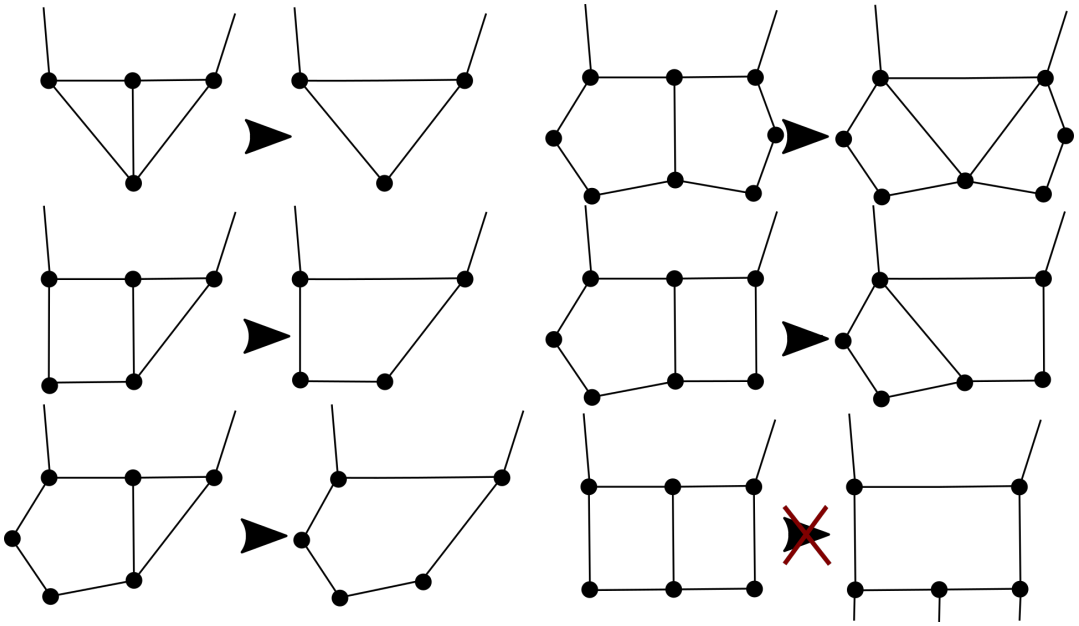


Figure 3.31.: These are all the cases for T-vertex removal. The faces are changed such that the mesh still only contains triangles, quads and 5-gons. Not all cases improve the quality of the mesh: combining two quads (third right) is not feasible because this would add a non-quad and might just shift the clean-up to the two faces below, thus creating more non-quads.

the error to the reference mesh. Hence, only quads where the diagonal has a pair of adjacent edges which lie on a common line are removed, and all others are split into two triangles. A pair of outgoing edges from both ends of the diagonal is considered to be on the same line as the diagonal when both of their angles to the diagonal are less than 10 degrees. This angular threshold was determined empirically through observations over all compared meshes. Figure 3.32 shows how detail is lost when a quad is wrongfully removed instead of split.

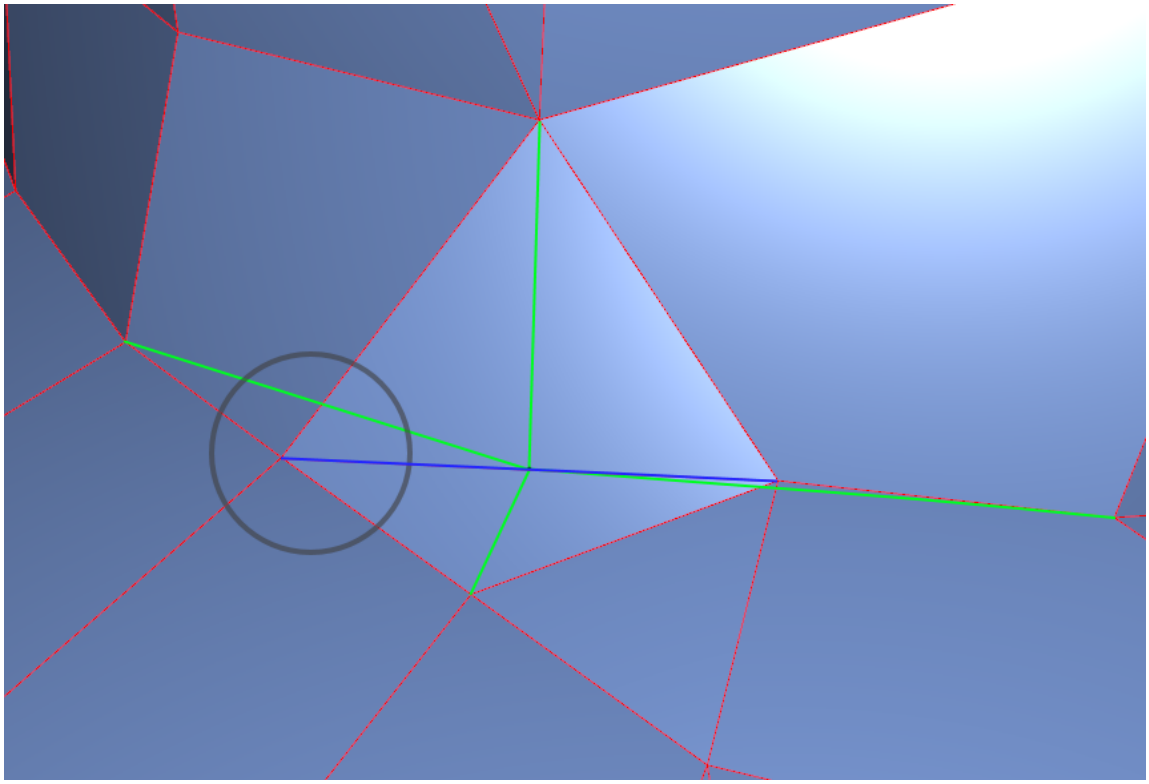


Figure 3.32.: The shown quad has no adjacent edges which lie on a common line and should be split instead of removed. The blue line shows the topology after a split, while the green lines show the topology after removing the quad by merging the two end-vertices of the blue line. The grey circle shows why removing the quad loses detail in the extracted mesh. The corner is gone and the green line cuts into the shape described before the quad removal.

Finding these quads is a delicate task because false-positives must be avoided. The most intuitive criteria are that the diagonal lies along the orientation of the two end-vertices, that one diagonal is much farther away from the reference mesh (the diagonal not on the feature line 'cuts' into it), and that the quad is bent strongly along the diagonal on the feature line. The following criteria produce the desired classification of mis-aligned quads:

- The average angle between the orientation field of both end-vertices and the diagonal (with 90-degree rotation symmetry) must be smaller than 15 degrees.
- The angle of the two triangles on both sides of the diagonal on the feature line must be greater than 50 degrees.
- The average largest principal curvature of both end-vertices of the diagonal must be greater than the one of the other diagonal.
- The closest face of the reference mesh must be closer to the centre of the diagonal than the centre of the other diagonal.

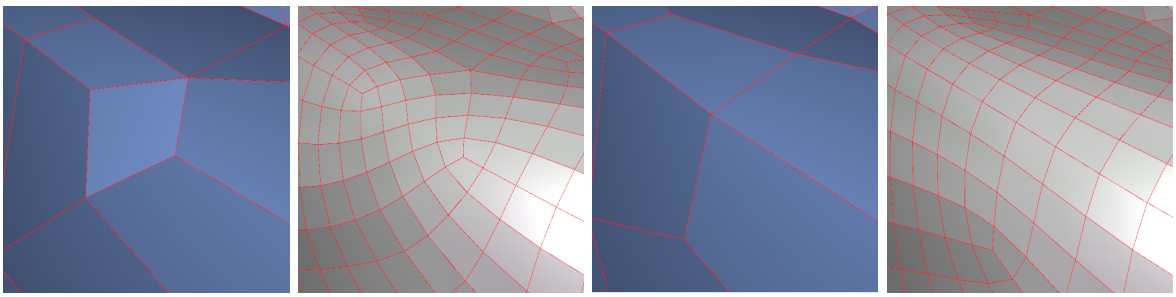
The curvature and orientation of the extracted vertices is an average over the curvature and orientation of the vertices that contributed to them in the position field.

The first three criteria make sure that only quads on sharp feature lines are considered. The fourth criteria excludes false-positives where the removal or split would be performed on the wrong diagonal and thus worsening the result.

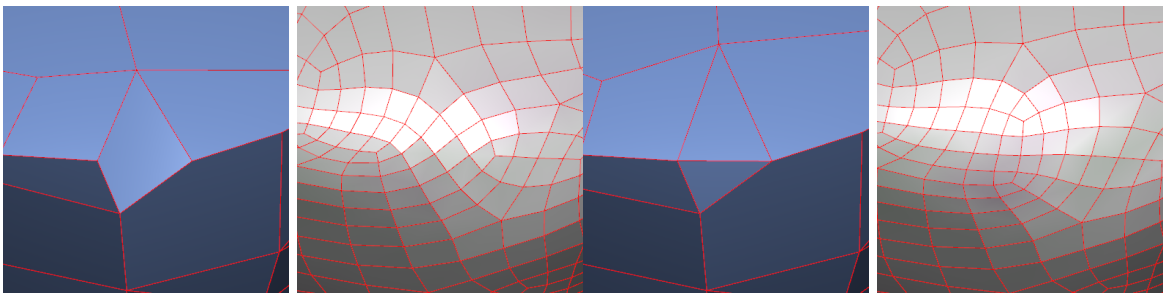
This post-processing improves the feature alignment of the extracted mesh. Edges follow the feature lines again and when a quad is removed, the endpoints of the diagonal are combined. Mis-aligned quads on feature lines tend to have endpoints with valence-3 (i.e. extraordinary) vertices and their combination creates a regular vertex. When splitting vertices, two triangles are introduced. Both cases of clean-up, their cleaned version and a corresponding subdivision control mesh that was subdivided twice are shown in Figure 3.33.

3.7.3. Flat Neighbouring Triangles

When two neighbouring triangles lie on a similar plane, they are merged into one quad. This reduces the number of non-quads and thus improves the quality of the mesh. These kinds of triangles are avoided and removed during extraction but because new triangles are introduced during the post-processing, newly created ones must be addressed. Since only triangles on similar planes are combined, the loss in detail is small. Two triangles are considered to lie on a similar plane when their normals differ by less than 10 degrees, which is a good threshold when comparing all meshes from Jakob et al., 2015's work.



(a) Quad to be removed (b) After quad removal



(c) Quad to be split (d) After quad split

Figure 3.33.: The difference between subdivision meshes where mis-aligned quads exist (a, c) and where they have been removed (b, d) is most noticeable when looking at the topology of the mesh. The feature lines are kept intact by removing or splitting the quads. Removing (top) is the preferred solution but when the adjacent edges do not form a straight line (bottom), the quads have to be split to avoid the loss of detail.

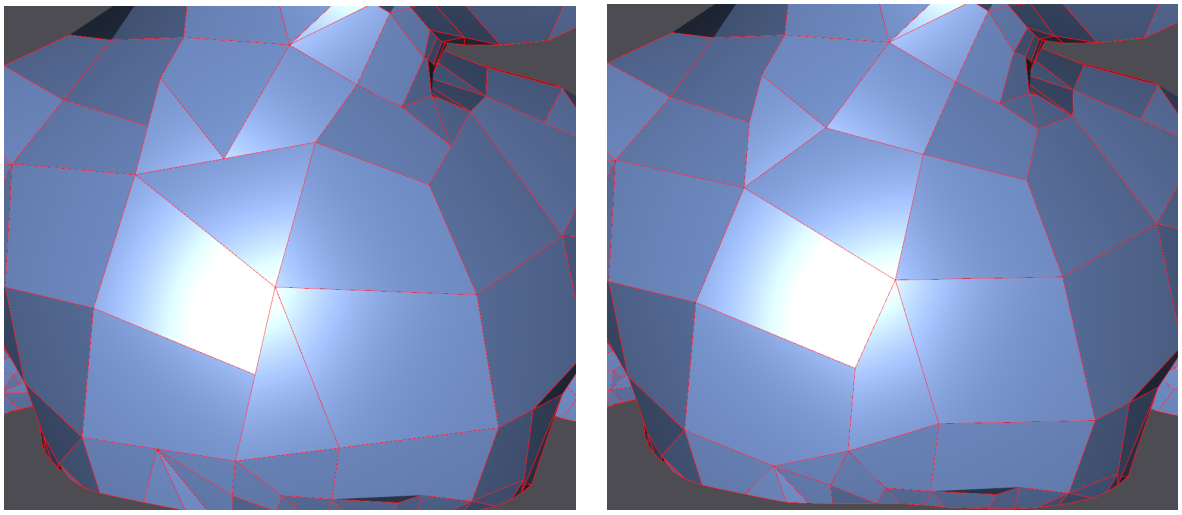


Figure 3.34.: The difference between a non-smoothed (left) and a smoothed (right) extracted mesh. Edge lengths become more similar and the distortion is lower after smoothing, which improves the average quad quality.

3.7.4. Smoothing

The position field optimization does not always produce perfect quads. Especially in areas where large edge lengths meet short edge lengths, some quads might be very distorted. To improve the quality of distorted quads without losing detail, tangential smoothing is performed. New vertex points are computed by averaging over all neighbours in a 1-ring neighbourhood and then projecting them onto the tangential plane of the original vertex. To preserve the features of the mesh, vertices are not smoothed when the largest angle between the normals of their neighbouring faces is greater than 50 degrees. Three iterations of smoothing are able to restore distorted quads while not smoothing edge lengths by too much. The difference between a smoothed and non-smoothed mesh can be seen in Figure 3.34.

