Geometric Modeling for Adaptive Surface Reconstruction of Computed Tomography Data

Vom Fachbereich Informatik der

Technischen Universität Kaiserslautern

zur Verleihung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Dennis Mosbach

Datum der wissenschaftlichen Aussprache: 22.05.2020

D 386

Abstract

B-spline surfaces are a well-established tool to analytically describe objects. They are commonly used in various fields, e.g., mechanical and aerospace engineering, computer aided design, and computer graphics. Obtaining and using B-spline surface models of real-life objects is an intricate process. Initial virtual representations are usually obtained via scanning technologies in the form of discrete data, e.g., point clouds, surface meshes, or volume images. This data often requires pre-processing to remove noise and artifacts. Even with high quality data, obtaining models of complex or very large structures needs specialized solutions that are viable for the available hardware. Once B-spline models are constructed, their properties can be utilized and combined with application-specific knowledge to provide efficient solutions for practical problems.

This thesis contributes to various aspects of the processing pipeline. It addresses pre-processing, creating B-spline models of large and topologically challenging data, and the use of such models within the context of visual surface inspection. Proposed methods improve existing solutions in terms of efficiency, hardware restrictions, and quality of the results. The following contributions are presented:

Fast and memory-efficient quantile filter: Quantile filters are widely used operations in image processing. The most common instance is the median filter which is a standard solution to treat noise while preserving the shape of an object. Various implementations of such filters are available offering either high performance or low memory complexity, but not both. This thesis proposes a generalization of two existing algorithms: one that favors speed and one that favors low memory usage. An adaptable hybrid algorithm is introduced. It can be tuned for optimal performance on the available hardware. Results show that it outperforms both state-of-the-art reference methods for most practical filter sizes.

Robust B-spline reconstructions of isosurfaces in volume images: The micro-structure of wood-based thermal insulation materials is analyzed to research heat conductivity properties. High-quality scans reveal a complex system of cellulose fibers. B-spline models of individual fibers are highly desirable to conduct simulations. Due to the physical processing of the material, the surfaces of those fibers consist of challenging elements like loose filaments, holes, and tunnels. Standard solutions fail to partition the data into a small number of quadrilateral cells which is required for the B-spline construction step. A novel approach is presented that splits up the data processing into separate topology and geometry pipelines. This robust method is demonstrated by constructing B-spline models with 236 to 676 surfaces from triangulated isosurfaces with 423 628 to 1 203 844 triangles.

Local method for smooth B-spline surface approximations: Constructing smooth B-spline models to approximate discrete data is a challenging task. Various standard solutions exist, often imposing restrictions to knot vectors, spline order, or available degrees of freedom for the data approximation. This thesis presents a local approach with less restrictions aiming for approximate G^1-continuity. Nonlinear terms are added to standard minimization problems. The local design of the algorithm compensates for the higher computational complexity. Results are shown and evaluated for objects of varying complexity. A comparison with an exact G^1-continuous method shows that the novel method improves approximation accuracy on average by a factor of 10 at the cost of having small discontinuities in normal vectors of less than 1 degree.

Model-based viewpoint generation for surface inspection: Within modern and flexible factories, surface inspection of products is still a very rigid process. An automated inspection system requires the definition of viewpoints from which a robot then takes pictures during the inspection process. Setting up such a system is a time-intensive process which is primarily done manually by experts. This work presents a purely virtual approach for the generation of viewpoints. Based on an intuitive definition of analytic feature functionals, a non-uniform sampling with respect to inspection-specific criteria is performed on given B-spline models. This leads to the definition of a low number of viewpoint candidates. Results of applying this method to several test objects with varying parameters indicate that good viewpoints can be obtained through a fast process that can be performed fully automatically or interactively through the use of meaningful parameters.

Acknowledgements

The research and writing of this thesis were conducted at the Image Processing Department of the Fraunhofer Institute for Industrial Mathematics (Fraunhofer ITWM) and the Computer Graphics and HCI group at the University of Kaiserslautern (TUK). It was funded by the Fraunhofer High Performance Center for Simulation- and Software-Based Innovation and supported by the German Research Foundation (DFG) within the IRTG 2057 "Physical Modeling for Virtual Manufacturing Systems and Processes", the German Federal Ministry of Education and Research through project 05M2013 (AniS), as well as a grant by the Deutsch-Französische Hochschule (DFH). The wood insulation investigation in Chapter 4 was funded by the German Federal Ministry of Food and Agriculture (BMEL) through its Agency for Renewable Resources, project LowLambda.