3.8. Subdivision Control Mesh Creation

The next step is creating a Catmull-Clark subdivision control mesh from the extracted mesh. First, the quad-dominant mesh is converted to a pure quad mesh via one iteration of subdivision. Next, a subdivision control mesh is created via iteratively moving the vertices of the quad mesh along their normal until the limit point closely approximates the input mesh. Thaller, Augsdörfer, and Fellner, 2016 have formulated this method. They stop either after 20 iterations - which they have determined to be enough - or when the maximum distance between old and new positions for vertices are below a threshold ϵ. In this thesis ϵ is $1/10000$ of the greatest dimension of the reference meshes bounding box. Boundary vertices remain fixed with the

implication that they are fixed during subdivision as well. Because of the fixation, the limit surface will not be a cubic B-spline at the boundary. Since the goal is to produce a mesh without boundary, this case is very rare. The algorithm for the creation of a subdivision control mesh is as follows:

mesh ← output of curvature adaptive mesh generation
while Less than 20 iterations **do**
  for all vertices ∈ mesh **do**
    if vertex is not on boundary **then**
      $LimitPoint \leftarrow$ limit point of vertex
      $I \leftarrow$ intersection of reference mesh and Ray[$LimitPoint$, normal]
      $NewVertexPos \leftarrow vertex + (I - LimitPoint)$
    end if
  end for
  if $\max_{v \in mesh} |NewVertexPos - OldVertexPos| < \epsilon$ **then**
    break while loop
  end if
  mesh ← all $NewVertexPos$
end while

In order for this to work, the limit points of the extracted mesh need to be calculated. Catmull-Clark subdivision surfaces correspond to bi-cubic b-splines in the regular region. In the non-regular region, for extraordinary vertices with valences not equal to four, Halstead, Kass, and DeRose, 1993 have derived a linear combination to compute the limit point from the control points. It can be expressed as a stencil which is convoluted with the mesh. Stencils for vertices with valence four and five can be seen in Figure 3.35 and Figure 3.36. For regular vertices, this evaluation is equal to the evaluation of a bi-cubic patch in the regular region(Catmull and Clark, 1978). In the general case, for a vertex v with valence n, neighbours e and opposite vertices g the stencil can be expressed by the following equation:

$$\frac{1}{n(n+5)}\left[n^2 v + 4\Sigma_j e_j + \Sigma_j g_j\right]$$

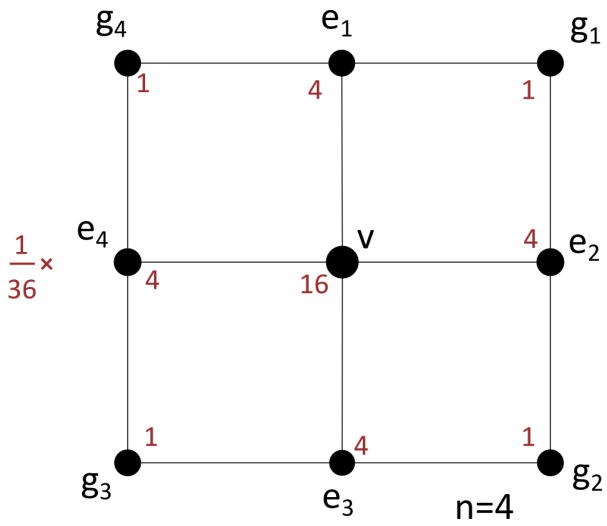


Figure 3.35.: The stencil together with the labels of the vertices and their respective weights for $n = 4$

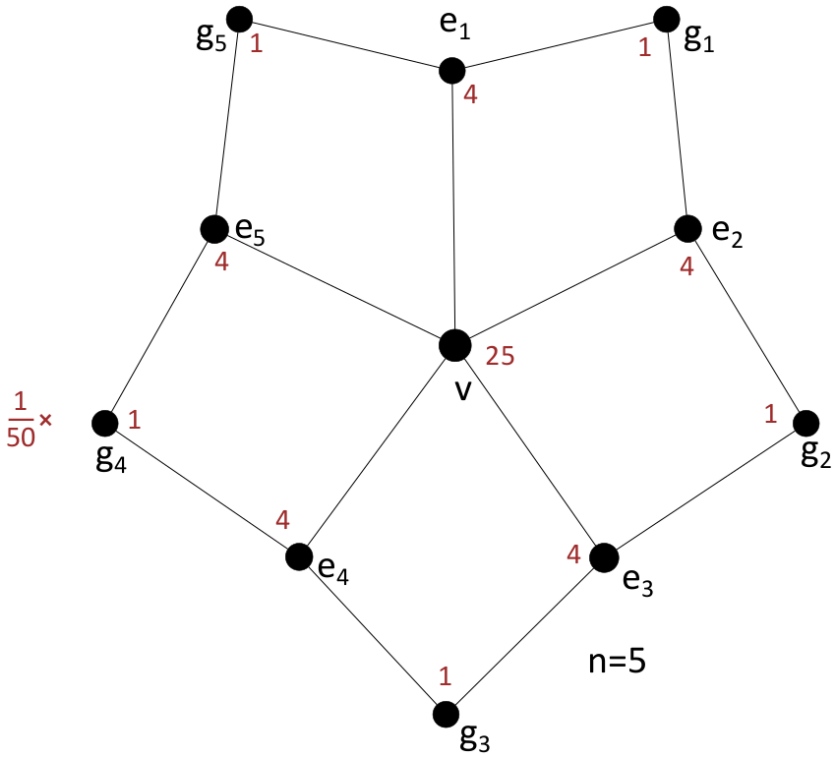


Figure 3.36.: The stencil together with the labels of the vertices and their respective weights for $n = 5$

3.9. Software Interface

To show the methods developed in this thesis, the application developed by Jakob et al. was expanded. The following will describe the application and where to find the introduced settings. For the sake of brevity, only the changed or newly created settings will be described. Figure 3.37 shows an overview.

The first set of settings is found in the panel for *Pre-Load Options* shown in Figure 3.38. There, the settings for the estimation and averaging of curvature, and the hierarchy construction can be found. Descriptions of what these settings are and what they change can be found in Subsections 3.1.3 and 3.1.4.

After loading the mesh via the panel *Open Mesh*, further behavioural changes can be set under *Configuration details*: Iterations per level described in Section 3.4, the method for curvature transformation to edge length and a factor that determines either the constant factor of the radius or the epsilon on which to base the edge length (depending on the chosen method) from Section 3.2, the curvature to use as defined in Subsection 3.1.2, whether to use averaged curvature, and the settings concerning the position field that were introduced in Section 3.4. Figure 3.41 gives an overview of these settings.

When choosing the transfer function as the method for calculating the edge length, a transfer function can be defined under *TransferFunction*, as shown in Figure 3.39.

The application of Jakob et al. is able to display different layers of information of the mesh or to filter displayed information based on hierarchy level. Various layers have been added: The resulting subdivision control mesh, curvature information and principal directions, feature vertices and the difference between orientations and principal directions. The layer for curvature information is coloured such that the maximal/minimal curvatures have the most intensive colour (violet being positive, and green being negative curvature). Maximal/minimal curvatures are used as normalisation factors by default. When the box next to the layer is ticked, then the non-averaged curvature is always used for normalisation - this makes the visualisation independent of the averaging. These layers can be accessed in panel *Advanced* which is pictured in Figure 3.40.

The last panel, *Export mesh*, provides options to switch between exporting a pure quad mesh and a quad-dominant mesh, to display additional edges for non-quads (non-quads are stored as a ring of triangles around a virtual vertex), whether non-manifold removal should be performed after extraction and the minimal cluster size for vertex extraction. The option for additional smoothing after the extraction was left in for flexibility. It is not needed for the purposes of this thesis as it is non-tangential smoothing and will decrease the accuracy of the result. The panel also has buttons for mesh extraction, for post-processing (optimization), for creating the subdivision control mesh, and for performing optional subdivision steps. The subdivision control mesh is coloured based on the distances of the control vertices to the reference mesh.

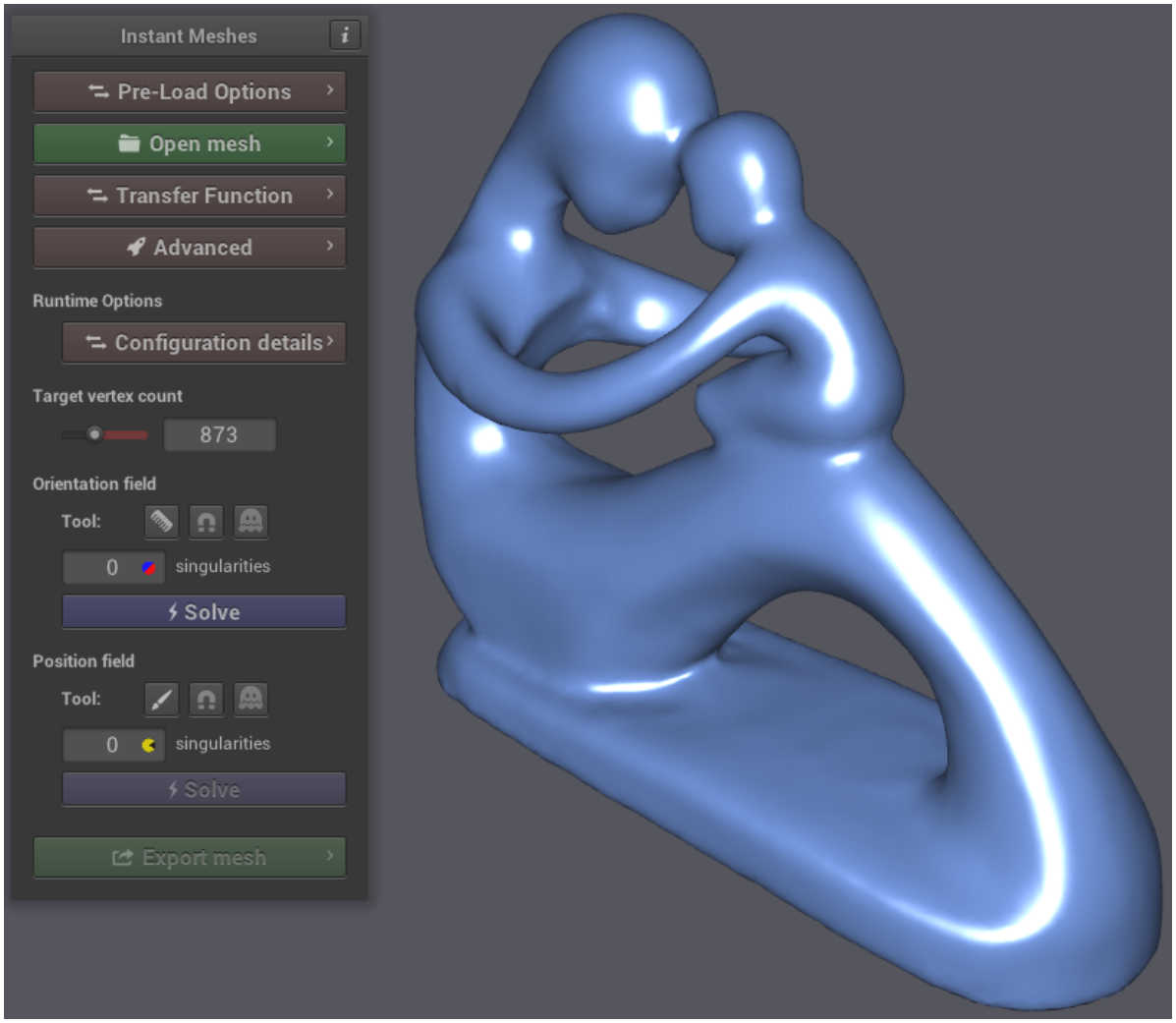


Figure 3.37.: An overview of the look of the modified application of Jakob et al. To the left is the control panel where the different steps of the algorithm can be executed and additional panels for options can be accessed.

The colour intensity is normalised such that a distance of one percent of the maximal bounding box axis of the reference mesh has the most intensive colour. Violet means a difference along the normal of the control vertices while green is a difference in the negative direction of the normal. The resulting mesh can then be saved as '.ply' or '.obj' file. The *Export mesh* panel can be seen in Figure 3.42

Additionally, the application has a console window which displays information concerning the current or completed tasks. When extracting a mesh or when the extracted mesh is processed further, the quality criteria defined in Chapter 4 are displayed.