Performing the research, implementation of related and new algorithms, and writing this thesis was a lengthy and extensive process that would not have been possible without the help and support of many people.

In particular, I would like to thank my academic advisers Prof. Dr. Hans Hagen and Prof. Dr. Bernd Hamann who were always accessible whenever I needed help and supported me with their knowledge, guidance, and experience. I thank my co-adviser Dr. Katja Schladitz who was willing to familiarize herself with new topics and always provided helpful feedback, even in busy times. I also thank Dr. Oliver Wirjadi, my co-adviser from the time I started with this project. Through his advice, ideas, and long term visions, he helped create the foundation of this thesis.

Furthermore, I would like to thank my colleague Petra Gospodnetić for the encouragement, scientific discussions over coffee, and constructive feedback. Working together on various challenges was a fun and fruitful experience. I also thank my office mate Dr. Markus Kronenberger for the many productive conversations as well as motivation and fresh ideas when I got stuck at a problem, Michael Godehardt for the insightful discussions about coding-related topics, Dr. Jonas Lukascyk for his creative problem solving approaches, and Meghan Miller for proofreading and providing valuable feedback.

I am grateful for the pleasant work environment and positive attitude of all the people of the Image Processing Department, the HCI and Computer Graphics Group, and the members of the IRTG2057.

Finally, I thank my family and friends for their constant support, encouragement, and entertaining distractions especially during stressful times.

Table of Contents

Chapter 1

Introduction

Generating virtual models of real-world objects is a fundamental part and of great importance in many application-driven scientific fields. These models should consist of simple data structures, be easy to understand, and provide a good description of the real-world object. Meaningful representations are required for various tasks [47, 61, 82], e.g., visualization, simulation, analysis, and quality assurance. This thesis focuses on modeling three-dimensional (3D) objects by describing their surfaces. Various types of models are suitable within this context.

A widely used representation is the *explicit discrete model*. Common instances are surface meshes [22] and point clouds [41]. Such models consist of a finite set of points, describing an object's surface geometry. In a mesh, these points are additionally connected via edges and polygonal faces, defining the topology. A mesh consisting of purely triangular faces is called *triangular mesh*. Such meshes are widely used in computer graphics and engineering disciplines, e.g., raytracing [81] or finite element methods [86].

An object can also be represented by a 3D image, i.e., an integer lattice with color or intensity values assigned to lattice points [57]. This is called an *implicit discrete model*. The surface is implicitly given by a single number, called *isovalue*. Lattice points are classified as inside and outside based on whether the corresponding pixel value is higher or lower than the isovalue. Such an implicit surface can easily be converted into an explicit mesh representation via contouring methods, e.g., Marching Cubes [46].

Furthermore, objects can be described through *continuous models*, i.e., analytical descriptions via functions. This is often done by using parametric surfaces [21]. Those functions map each point in a parameter domain $\Omega \in \mathbb{R}^2$ to a surface point in world coordinates (usually $\mathbb{R}^3$). The continuous nature of these surfaces allows for the calculation of derivatives and integrals, either analytically or using numeric methods. Continuous models can be converted into a discrete representation by sampling them with arbitrary resolution to create a finite number of surface points.

State-of-the-art editing software allows users to manually model objects of various shapes both in continuous or discrete form. Numerous modeling tools are freely available and open-source projects are continuously developed and improved, e.g., MeshLab [9] and Blender [5]. However, a manual approach is mostly useful when designing new objects. Modeling existing real-world objects accurately by hand is generally only possible for simple structures.

Virtual representations of complex objects are typically acquired via 3D scanning. Data acquisition methods, like laser scanning [44] or computed tomography (CT) [33], generate discrete models. However, they often including noise and other artifacts that may cause difficulties for the subsequent processing. Depending on the resolution of the scanning hardware, storing the resulting models can require large amounts of memory. Lowering the resolution might result in the loss of important features. Furthermore, the computation of geometric properties, e.g., normal vectors or curvature, strongly depend on the quality of the discretization.

For these reasons, a representation with parametric surfaces is often desirable. However, obtaining those continuous models is a non-trivial task. Many methods have been researched to reverse engineer various types of parametric surfaces.

The focus of this thesis is the generation and use of B-spline surfaces. These surfaces are a specific type of parametric surfaces and are commonly used in various fields, e.g., mechanical and aerospace engineering, computer aided design (CAD), and computer graphics. A B-spline surface consists of piecewise polynomial pieces of a predefined order. Its shape is defined by a grid of control points. Single B-spline surfaces are capable of representing small and complex regions with high precision, but can also be used to define large and mostly flat surfaces with the same amount of control points. A *B-spline surface model* consists of multiple continuously connected B-spline surfaces. It can represent complex objects, combining regions with distinct features and large flat parts, at low memory costs. Furthermore, B-splines are widely used in industry. Hence, B-spline surface models are compatible with a wide range of production, simulation, and analysis tools.

This thesis presents various improvements to the process of generating and using B-spline models. Contributions include pre-processing of discrete input data, generating B-spline models when faced with difficult topological situations and large data, and using the properties of the resulting B-spline models to gain application-specific knowledge.

First, Chapter 2 reviews the relevant concepts related to B-spline surfaces. Additionally, fundamental methods to construct B-spline models from discrete data are discussed. Important equations are stated in a unified notation and with a focus on easy implementation. This chapter summarizes relevant literature and existing research. All subsequent chapters present new contributions.

Chapter 3 covers a novel hybrid approach for quantile filtering of 3D image data. It is based on two existing solutions which are either fast or memory efficient. The proposed algorithm interpolates between these solutions. This allows for a compromise between speed and memory requirements. As a consequence, the method can optimally adapt to the available hardware.

Often, even when presented with scans of complex structures, B-spline models are difficult to construct. For example, a complicated surface topology does not allow for a good partition into quadrilateral cells which is required for the subsequent B-spline approximation step. Chapter 4 discusses an example in which cellulose fibers are reconstructed from 3D images. Such models are needed for various applications in material science [3, 47, 61]. The data is available in the form of high-resolution CT scans without noise or artifacts. However, during manufacturing, the material has undergone physical processing causing damage to the fibers' surfaces. The presented method employs a topological simplification to partition the data into a low number of cells while still using the original geometry for the B-spline approximation step. This stable reconstruction scheme allows for the removal of unwanted small holes in the fiber walls while preserving the overall shape. It provides models with low memory requirements that are well-suited for the subsequent simulation process.

When it comes to constructing B-spline models with smooth transitions between individual surfaces, several approaches exist, but these impose strong restrictions to the available degrees of freedom [20, 60]. Chapter 5 presents an alternative approach for the construction of models with approximate tangent plane continuity along surface boundaries. Instead of formulating restrictive conditions that have to be met, smoothness is achieved via nonlinear optimization. To counter the high computational complexity that is required to solve these problems, the method is designed to work locally. An additional benefit from such an approach is that it can easily deal with large, even incomplete, input data.

Finally, Chapter 6 demonstrates the use of B-spline models within the context of automatic surface inspection. The task is to determine good camera positions for a given real-world object with the constraint that all relevant parts of the object are sufficiently covered. Those positions are then traversed automatically using robot arms. The taken pictures are automatically processed with image processing tools to find defects. However, defining camera positions and orientations for the robot is a non-trivial task. When done manually by an expert, it takes days, often even weeks, to set up such a system. Automatic approaches based on discrete (mesh) models offer partial solutions, but still struggle with computational complexity when aiming for optimal results. Using the mathematical properties of a B-spline surface model, a novel method is presented that produces a low number of meaningful viewpoint candidates. The distribution of these viewpoints is controlled through so-called feature functionals that measure the prominence of certain geometric features within a given area of the surface. Several practical feature functionals are described and analyzed that allow experts to factor in application-specific knowledge. An optimal subset of final viewpoints for the inspection is then found using a straightforward greedy approach.

Chapter 2

Fundamental Methods for Constructing B-spline Surface Approximations

A key component used throughout this thesis is the construction of B-spline surface models to approximate objects given in the form of discrete data. This task occurs in many variations and a wide range of specialized solutions exists [17, 20, 27, 42, 45, 85]. The fundamental principles are well-researched and covered in depth by various textbooks, e.g., [11, 21, 32, 62, 64]. This chapter summarizes the key concepts and states necessary equations for the relevant steps in a unified notation. Formal definitions can be found in literature.