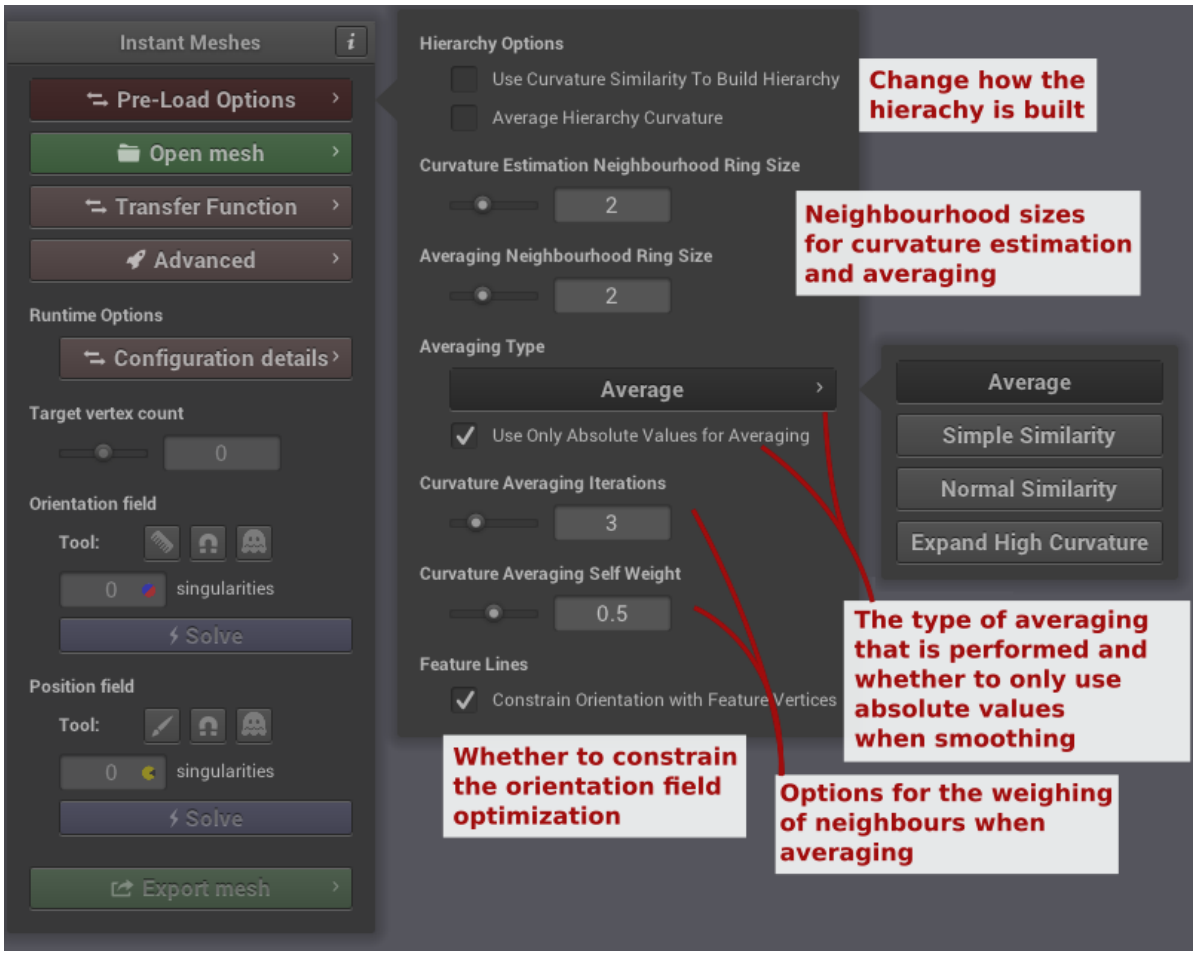


Figure 3.38.: *Pre-Load Options* panel and the settings that can be changed there.

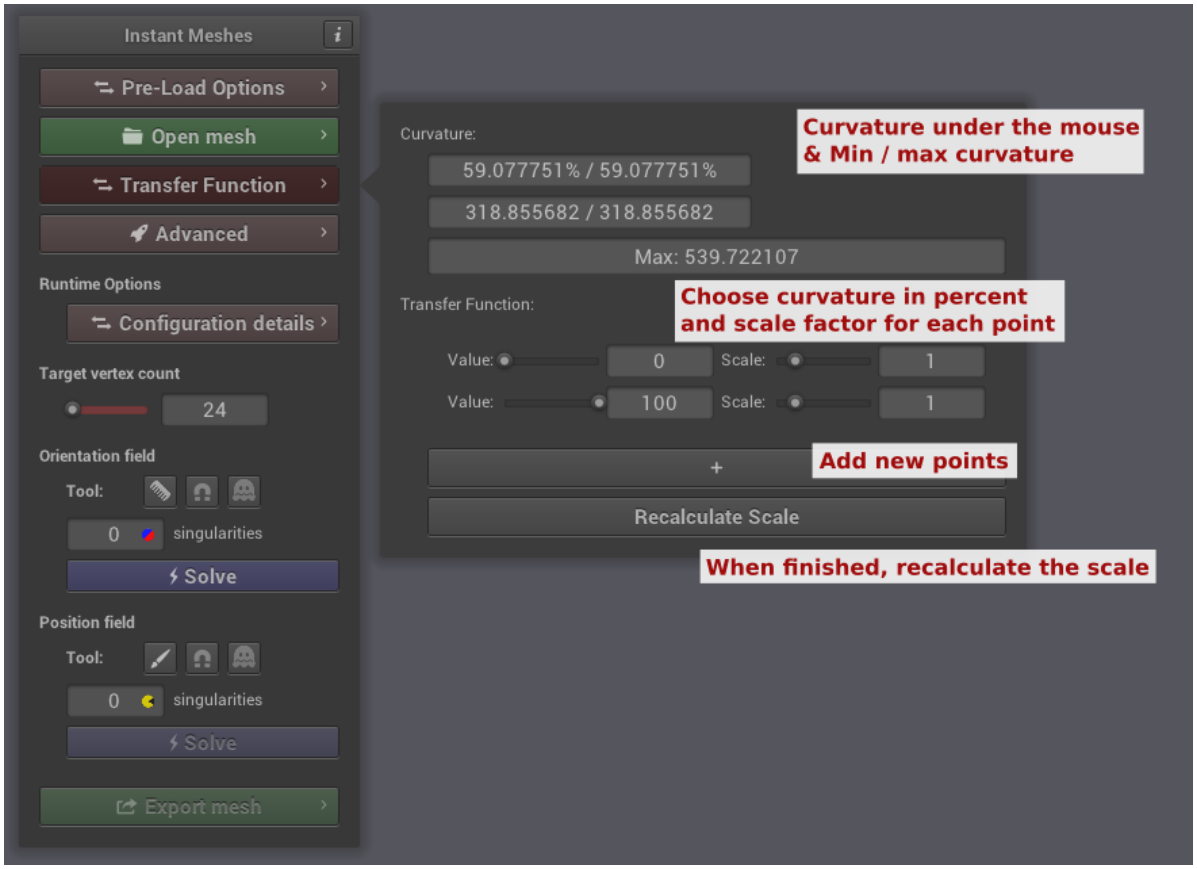


Figure 3.39.: View of the transfer function editor where the points for a transfer function can be defined.

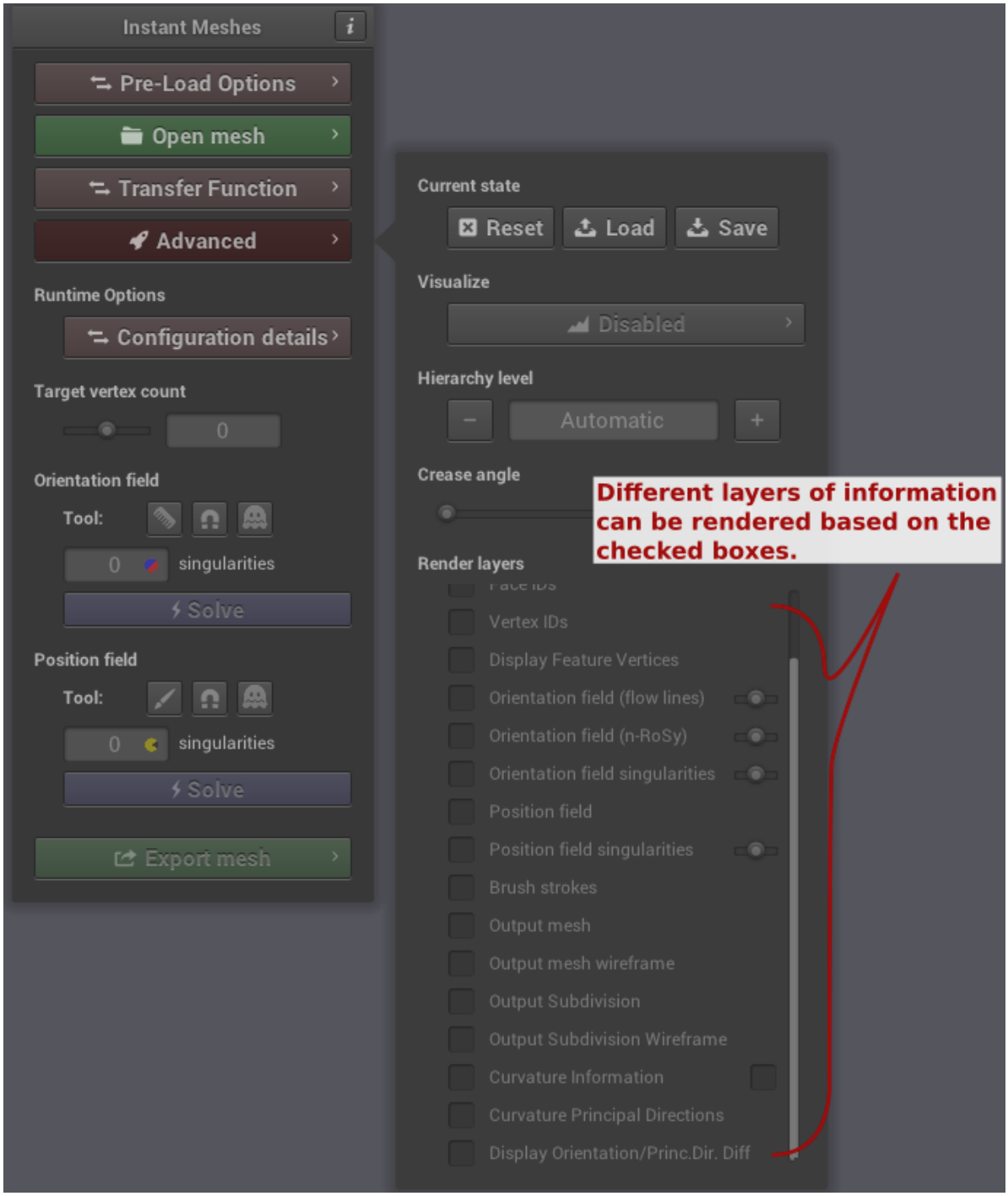


Figure 3.40.: The *Advanced* panel and the different layers that can show additional information.

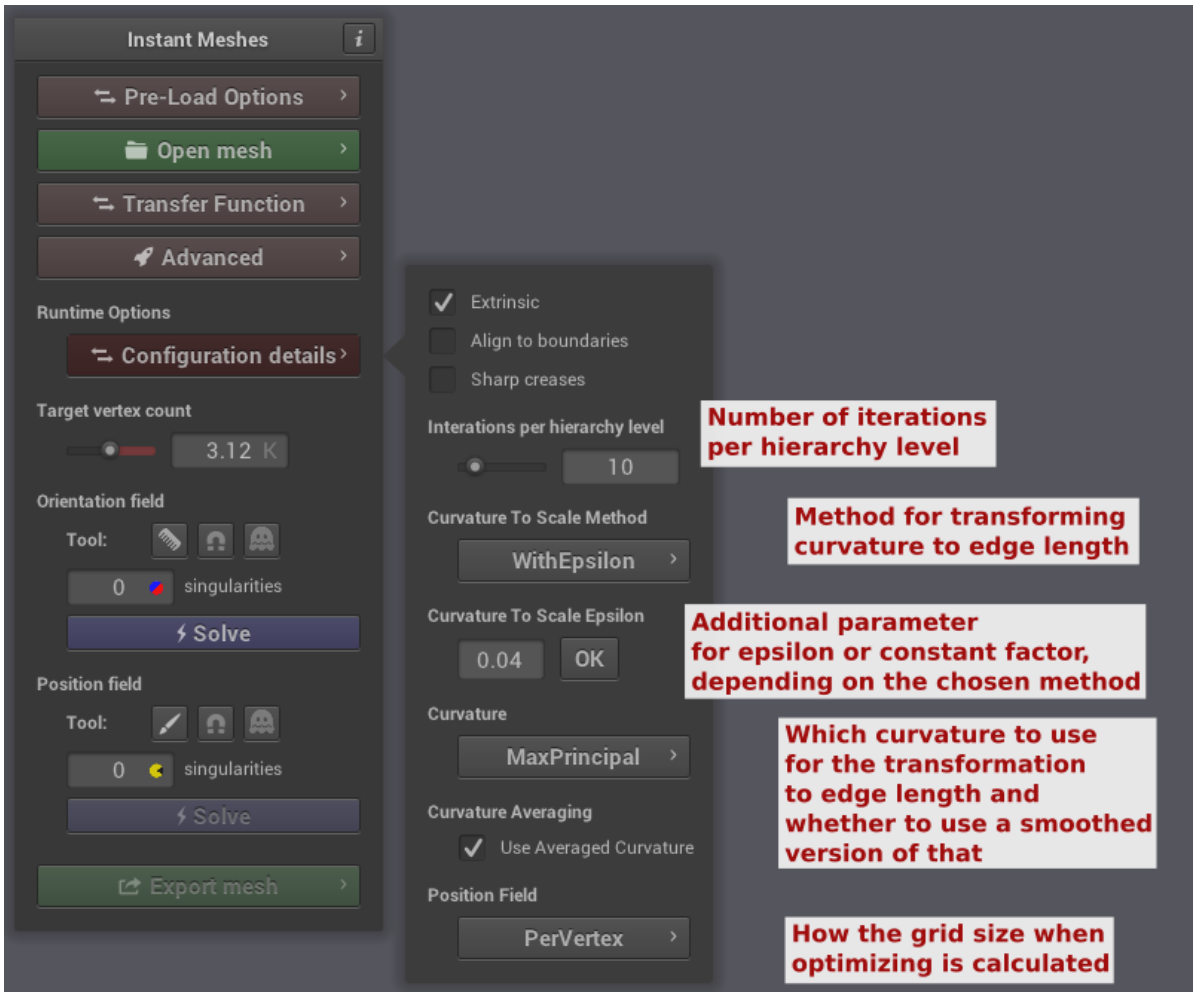


Figure 3.41.: The run-time options that are available.

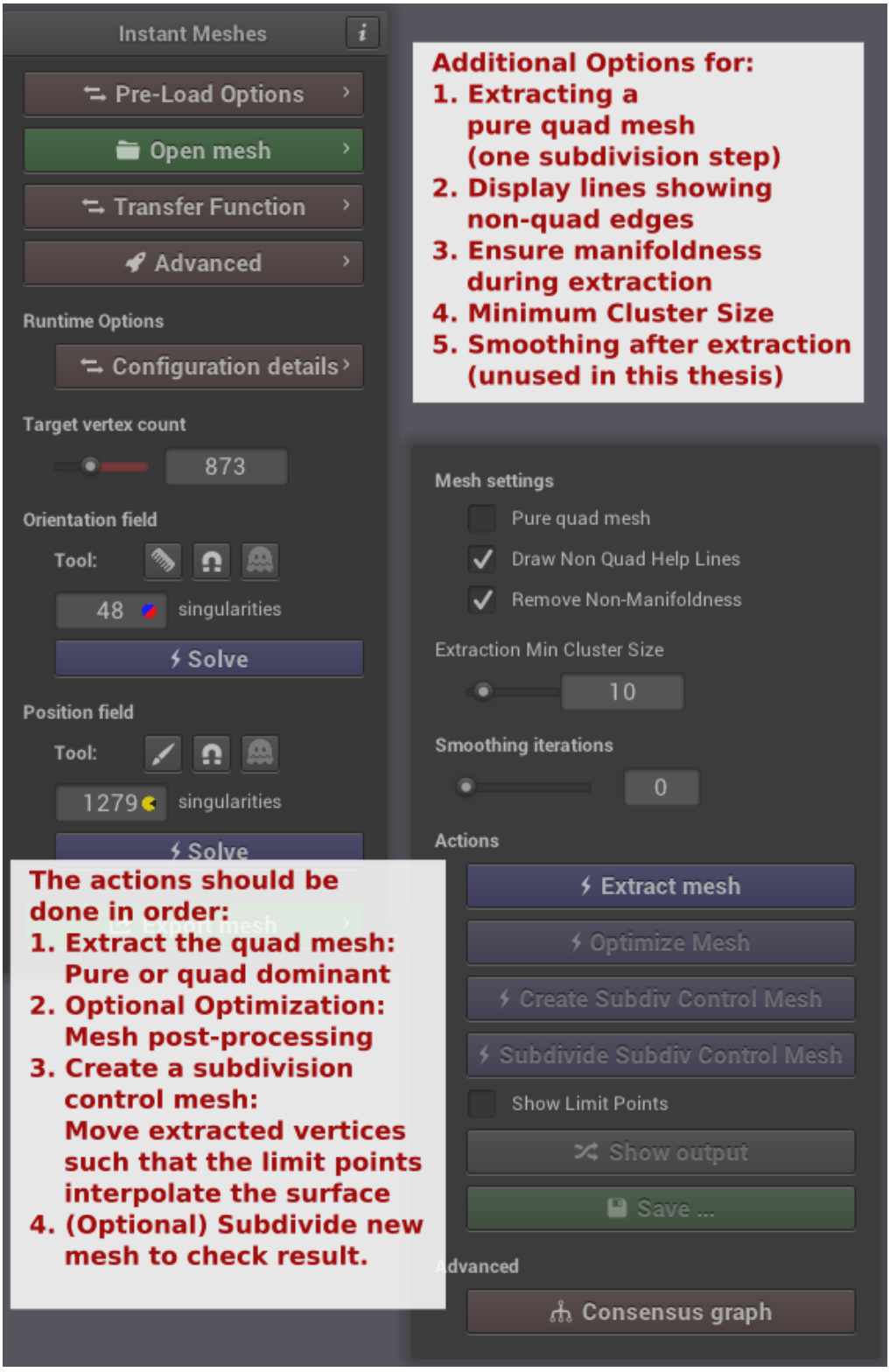


Figure 3.42.: The *Export Mesh* panel which contains the final steps for creating a quad mesh from the reference and to create or subdivide a subdivision control mesh.

4. Observations

To achieve the goal of this thesis, different ideas for improvement in all steps of the meshing were tried out. Chapter 3 has introduced the changes made to the original method in Jakob et al., 2015. These changes must be evaluated to find an ideal combination. The evaluation is made on the extracted quad-dominant meshes and not on the subdivision control meshes because the quad-dominant meshes set the basis for good subdivision control meshes. The following options were introduced:

Curvature Estimation (Subsection 3.1.1): The size of the neighbourhood of the estimation can be changed.

Choice of Curvature (Subsection 3.1.2): Mean curvature, Gaussian curvature, curvature in both directions (i.e. rectangles instead of squares) or the largest absolute principal curvature may be chosen to adapt edge lengths.

Edge Length Computation (Section 3.2): The transfer from curvature to edge length may be based on a transfer function, a simple inverse of the curvature, or the root of the inverse. Additionally, an epsilon can be chosen when not using a transfer function which defines a factor for the simple inverse or the allowed error for the root of the inverse.

Curvature Averaging (Subsection 3.1.3): It is possible to choose whether to use averaged curvature. An average of the curvature may be computed as a simple average, a similarity-based one or an average that tries expand high curvature regions. The calculation itself has options for the number of iterations, a self-weight, the size of the neighbourhood and whether to use only absolute values for smoothing.

Hierarchy Computation (Subsection 3.1.4): The weights for the merging of vertices may be changed from normal to curvature similarity-based ones. Also, curvature can either be propagated up the hierarchy or can be computed for each level.

Orientation Field Optimization (Section 3.3): The orientation field may be constrained with the orientations of vertices at creases of the reference mesh.

Position Field Optimization (Section 3.4): The size of the grid for a pair of vertices may be the average of the sizes of both points, the reference point's size or separate for both vertices.

Iterations: The number of iterations on each level of the hierarchy for the position field and orientation field optimization can be chosen.

As mentioned on various occasions in Chapter 3, some options are clear favourites when compared with their alternatives: For curvature computation, the analysis in Subsection 3.1.1 shows that a 2-ring neighbourhood is the ideal middle-ground between precision and noise reduction. Subsection 3.1.2 explains how using the largest absolute principal curvature is the best choice of curvature information to define the position field grid scale. As discussed in Subsection 3.1.4, the changes to the construction of the hierarchy have not improved the results. The best options are the original construction of the hierarchy while estimating the curvature separately for each of its levels. Finally, Section 3.3 shows that the grid sizes of the position field optimization should be defined per-vertex. The demonstration for per-vertex grid sizing shown in Figure 3.23 is consistent for all tested reference meshes.

Unfortunately, not all choices are as clear. To find the best choices and to compare the results, different metrics must be calculated. The chosen metrics are some of the most prevalent ones for comparing results in the papers that were discussed in Chapter 2. The following metrics are computed after extraction, after optimization and after creating a subdivision control mesh:

- Percent reduction of vertices compared to the reference mesh.
- Percentage of EV's.
- Percentage of non-quads.
- Average distance between diagonal centre points, to measure the planarity of quads.
- Average angle for quad corners, to measure the orthogonality of quads.
- Average ratio of opposite edge lengths of quads (as a measure for the deviation from rectangles/squares)
- The average of an adapted *quad quality measure* introduced in Marinov and Kobbelt, 2006, over all quads.
- Number of non-manifold and boundary vertices.
- The Hausdorff distance between the extracted mesh and the reference mesh.
- The average edge length and the standard deviation of edge lengths (to detect meshes with uniform edge lengths).
- The average and maximum distance of face centres to the reference mesh. Coarse meshes cannot model fine detail of the mesh well, and so their face centres are further away from the reference mesh compared to meshes with more density in these areas.

The *quad quality measure* from Marinov and Kobbelt, 2006 combines orthogonality, parallelism and the deviation from a square and a rectangle into one. For this thesis, the distinction between rectangle and square was omitted because it is not as important as the other three properties. For a quad with inner angles $\alpha_0, \ldots \alpha_3$ and edge lengths $l_0, \ldots l_3$, the quad quality measure E has its lowest, most optimal value at $E = 0$ and is defied as:

$$\text{Orthogonality } O = \sum_{i=0}^{3} |\alpha_i - \pi/2|$$

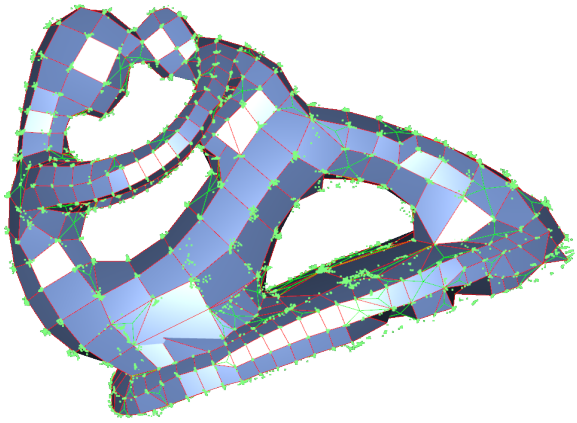
$$\text{Paralellism } P = \sum_{i=0}^{3} |\alpha_i + \alpha_{i+1 \pmod 4} - \pi|$$

$$\text{Rectangularity } R = \left(2 - \frac{min(l_0, l_2)}{max(l_0, l_2)}\right) \cdot \left(2 - \frac{min(l_1, l_3)}{max(l_1, l_3)}\right)$$

$$E = [(1 - O) \cdot (1 - P) \cdot R] - 1$$

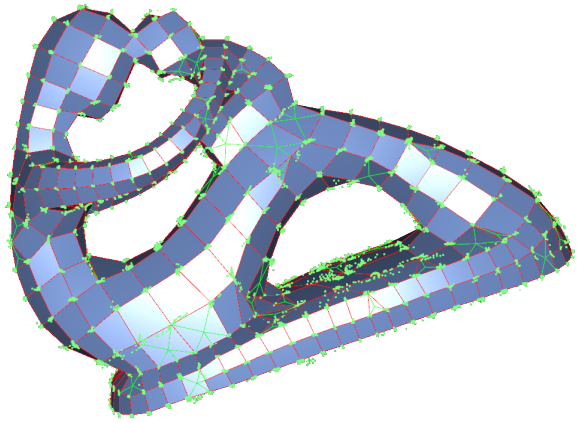
The first observation when comparing results is that averaged curvature produces much better results as non-averaged curvature. This is because when optimising the position field, similar grid sizes improve the results. Since the curvature information is just an estimation based on a reference mesh, noise cannot be avoided. Averaging reduces the noise of this estimation. It has to be noted that, while averaging improves the results, too much averaging will cause a mesh that has well-formed quads (according to the metric for quad quality) but is not adaptive to the curvature anymore because all points will have similar curvature information. Figure 4.1 shows a comparison between averaged and non-averaged curvature together with some quality criteria. As discussed in Subsection 3.1.3, absolute curvature is used when averaging for all future comparisons.

#Vert	705
EV's	51.49%
Non-Quads	20.33%
Hausdorff	3.793
Avg. Quad Quality	6.071
Avg. Edge Length	6.647
Edge Length St. Dev.	5.313
Max Face Centre Dist.	4.295



(a) Without Averaging

#Vert:	741
EV's	36.98%
NonQuads	14.82%
Hausdorff	2.245
Avg. Quad Quality	4.109
Avg. Edge Length:	6.841
Edge Length St. Dev.	4.746
Max Face Centre Dist.	2.217



(b) With Averaging

Figure 4.1.: This comparison shows the tremendous difference that averaging makes. While averaging increases the number of vertices slightly (as very low curvature areas shrink slightly) and the edge lengths become more uniform (as can be observed in the edge length standard deviation), all other aspects of the extracted meshes' quality improves. Most notably are the drop in the number of EV's and the improved face centre distance. The metrics not shown here were improved as well but are left out for brevity.

The remaining options are the curvature transformation to edge length, curvature averaging and the number of iterations of the position field optimizer on each level of the hierarchy. An analysis to find the ideal combination was performed. Together with above-mentioned fixed options, the following is the range of options with which the meshes were tested:

Curv. Estimation Neighbourhood	2-ring
Used Curvature	Largest absolute principal curvature
Averaging	Yes
Averaging Type	*Simple Average*, *Expand High Curvature*
Averaging Neighbourhood	1-ring and 2-ring
Averaging Iterations	1, 3, 5
Averaging Self-Weight	0, 0.2, 0.5, 0.7
Hierarchy Creation	Normal-Based
Curvature of Hierarchy Levels	Computed at each level
Position Field Grid Size	Per-Vertex
Iterations Per Level	6, 10, 15
Edge Length Epsilon	0.01, 0.02, 0.03, 0.04
Orientation Field Constraints	Yes and No

Averaging types *Simple Average* and *Expand High Curvature* were chosen because they showed the most promise. The options concerning the iteration number and the self-weight were chosen such that they cover low, medium and high cases, from which their influences can be inferred.