A B-spline surface is defined by an order $k \in \mathbb{N}$, a knot vector for each parameter direction, and a rectangular grid of control points. To keep the construction of a model with multiple continuously connected surfaces as simple as possible, the B-splines considered here are restricted to quadratic grids of $n_c \times n_c$ control points $b_{i,j} \in \mathbb{R}^3$ and use the same knot vector $\tau = (\tau_1, ..., \tau_{n_c+k+1})$ in both u and v direction with increasing knots $\tau_i \leq \tau_{i+1}$. The piecewise polynomial basis functions of order k associated with knot vector τ are defined recursively by

$$N_{i,0,\tau}(t) = \begin{cases} 1, & \text{if } t \in [\tau_i, \tau_{i+1}) \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,k,\tau}(t) = \frac{t - \tau_i}{\tau_{i+k} - \tau_i} N_{i,k-1,\tau}(t) + \frac{\tau_{i+k+1} - t}{\tau_{i+k+1} - \tau_{i+1}} N_{i+1,k-1,\tau}(t).$$

These basic functions have several useful properties:

- *Non-negativity*: $N_{i,k,\tau}(t) \geq 0$
- *Local support*: $N_{i,k,\tau}(t) = 0$, if $t \notin [\tau_i, \tau_{i+k+1})$

- *Decomposition of one*: $\sum_{i=1}^{n_c} N_{i,k,\tau}(t) = 1$, for $t \in [\tau_{k+1}, \tau_{n_c+1})$

- *Derivatives*: $\frac{d}{dt}N_{i,k,\tau}(t) = \frac{k}{\tau_{i+k}-\tau_i}N_{i,k-1,\tau}(t) - \frac{k}{\tau_{i+k+1}-\tau_{i+1}}N_{i+1,k-1,\tau}(t)$ for $k > 1$.

Since k and τ are usually given constants, they are omitted for a simpler notation, i.e., $N_i(t) = N_{i,k,\tau}(t)$. Possible divisions by zero in the presence of multiple knots are avoided by defining $0/0 = 0$. Basis functions until order $k = 3$ are shown in Figure 2.1.

The B-spline surface is given by

$$\begin{aligned} S :&\Omega \to \mathbb{R}^3 \\ S(u,v) =&\sum_{i=1}^{n_c}\sum_{j=1}^{n_c} N_i(u)N_j(v)b_{i,j}. \end{aligned}$$

with a rectangular parameter region $\Omega = [u_{\text{min}}, u_{\text{max}}] \times [v_{\text{min}}, v_{\text{max}}] \subset \mathbb{R}^2$. Throughout this thesis, the unit square is used as parameter region, i.e., $\Omega = [0,1]^2$. An example surface is shown in Figure 2.2. Important properties of B-spline surfaces are:

- *Local support*: Changing one control point $b_{i,j}$ affects only the surface segment $\{S(u,v) \,|\, u \in [\tau_i, \tau_{i+k+1}), v \in [\tau_j, \tau_{j+k+1})\}$.

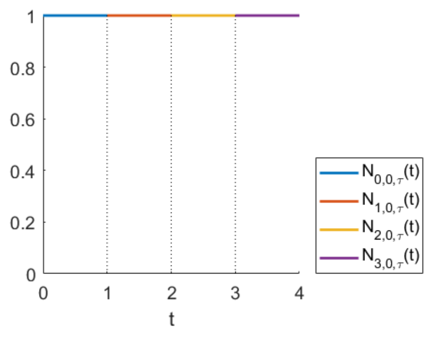
- *Convex hull property*: Every point on the surface $S(u,v)$ lies within the convex hull of the control points $b_{i,j}$.

- *Affine invariance*: Applying an affine transformation to the control points has the same effect as applying this transformation to the surface.

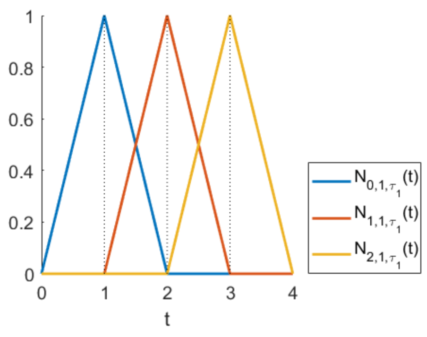
The curve obtained by evaluating a surface along an edge of its parameter domain is called boundary curve. The four boundary curves are denoted by $C^{(1)}(t) = S(t,0)$, $C^{(2)}(t) = S(1,t)$, $C^{(3)}(t) = S(t,1)$, $C^{(4)}(t) = S(0,t)$ with $t \in [0,1]$.