The metrics were analysed over the same meshes provided in Jakob et al., 2015. Jakob et al. have incorporated a batch mode into their software which allows an automated run of the extraction without a user interface and the need for user interaction. This batch mode was adjusted to be configured with above-mentioned options and to test a range of different option sets. The resulting meshes of these batch runs are saved as '.ply' files and the computed quality criteria are saved in '.csv' files for further evaluation. Since the quality criteria are saved in a format readable for matrix-based computation, Matlab (MATLAB, 2017) was used to find optimal option sets. To avoid randomization artefacts, all meshes were run three times for the whole range of options.

To find the optimal option set, a measure for optimality must be created. Finding this optimality measure is non-trivial as some metrics stand in conflict with each other. Because not all metrics are needed the following ones were used: Average quad quality measure, Hausdorff distance, inverted standard deviation of edge lengths (more is better, it counteracts over-averaging), and the maximum distance of face centres to the original mesh. Meshes with boundaries of non-manifold elements were discarded because closed manifold meshes are the goal. The optimality measure is

the squared sum of the normalised values of these metrics. The best combination of options according to this optimality measure differs even for multiple passes of the same mesh. However, certain trends can be observed. The following is the set of options derived from this evaluation that is recommended for best results overall:

Curv. Estimation Neighbourbood	2-ring
Used Curvature	Largest absolute principal curvature
Averaging	Yes
Averaging Type	*Simple Average*
Averaging Neighbourhood	1-ring or 2-ring
Averaging Iterations	$1 - 3$
Averaging Self-Weight	$0.0 - 0.5$
Hierarchy Creation	Normal-Based
Curvature of Hierarchy Levels	Computed at each level
Position Field Grid Size	Per-Vertex
Iterations Per Level	15
Edge Length Epsilon	$0.02 - 0.03$
Orientation Field Constraints	Yes

Averaging in this set of options is a well-balanced mixture of not over-averaging but still removing enough noise and smoothing out the change of curvature. The iterations per level have to be raised compared to the work in Jakob et al., 2015 because of the increased complexity of the optimization through the different grid sizes in the parametrization. An increased run-time coming from more iterations is acceptable for the increased quality of the results.

5. Results

The meshes resulting from the work in this thesis are able to represent reference meshes accurately, with an emphasis on distributing vertices so that details are preserved. A change in density leads to vertices with a valence different to four or to the introduction of non-quads. Therefore, the resulting meshes in this thesis have more singularities when compared with other approaches. Figure 5.1 displays the whole pipeline of the developed method. The resulting quad-dominant mesh has uniform quads in similar curved regions of the reference mesh and adapts to changes in curvature with non-quads or EV's. It has to be noted that the sizing of the faces is dependent on the density of the reference mesh. As mentioned in Section 3.2, the minimal edge length must be capped. To decrease the edge length further, the reference meshes' density must be increased through subdivision. Different levels of detail representation can be achieved when changing the desired density. Figure 5.2 shows results of the same mesh with different desired densities. The developed approach is able to reduce the number of vertices by up to 95% for the standard edge length setting.

Most isotropic approaches find optimal pure quad or quad-dominant meshes while trying to minimise the number of above-mentioned irregularities. This limits their abilities to adapt to curvature. When compared with the method developed in this thesis, the isotropic meshes of Jakob et al. have fewer extraordinary vertices and non-quads, and have quads with better quality. However, they are less dense in high curvature regions and thus cannot represent high-frequency detail as well when compared to the work presented in this thesis. The number of EV's and non-quads of the method developed in this thesis depends on the number of transitions between low and high curvature regions because quad-dominant meshes need non-quads or EV's to make the corresponding transition of edge lengths. To compare resulting quad-dominant meshes between the work in Jakob et al., 2015 and this thesis, all reference meshes from Jakob et al.'s were processed. For the original work, the default settings were used and for the work in this thesis the accuracy was set such that a similar number of vertices was extracted. Both

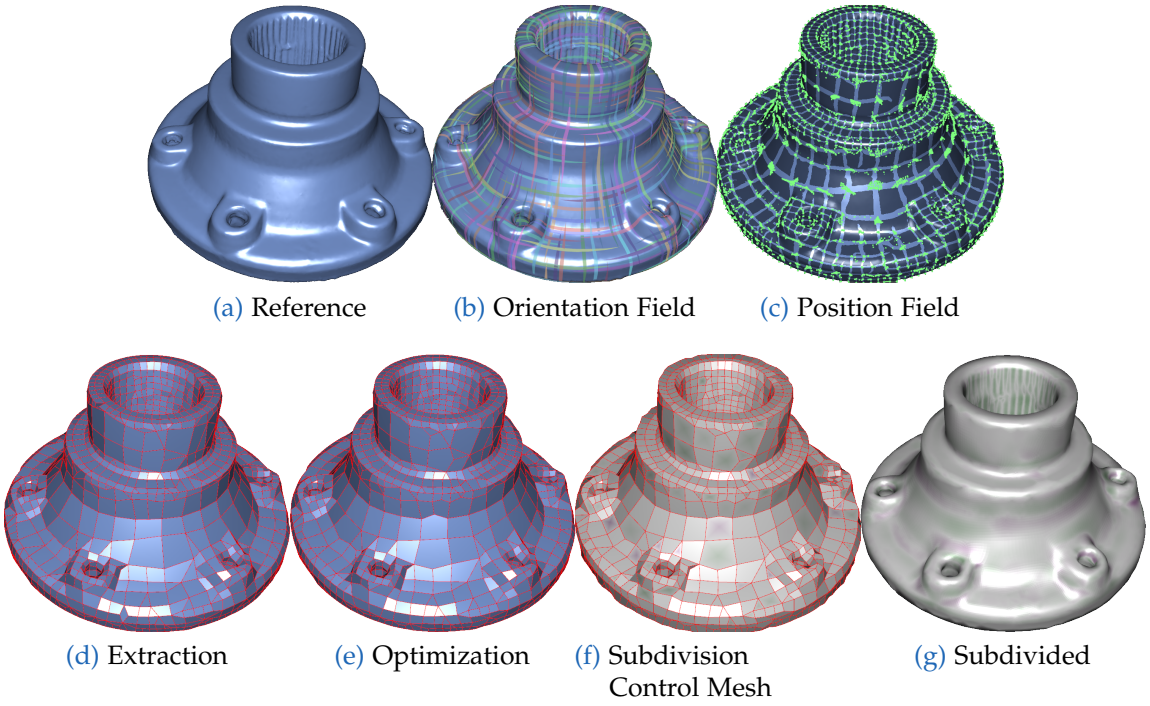


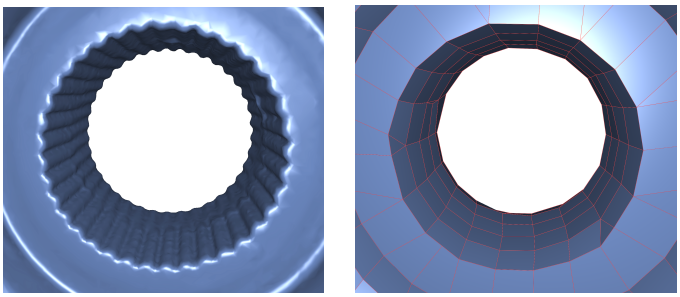
Figure 5.1.: This shows the whole pipeline of the developed algorithm. After loading the reference mesh (a) and estimating the curvature, the orientation field (b) and the position field (c) are computed. The extracted mesh (d) is optimised (e) and its vertices are moved to create a control mesh (f) where the limit surface closely resembles the reference mesh, represented via subdividing three times(g). The colours on (f) and (g) indicate positive (violet) and negative (green) distance form the reference mesh.

types of mesh are transformed into subdivision control meshes while computing their error metrics. Averaged over all meshes, the following can be observed:

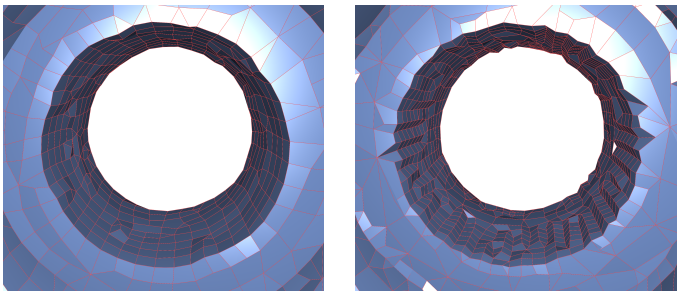
Error Metric	EV's	Hausdorff Distance	Avg. Quad Quality	Avg. Face Centroid Distance
Jakob et al.	0.28%	0.468	0.489	0.122
Thesis	0.89%	0.463	1.291	0.115
% Difference	+246.011%	-2.902%	+180.787%	-6.056%

This proves that the shifted vertex densities improve the Hausdorff distance between limit surface and reference mesh. It also shows how the focus on adaptive edge length worsens the number of extraordinary vertices and the quad quality.

Figures 5.3 shows the improved accuracy in detailed regions that curvature adapted edge lengths are able to achieve when compared to the isotropic approach in Jakob et al., 2015. The most notable differences are in regions with fine detail.



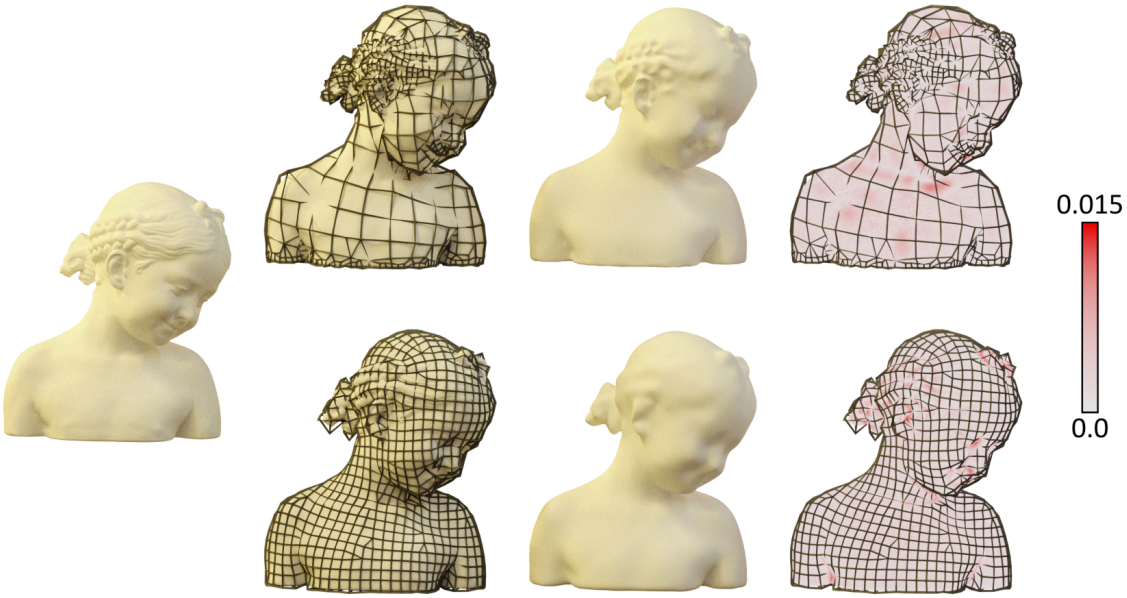
(a) Reference (b) Low Density



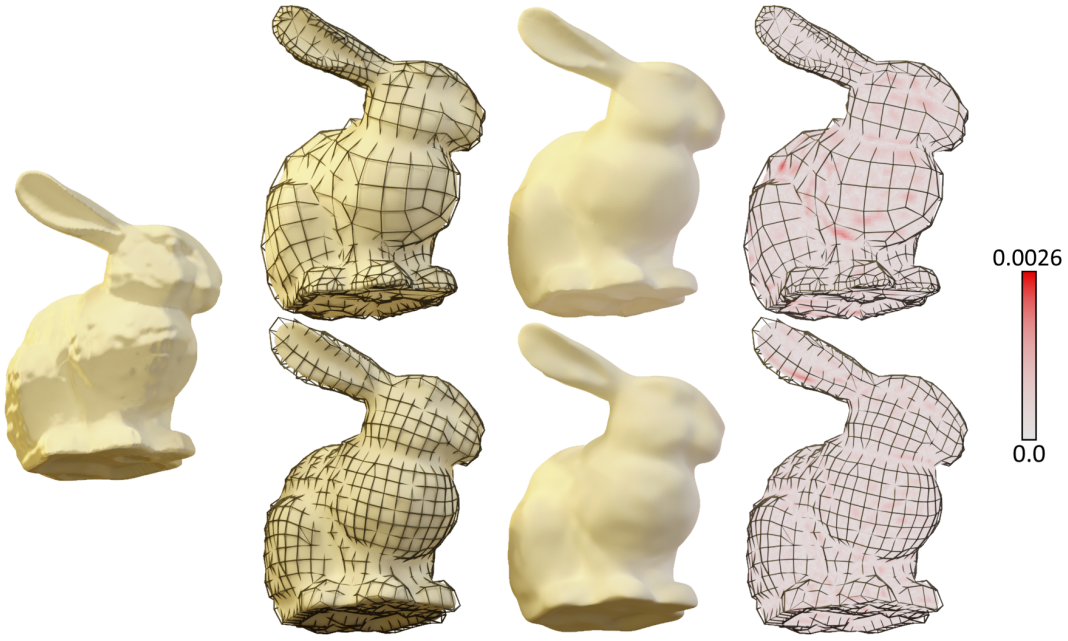
(c) Default (d) High Density

Figure 5.2.: With the global edge length, higher and lower densities can be enforced. For low (b) and default (c) densities, no subdivision is necessary but when trying to model the grooves of the reference mesh (d), a denser version of the reference mesh is necessary.

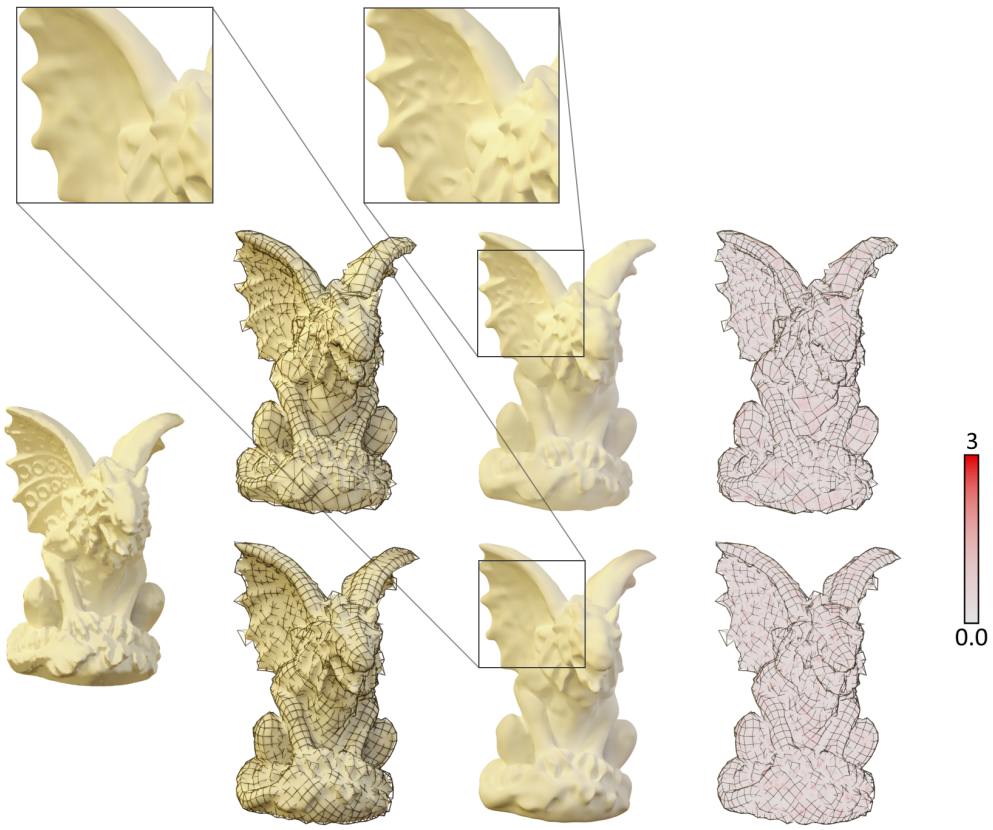
While anisotropic approaches take the detail of the reference mesh into account when extracting a mesh, their changes in edge length are less significant when compared to the results in this thesis, or they may need additional manual input. While they have more regularity, they cannot adapt to changing curvature as quickly and tend to have a more uniform distribution of edge lengths. Figures 5.4, 5.5, 5.6 and 5.7 compare the results of this thesis and of methods which have an anisotropic nature and adapt to the curvature of the reference mesh to a certain extent. For better comparability, the results are not transformed to subdivision control meshes yet.



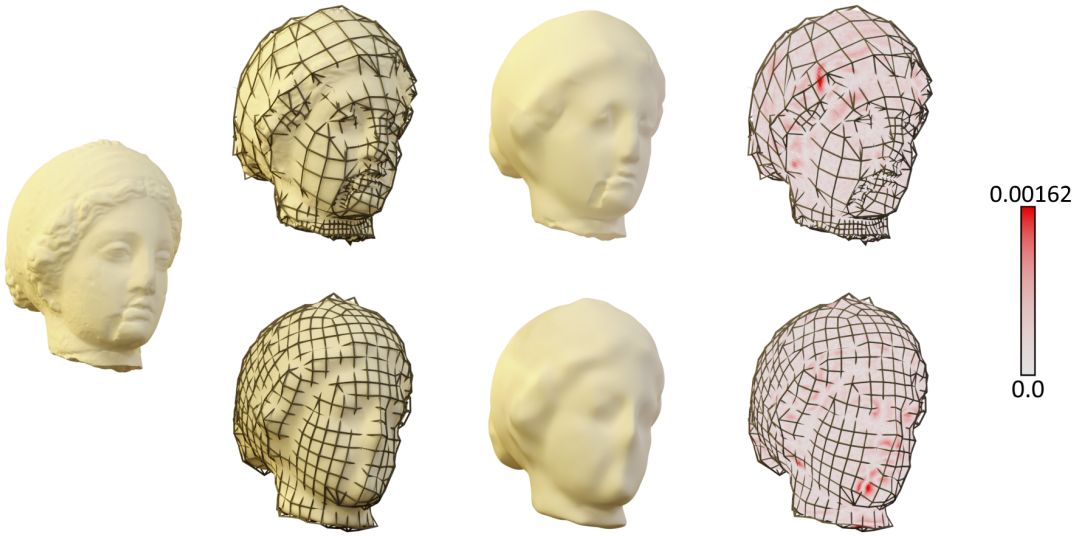
Bimba



Bunny



Gargoyle



Igea

Lion Vase

Figure 5.3.: A comparison of the results of Jakob et al., 2015 and the work of this thesis. The extracted quad-dominant meshes have about the same number of vertices and are both transformed to subdivision control meshes with the method in this thesis. On the left are the surfaces resulting from scan captures. 2nd from left: The subdivision control meshes generated from the scan data. The limit surfaces derived from control meshes are shown (3rd from left) before colouring the error between the scan data and limit surfaces (right).The top rows show the results of the method presented in this thesis. Their control meshes are dense in regions where it is required to capture fine detail, but coarse in low-curvature regions. The resulting limit surface shows a lot more detail when compared to the bottom rows, which shows results from Jakob et al., 2015's method, which is more isotropic. As a result, the bottom limit surfaces contain considerably less detail and the errors to the scan data are larger.

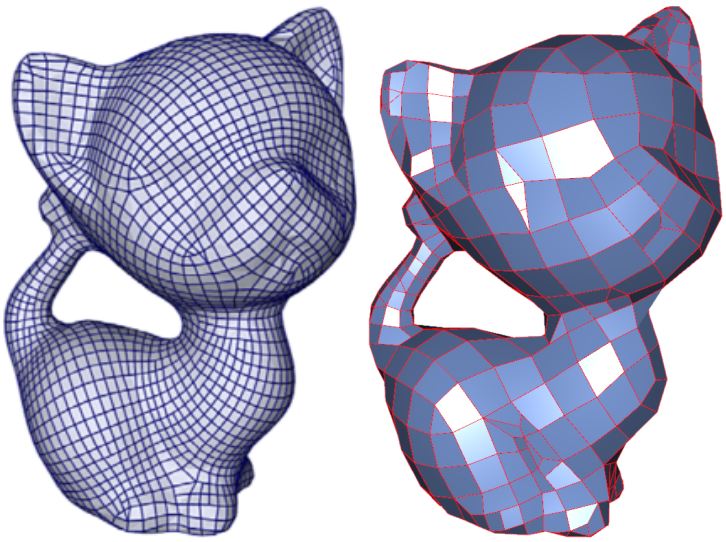


Figure 5.4.: A comparison between the work in Lai, Kobbelt, and Hu, 2010 (left) and an extracted and optimised quad-dominant mesh from this thesis (right). While they do change the sizes of their quads to fit the detail of the mesh, the differences are not as pronounced (e.g. the cat's tail). In their work they mention that a precise control over the size of the faces is not easy - whereas in this work it can be done by setting a desired estimated number of vertices.

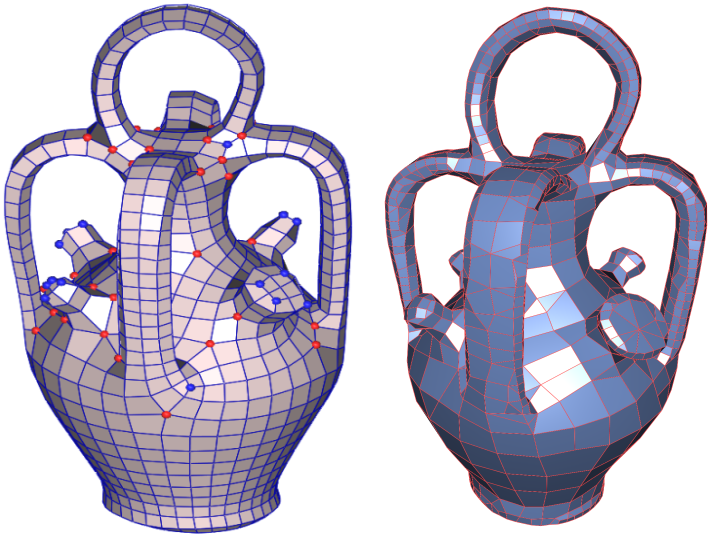


Figure 5.5.: While the work of Zhang et al., 2010 (left) shows more regularity than the approach in this thesis (right), it cannot change density quickly. While it is possible to enforce smaller quads, these areas have to be defined manually.

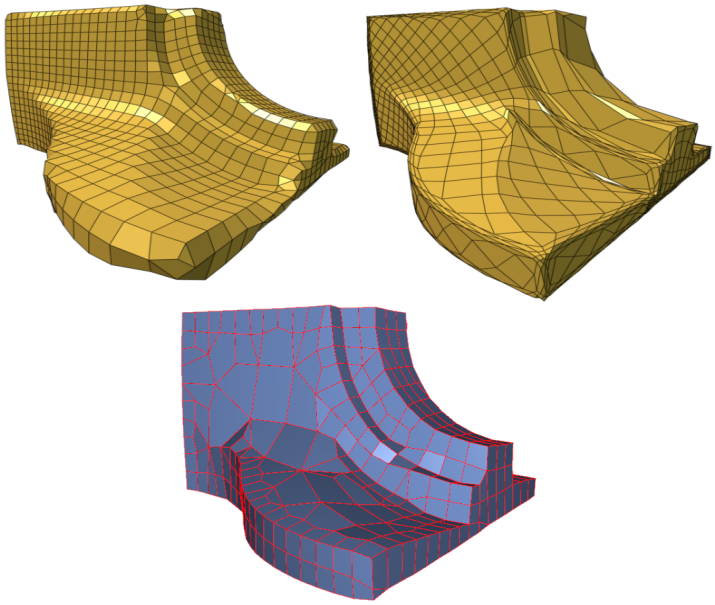


Figure 5.6.: When compared with the results in Kovacs, Myles, and Zorin, 2011 with different underlying parametrizations (top), the approach in this thesis approach (bottom) either adapts better to differences in surface detail (top left) or produces faces that follow principal curvature more closely (top right).

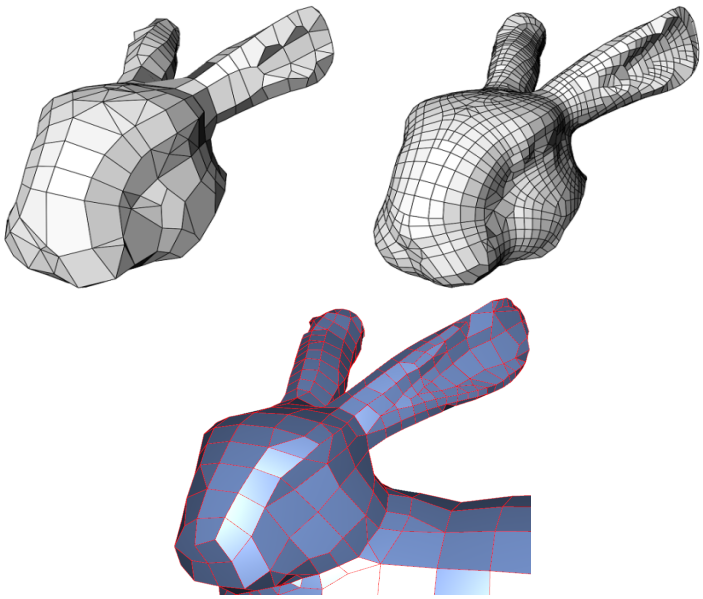


Figure 5.7.: While the density of the results of Alliez et al., 2003 (top) can be similar to the results in this thesis (bottom), they do not show the same variance in edge lengths.

All additional methods and many options for the behaviour of the algorithms are added to the software created in Jakob et al., 2015. With the embedded batch mode, a user may extract quad-dominant or subdivision control meshes quickly based on a set of default options - six seconds on the Stanford Bunny with 55684 vertices, 111364 faces (on a PC with 16GB RAM, and a four core CPU with 3.5GHz), for example. Because of the variety of options, it is also possible to find better fitting sets of options for specific meshes. Information (i.e. timing and mesh quality) is displayed on a console window or output into additional files when in batch mode.