When using end knots with multiplicity $k+1$, the B-spline surface interpolates the B-spline curves defined by the outer rows of control points. This property is essential for the design of continuous B-spline models comprised of multiple surfaces. Hence, the knot vectors used here are chosen as

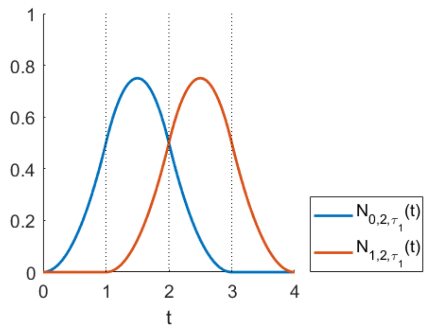
$$\tau = (\underbrace{0, ..., 0}_{k+1 \text{ times}}, \tau_{k+2}, ..., \tau_{n_c}, \underbrace{1, ..., 1}_{k+1 \text{ times}}) \in [0,1]^{n_c+k+1}. \tag{2.1}$$



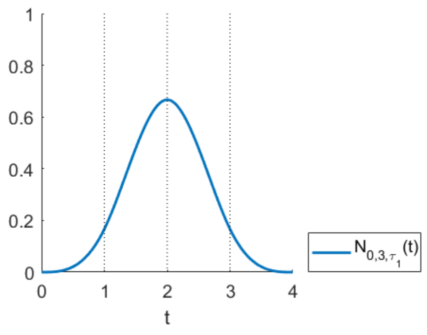
(a) *Basis functions of order* $k = 0$



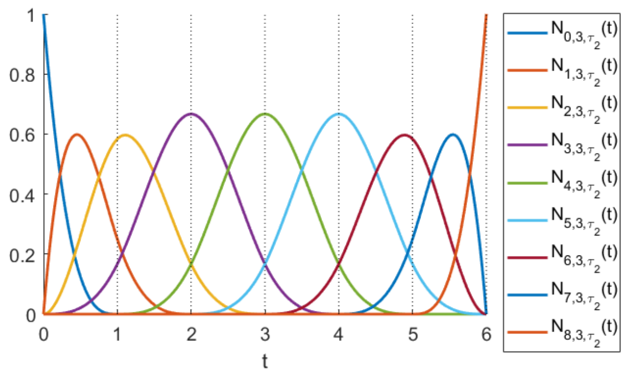
(b) *Basis functions of order* $k = 1$



(c) *Basis functions of order* $k = 2$



(d) *Basis function of order* $k = 3$



(e) *Full set of basis functions of order* $k = 3$

Figure 2.1: *B-spline basis functions. (a)-(d) show the intermediate steps for the recursive definition of a single basis function of order* 3 *with uniform knot vector* $\tau_1 = (0, 1, 2, 3)$*. A full set of* 9 *basis functions of order* 3 *for a larger knot vector* $\tau_2 = (0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6)$ *is shown in (e).*

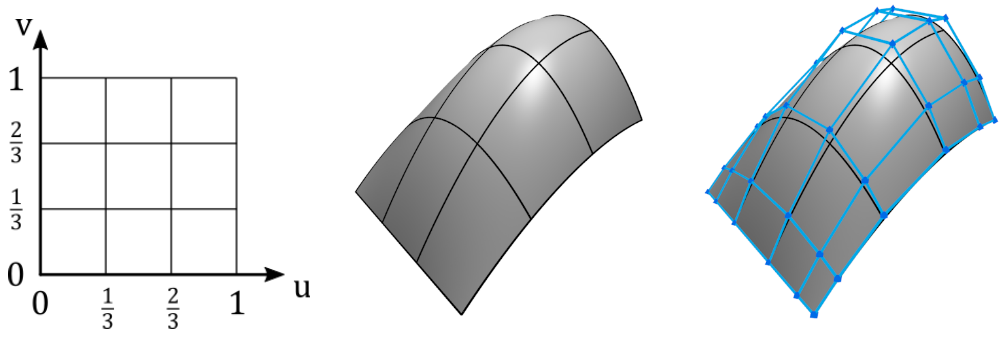


Figure 2.2: *B-spline surface of order* $k = 3$, 6×6 *control points, and knot vector* $\tau = (0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1)$. *The unit square in* $\mathbb{R}^2$ *(a) is mapped to a surface in* $\mathbb{R}^3$ *(b). Isoparametric lines corresponding to the individual knots* τ_i *are drawn in black. The* 6×6 *grid of control points is shown in (c)*

.

This allows for the explicit expression of the boundary curves as

$$C^{(1)}(t) = \sum_{i=1}^{n_c} N_i(t) b_{i,1}$$

$$C^{(2)}(t) = \sum_{j=1}^{n_c} N_j(t) b_{n_c,j}$$

$$C^{(3)}(t) = \sum_{i=1}^{n_c} N_i(t) b_{i,n_c}$$

$$C^{(4)}(t) = \sum_{j=1}^{n_c} N_j(t) b_{1,j}.$$

Figure 2.3 illustrates this concept.