6. Conclusion and Future Work

In this thesis, a quad-dominant meshing method with a focus on curvature-dependent density to accurately capture high-frequency detail and a subsequent transformation of the extracted meshes to subdivision control meshes was created. Because the extraction is fast and initialisations are random, trying different options to find good solutions is easy. Also, possible defects in the parametrization have been addressed through post-processing steps. Through this, it is possible to create subdivision control meshes which precisely model arbitrary meshes with a low number of vertices while still accurately representing the reference mesh. The trade-off, however, is a diminished regularity of the surface.

The design is very flexible and may be adapted to more specialised applications. The focus of this thesis was on an automated process which is able to produce satisfying results for a variety of inputs. Further work may include more post-processing to ensure that the extracted meshes fulfil the demands of more specific applications, more specialised settings for a subset of meshes or different constraints on the parametrization.

7. Acknowledgements

I would like to thank Jakob and his partners for all their intensive work in the field of re-meshing, as well all the sources of the meshes used by Jakob et al., which made my thesis possible.

Furthermore, I wish to extend my gratitude to Dipl.-Ing. Andreas Riffnaller-Schiefer for his excellent help offered with some of the imagery.

A heartfelt thank you to Dr.techn. Karl Voit for the wonderfully structured LaTeX-template used in the writing of my thesis.

I wish to thank Ass.Prof. Ursula Augsdörfer for all her guiding feedback, helpful mentoring as well as for all her thoughtful recommendations.

Appendix A.

Extracted Meshes from Parametrizations

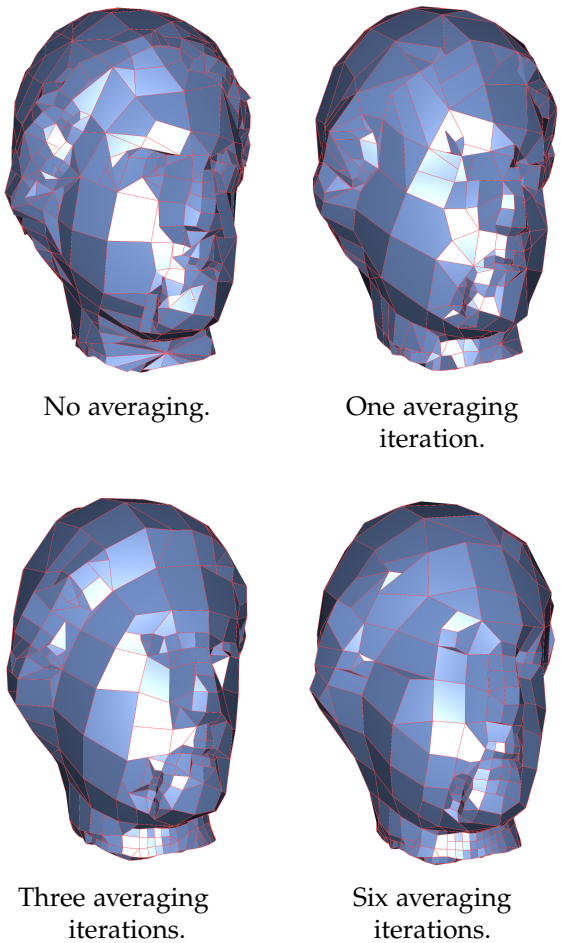
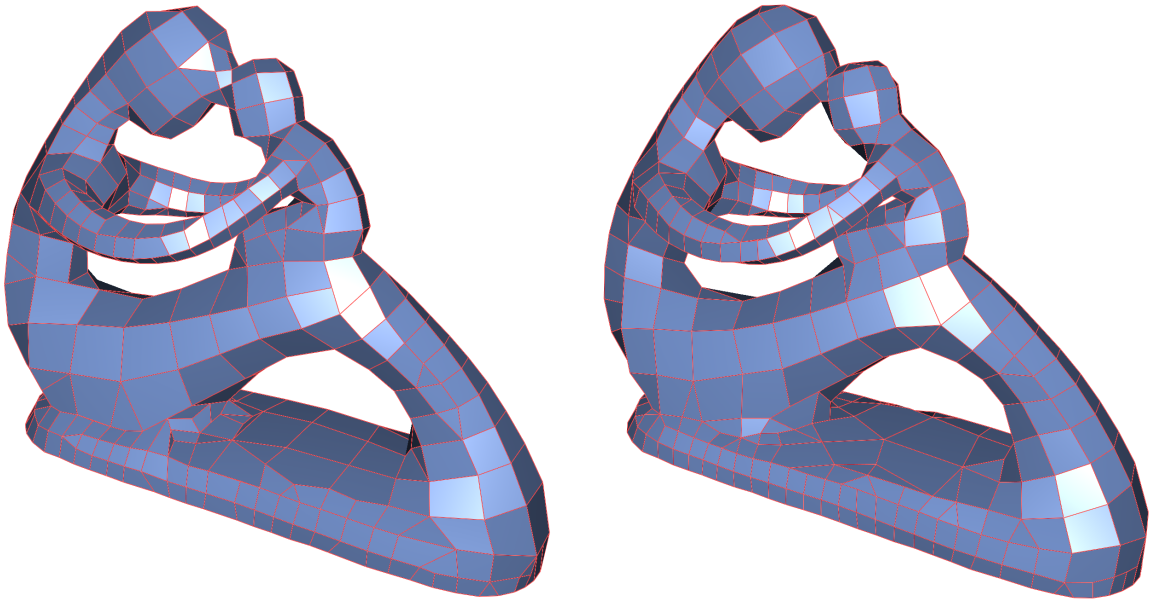


Figure A.1.: This figure shows the resulting quad-dominant meshes for the parametrizations of different averaging iterations shown in Figure 3.5.



Extracted mesh for similarity-based average with $k = 2$.

Extracted mesh for simple averaging.

Figure A.2.: This figure shows the resulting mesh from the parametrizations of Figure 3.7.

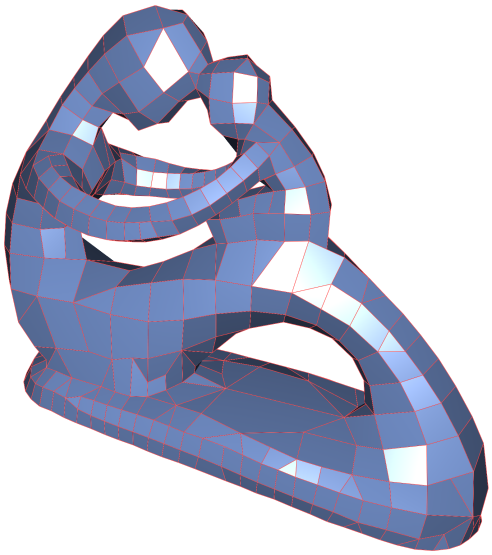
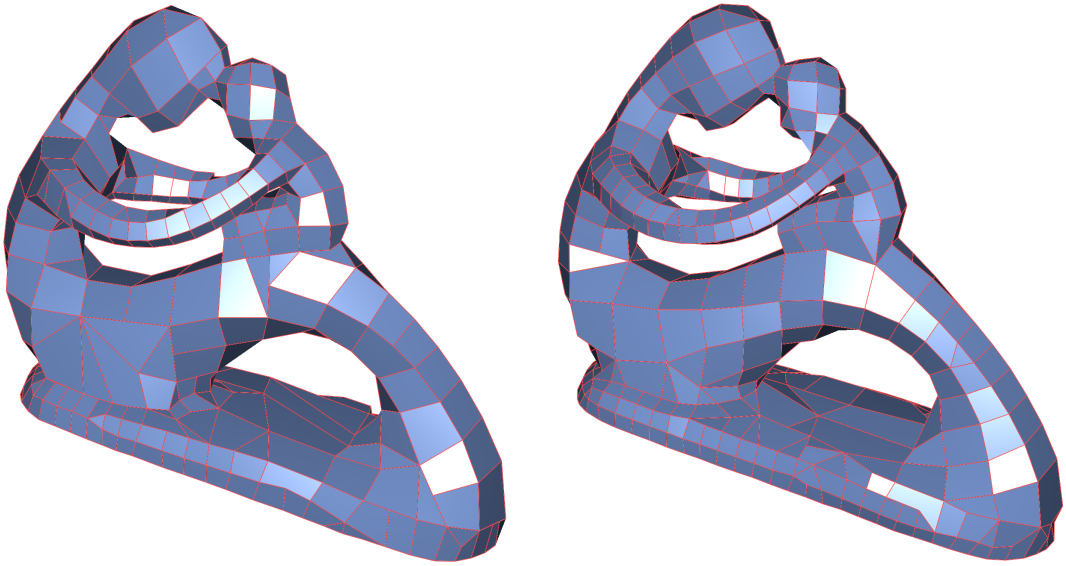


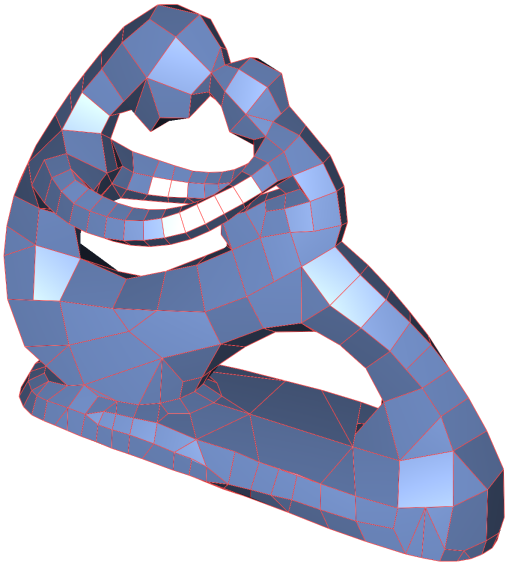
Figure A.3.: This is the mesh that was extracted from the parametrization in Figure 3.8 which displays the effect of the *extend high curvature* averaging method.



(a) Extracted mesh for *normal-based* hierarchy and *curvature summation*.

(b) Extracted mesh for *normal-based* hierarchy and *curvature estimation per level*.

Figure A.4.: These are the resulting extracted meshes from the parametrizations in Figure 3.12. They compare hierarchies with *curvature summation* and on *curvature estimation per level*.



(a) Resulting mesh for *curvature similarity-based* hierarchy and *summed curvature*.

Figure A.5.: This is the extracted mesh from the parametrization in Figure 3.13 which uses a *curvature similarity-based* hierarchy and *summed curvature*.

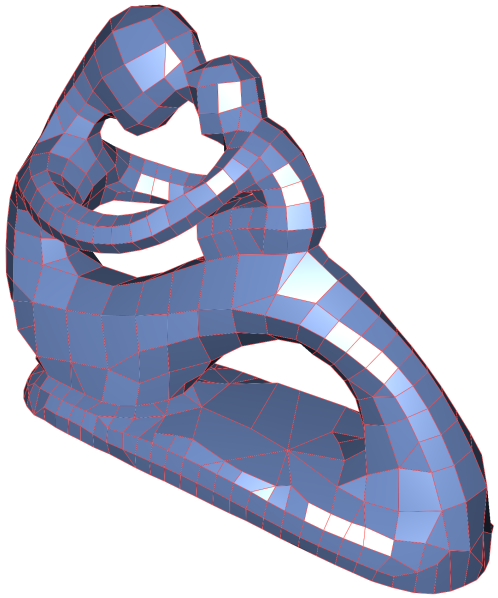
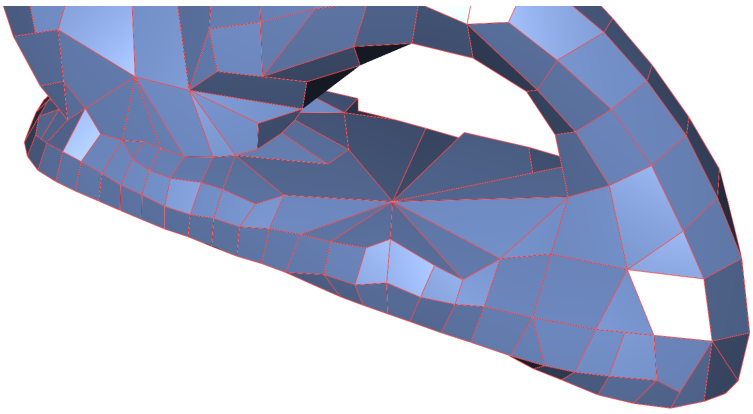
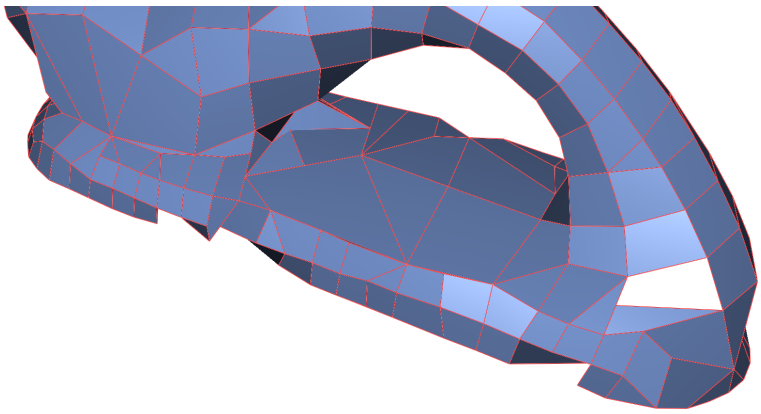


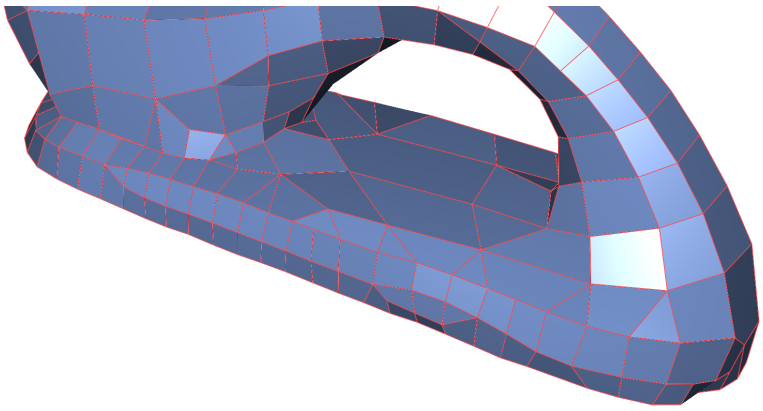
Figure A.6.: This figure shows the extracted mesh for an edge length based on the osculating circle fixed factor formulation with a factor of 0.8. The parametrization this mesh is based on can be seen in Figure 3.18.



(a) Arithmetic mean scaling.



(b) Reference point only scaling.



(c) Per-vertex scaling.

Figure A.7.: These are the extracted meshes for the parametrizations shown in Figure 3.23. They are computed with different position field grid scaling methods.

Bibliography

Alliez, Pierre et al. (2003). "Anisotropic polygonal remeshing." In: *ACM Transactions on Graphics* 22.3, p. 485. ISSN: 07300301. DOI: 10.1145/882262.882296. URL: http://portal.acm.org/citation.cfm?doid=882262.882296 (cit. on pp. 14, 45, 47, 90).

Bommes, David, Lévy Bruno, et al. (2012). "State of the Art in Quad Meshing." In: *Eurographics STARS*. Vol. xx, pp. 1–24. URL: http://hal.inria.fr/INRIA/hal-00804550 (cit. on pp. 5, 7, 8).

Bommes, David, Marcel Campen, et al. (2013). "Integer-grid maps for reliable quad meshing." In: *ACM Transactions on Graphics* 32.4, p. 1. ISSN: 07300301. DOI: 10.1145/2461912.2462014. URL: http://dl.acm.org/citation.cfm?doid=2461912.2462014 (cit. on p. 10).

Bommes, David, Henrik Zimmer, and Leif Kobbelt (2009). "Mixed-integer quadrangulation." In: *ACM Transactions on Graphics* 28.3, p. 1. ISSN: 07300301. DOI: 10.1145/1531326.1531383. URL: http://portal.acm.org/citation.cfm?doid=1531326.1531383 (cit. on pp. 3, 6, 7, 10, 12, 17).

Campen, Marcel, David Bommes, and Leif Kobbelt (2012). "Dual loops meshing." In: *ACM Transactions on Graphics* 31.4, pp. 1–11. ISSN: 07300301. DOI: 10.1145/2185520.2185606. URL: http://dl.acm.org/citation.cfm?doid=2185520.2185606 (cit. on p. 11).

Catmull, Edwin and James Clark (1978). "Recursively generated B-spline surfaces on arbitrary topological meshes." In: *Computer-Aided Design* 10.6, pp. 350–355. ISSN: 00104485. DOI: 10.1016/0010-4485(78)90110-0. URL: http://www.sciencedirect.com/science/article/pii/0010448578901100 (cit. on pp. 1, 5, 67).

Chen, Xin and Francis Schmitt (1992). "Intrinsic surface properties from surface triangulation." In: *Computer Vision — ECCV'92, Lecture Notes in Computer Science, Volume 588*. ECCV '92 1. London, UK, UK: Springer-Verlag, pp. 739–743. ISBN: 978-3-540-55426-4. DOI: 10.1007/3-540-55426-2_83. URL: http://dl.acm.org/citation.cfm?id=645305.648694 (cit. on p. 20).

Cohen-Steiner, David, Pierre Alliez, and Mathieu Desbrun (2004). "Variational shape approximation." In: *ACM Transactions on Graphics*. Vol. 23. SIGGRAPH '04. New York, NY, USA: ACM, p. 905. ISBN: 0730-0301. DOI: 10.1145/1015706.1015817. URL: http://doi.acm.org/10.1145/1186562.1015817 (cit. on p. 12).

Dong, Chen-shi and Guo-zhao Wang (2005). "Curvatures estimation on triangular mesh." In: *Journal of Zhejiang University SCIENCE* 6.Suppl. I, pp. 128–136. ISSN: 1009-3095. DOI: 10.1631/jzus.2005.AS0128. URL: http://www.zju.edu.cn/jzus/2005/A05S1/A05S121.pdf (cit. on pp. 20–22).

Dong, Shen, Peer-Timo Bremer, et al. (2006). "Spectral surface quadrangulation." In: *ACM Transactions on Graphics*. SIGGRAPH '06 25.3, p. 1057. ISSN: 07300301. DOI: 10.1145/1141911.1141993. URL: http://doi.acm.org/10.1145/1141911.1141993 (cit. on p. 8).

Dong, Shen, Scott Kircher, and Michael Garland (2005). "Harmonic functions for quadrilateral remeshing of arbitrary manifolds." In: *Computer Aided Geometric Design* 22.5, pp. 392–423. ISSN: 01678396. DOI: 10.1016/j.cagd.2005.04.004 (cit. on p. 14).

Gatzke, Timothy D. and Cindy M. Grimm (2006). "Estimating Curvature on Triangular Meshes." In: *International Journal of Shape Modeling* 12.01, pp. 1–28. ISSN: 0218-6543. DOI: 10.1142/S0218654306000810. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218654306000810 (cit. on p. 20).

Goldfeather, Jack and Victoria Interrante (2004). "A novel cubic-order algorithm for approximating principal direction vectors." In: *ACM Transactions on Graphics* 23.1, pp. 45–63. ISSN: 07300301. DOI: 10.1145/966131.966134. URL: http://portal.acm.org/citation.cfm?doid=966131.966134 (cit. on pp. 21, 22, 31).

Halstead, Mark, Michael Kass, and Tony D. DeRose (1993). "Efficient, fair interpolation using Catmull-Clark surfaces." In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*. SIGGRAPH '93. New York, NY, USA: ACM, pp. 35–44. ISBN: 0897916018. DOI: 10.1145/166117.166121. URL: http://portal.acm.org/citation.cfm?doid=166117.166121 (cit. on pp. 6, 67).

Hormann, Kai and Günther Greiner (2000). "Quadrilateral remeshing." In: *Proceedings of Vision, Modeling and Vizualization, 2000*, pp. 153–162 (cit. on p. 9).

Huang, Jin et al. (2008). "Spectral quadrangulation with orientation and alignment control." In: *ACM Transactions on Graphics*. SIGGRAPH Asia '08 27.5, p. 1. ISSN: 07300301. DOI: 10.1145/1409060.1409100. URL: http://doi.acm.org/10.1145/1409060.1409100 (cit. on pp. 9, 10).

Jakob, Wenzel et al. (2015). "Instant field-aligned meshes." In: *ACM Transactions on Graphics* 34.6, pp. 1–15. ISSN: 07300301. DOI: 10.1145/2816795.2818078. URL: http://dl.acm.org/citation.cfm?doid=2816795.2818078 (cit. on pp. 2, 15, 18, 19, 32, 38, 45, 48, 50, 52, 54, 57, 64, 77, 81–84, 88, 91).

Kälberer, Felix, Matthias Nieser, and Konrad Polthier (2007). "QuadCover - Surface Parameterization using Branched Coverings." In: *Computer Graphics Forum* 26.3, pp. 375–384. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2007.01060.x. URL: http://doi.wiley.com/10.1111/j.1467-8659.2007.01060.x (cit. on pp. 9, 10).

Kovacs, Denis, Ashish Myles, and Denis Zorin (2011). "Anisotropic quadrangulation." In: *Computer Aided Geometric Design* 28.8, pp. 449–462. ISSN: 01678396. DOI: 10.1016/j.cagd.2011.06.003 (cit. on pp. 11, 90).

Lai, Yu Kun, Leif Kobbelt, and Shi Min Hu (2010). "Feature aligned quad dominant remeshing using iterative local updates." In: *CAD Computer Aided Design* 42.2, pp. 109–117. ISSN: 00104485. DOI: 10.1016/j.cad.2009.02.017. URL: http://dx.doi.org/10.1016/j.cad.2009.02.017 (cit. on pp. 15, 27, 89).

Liu, Yang et al. (2011). "General planar quadrilateral mesh design using conjugate direction field." In: *ACM Transactions on Graphics*. SA '11 30.6, p. 1. ISSN: 07300301. DOI: 10.1145/2070781.2024174. URL: http://doi.acm.org/10.1145/2070781.2024174 (cit. on p. 10).

Marinov, Martin and Leif Kobbelt (2004). "Direct anisotropic quad-dominant remeshing." In: *Proceedings - Pacific Conference on Computer Graphics and Applications*. PG '04. Washington, DC, USA: IEEE Computer Society, pp. 207–216. ISBN: 0769522343. DOI: 10.1109/PCCGA.2004.1348351. URL: http://dl.acm.org/citation.cfm?id=1025128.1026044 (cit. on p. 14).

Marinov, Martin and Leif Kobbelt (2006). "A robust two-step procedure for quad-dominant remeshing." In: *Computer Graphics Forum* 25.3, pp. 537–546. ISSN: 01677055. DOI: 10.1111/j.1467-8659.2006.00973.x. URL: http://dblp.uni-trier.de/db/journals/cgf/cgf25.html%7B%5C#%7DMarinovK06 (cit. on pp. 12, 78, 79).

MATLAB (2017). *Version 9.2.0.538062 (R2017a)*. Natick, Massachusetts (cit. on p. 81).

Petitjean, Sylvain (2002). "A survey of methods for recovering quadrics in triangle meshes." In: *ACM Computing Surveys* 34.2, pp. 211–262. ISSN: 03600300. DOI: 10.1145/508352.508354. URL: http://portal.acm.org/citation.cfm?doid=508352.508354 (cit. on p. 20).

Ray, Nicolas, Wan Chiu Li, et al. (2006). "Periodic global parameterization." In: *ACM Transactions on Graphics* 25.4, pp. 1460–1485. ISSN: 07300301. DOI: 10.1145/1183287.1183297. URL: http://portal.acm.org/citation.cfm?doid=1183287.1183297 (cit. on p. 13).

Ray, Nicolas, Bruno Vallet, et al. (2008). "N-symmetry direction field design." In: *ACM Transactions on Graphics* 27.2, pp. 1–13. ISSN: 07300301. DOI: 10.1145/1356682.1356683. URL: http://portal.acm.org/citation.cfm?doid=1356682.1356683 (cit. on p. 16).

Rugis, John and Reinhard Klette (2006). "A scale invariant surface curvature estimator." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 4319 LNCS, pp. 138–147. ISSN: 03029743. DOI: 10.1007/11949534-14 (cit. on p. 48).

Shen, Jingjing et al. (2014). "Conversion of trimmed NURBS surfaces to Catmull-Clark subdivision surfaces." In: *Computer Aided Geometric Design* 31.7, pp. 486–498. ISSN: 01678396. DOI: 10.1016/j.cagd.2014.06.004 (cit. on p. 6).

Stam, Jos (1998). "Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values." In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. SIGGRAPH '98. New York, NY, USA: ACM, pp. 395–404. ISBN: 0897919998. DOI: 10.1145/280814.280945. URL: http://portal.acm.org/citation.cfm?doid=280814.280945 (cit. on pp. 1, 6).

Thaller, Wolfgang, Ursula Augsdörfer, and Dieter W. Fellner (2016). "Procedural mesh features applied to subdivision surfaces using graph grammars." In: *Computers and Graphics (Pergamon)* 58, pp. 184–192. ISSN: 0097-8493. DOI: 10.1016/j.cag.2016.05.014 (cit. on pp. 2, 6, 66).

Tong, Yiying et al. (2006). "Designing Quadrangulations with Discrete Harmonic Forms." In: *Eurographics Symposium on Geometry Processing*. SGP '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, pp. 201–210. ISBN: 30905673-36-3. URL: http://dl.acm.org/citation.cfm?id=1281957.1281983 (cit. on p. 8).

Zhang, Muyang et al. (2010). "A wave-based anisotropic quadrangulation method." In: *ACM Transactions on Graphics*. SIGGRAPH '10 29.4, p. 1. ISSN: 07300301. DOI: 10.1145/1833351.1778855. URL: http://portal.acm.org/citation.cfm?doid=1833351.1778855 (cit. on pp. 9, 89).

Zhang, Muyang et al. (2013). "A divide-and-conquer approach to quad remeshing." In: *IEEE Transactions on Visualization and Computer Graphics* 19.6, pp. 941–952. ISSN: 10772626. DOI: 10.1109/TVCG.2012.301. URL: http://dx.doi.org/10.1109/TVCG.2012.301 (cit. on p. 11).