When modeling objects with multiple surfaces, incident B-spline surfaces need to satisfy certain continuity requirements along their boundaries. The most common conditions are C^0- and G^1-continuity which are demonstrated in Figure 2.4.

Two B-spline surfaces are C^0-continuously connected, if they have a shared boundary curve. When using knot vectors with multiple ends as in Equation (2.1), C^0-continuity between two surfaces is achieved if and only if the control points of both surfaces are equal along the shared boundary.

Furthermore, two neighboring surfaces are G^1-continuously connected if they are C^0-continuously connected and there are no discontinuities in tangent planes along the

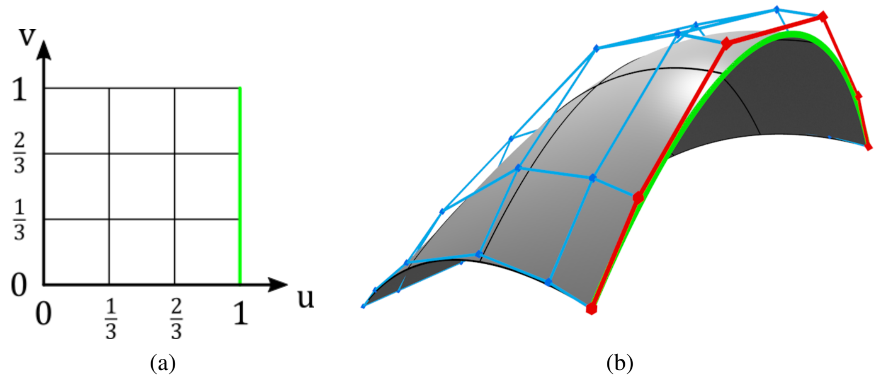


Figure 2.3: *Using a knot vector with $k+1$ multiple ends leads to interpolation of boundary curves. (a) shows a boundary of the parameter domain highlighted in green. The corresponding curve on the B-spline surface only depends on the outer row of control points, colored in red (b).*

shared boundary curve, i.e., at any point on this boundary the (normalized) normal vectors of both surfaces are equal.

A single B-spline surface is usually not sufficient to represent a complex object as it can support only disk-like surface topology. Therefore, a number of B-spline surfaces is needed, each modeling a part of the object. This means that the given data first needs to be partitioned into a number of quadrilateral cells, a so-called *quad-mesh*. Each cell needs to have disk-like topology so that one surface can later be used to approximate it. Furthermore, each data point is assigned a two-dimensional parameter value which indicates the approximate relative position of the data point within its containing cell. This is required to reduce the number of unknowns in the approximation step and avoid complicated nonlinear equations.

Optionally, the data points can have weights assigned to them. Weights can be chosen manually or based on statistical properties of the data. They allow to control how much a point contributes to the shape of the surface and becomes especially relevant when dealing with unevenly distributed data.

In total, the following is needed to start the B-spline approximation process:

1. A finite set of discrete data points $P \subset \mathcal{R}^3$
2. A quad-mesh that
   (a) defines a partition of the data into n_s quadrilateral cells $P^{(1)}, \ldots, P^{(n_s)}$ with $\bigcup_{j=1}^{n_s} P^{(j)} = P$ and
   (b) provides a topological neighborhood structure to the cells, i.e., identifies shared boundaries and corners.

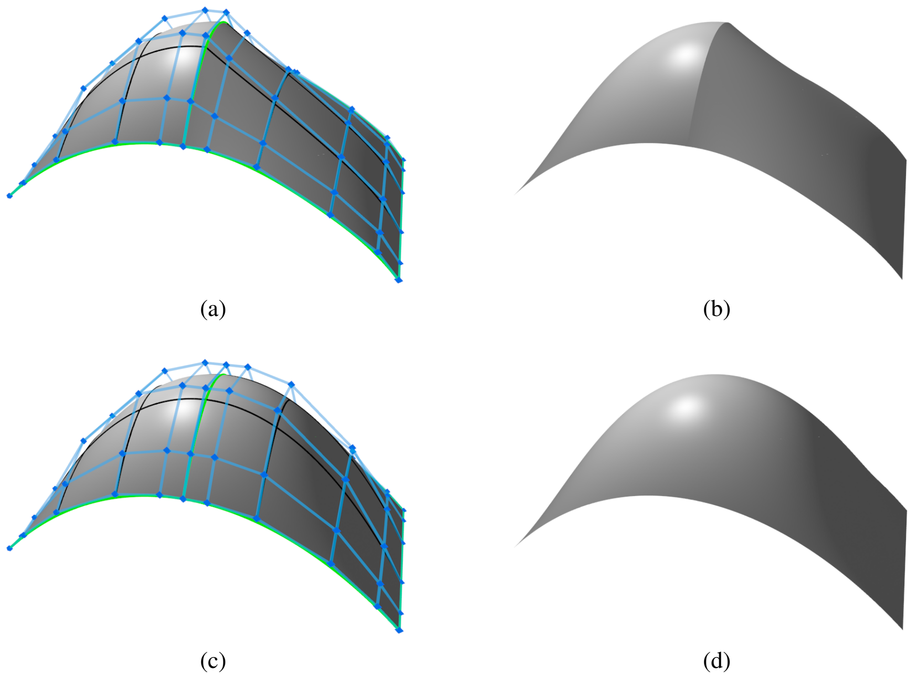


Figure 2.4: *Different types of continuity. (a) shows two B-spline surfaces with a C^0-continuous, but not G^1-continuous, connection. The discontinuity in normal vectors is clearly visible along the shared boundary curve (b). Two G^1-continuously connected surfaces (c) have no visible discontinuities in their shading (d).*

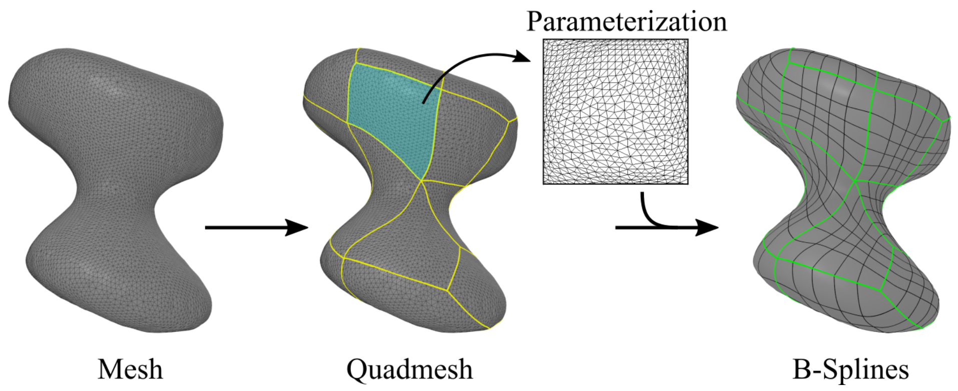


Figure 2.5: *Construction of a B-spline surface model approximating a triangulated surface. First, the input mesh is partitioned into a collection of quadrilateral cells. Each cell is parameterized over a square region providing a correlation between discrete data points and their approximate locations on the B-spline surface. The B-spline model is obtained by minimizing distances between the data points and B-spline surfaces using a least squares approach.*

3. For each cell j, a parameterization of the contained data points, i.e., each point $p_i \in P^{(j)}$ has a parameter value $(u_i, v_i) \in [0, 1]^2$.

4. A weight $w_i \in \mathbb{R}$ for each data point $p_i \in P$ *(optional)*

In this chapter, it is assumed that the initial input is provided in the form of a triangle mesh. The data points used for the B-spline approximation are the vertices of that mesh. Furthermore, the topological information, encoded in the mesh, is relevant for the computation of the quad-mesh for which Section 2.1 describes a selection of methods. The subsequent parameterization of the data is described in Section 2.2. If the input is provided in the form of a point cloud, it can easily be converted into a mesh by freely available tools, e.g., MeshLab [9]. Objects that are given implicitly as isosurface of a 3D volume image can be extracted as meshes with the Marching Cubes Algorithm [46], which is available in MAVI [24] or CGAL [78].

Once the initial data has been processed and the required input is available, Section 2.3 describes how to construct a B-spline model that approximates the given data points. Figure 2.5 illustrates the entire process.

2.1 Quad-mesh Generation

In order to compute a B-spline model to approximate a triangulated surface, the surface first needs to be subdivided into a collection of four-sided regions (quadrilaterals) with disk-like topology. For small or geometrically simple objects, this can be done manually, e.g., by defining a set of cutting planes or simply drawing boundary curves on the surface using CAD tools. When objects are too complex to be processed manually, an automatic method is needed.

Eck and Hoppe [17] describe a method to compute such a partition into quadrilateral cells. Their algorithm starts by computing a surface Voronoi tessellation, ensuring that all cells have disk-like topology. Then, the dual Delaunay-like complex is constructed, consisting of triangular cells. Finally, these cells are merged pairwise into quadrilateral cells. This approach provides good results and can be implemented easily.

Dong et al. [14] propose an approach based on surface harmonics. They compute eigenfunctions of the discrete Laplacian mesh operator. For a triangle mesh $\mathcal{M}$ with n_v vertices this operator can be expressed by a matrix $L \in \mathbb{R}^{n_v \times n_v}$ with coefficients

$$L_{i,j} = \begin{cases} \sum_{(i,k)\in\text{Edges}(\mathcal{M})} \omega_{i,k}, & \text{if } i = j \\ -\omega_{i,j}, & \text{if } (i,j) \in \text{Edges}(\mathcal{M}) \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Here, Edges$(\mathcal{M})$ is the set of all edges of the given mesh, denoted as tuples (i, j) of the incident vertex indices. The method uses the discrete harmonic weights [63] given by

$$\omega_{i,j} = \frac{1}{2}\left(\cot(\alpha_{i,j}) + \cot(\beta_{i,j})\right) \tag{2.3}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ denote the angles opposite to the edge (i, j) within the two incident triangles. The local extrema of these (scalar) eigenfunctions are usually evenly distributed across the surface. Constructing a Morse-Smale complex, i.e., connecting the extrema along gradient flows, leads to a quadrilateral partitioning of the surface. Topological clean-up methods on the Morse-Smale complex can be utilized to eliminate small cells in the presence of high-frequency noise. An open-source implementation of this algorithm is available within the Topology Toolkit [79].

Jakob et al. [37] present an algorithm, called Instant Meshes, that works locally and produces shape aligned quad-meshes. It is based on the computation of an orientation and position field. The orientation field defines the alignment of the quad-edges. It assigns each vertex of the input mesh a 90 degree rotationally symmetric cross which is derived from principle curvatures. By minimizing a smoothness term, these crosses define feature oriented field lines with few singularities. The position field determines the placement of the quad-vertices by mapping each triangle vertex to a two-dimensional lattice in its tangent. The edge length of these lattices is defined by the user and corresponds to the desired length of the quad-edges. Lattices are translated in their local coordinate system by a fraction of their edge length to nearly coincide with the lattices of nearby vertices. These translations define a local parameterization which is smooth and consistent with the orientation field. It is then used to guide a successive edge collapsing that leads to the final quad-mesh.

The Quadriflow algorithm, presented by Huang et al. [34], is based on the idea of Instant Meshes. It uses the same locally computed orientation and position fields and reduces singularities by adding linear and quadratic constraints. A global minimum-cost network flow problem in combination with local boolean satisfiability problems are solved to ensure that these constraints are met. Due to the low computational complexity of these problems, they can be solved very quickly, even for large meshes.

Freely available implementations for both Instant Meshes and Quadriflow are provided by the authors [34, 37].

2.2 Parameterization

The discrete data points inside each quadrilateral cell need to be correlated to the respective B-spline surface. This is done by computing a parameterization that maps each data point to the same parameter domain over which the B-spline surfaces are defined. Parameterizations are computed for each cell individually.

A common way to parameterize data is via discrete harmonic maps [16]. Let n_p be the number of data points (vertices) within the given cell. Furthermore, let $h : \mathbb{N} \to \mathbb{R}^2$ denote the mapping that assigns each vertex $i \in \{1, ..., n_p\}$ a parameterization $(u_i, v_i) \in [0, 1]^2$. An energy term for this function is defined as

$$E_{\text{harm}}[h] = \frac{1}{2} \sum_{(i,j) \in \text{Edges}(\mathcal{M})} \omega_{i,j} \|h(i) - h(j)\|^2. \tag{2.4}$$

Using the Laplacian operator, see Equation (2.2), its gradient is given by

$$\nabla E_{\text{harm}}[h] = L\overline{h} \tag{2.5}$$

where

$$\overline{h} = \begin{bmatrix} h(1)^T \\ \vdots \\ h(n_p)^T \end{bmatrix} \in \mathbb{R}^{n_p \times 2}.$$

The weights $\omega_{i,j}$ are the harmonic weights, defined by Equation (2.3).

A harmonic map can be constructed in two steps. First, the vertices on the boundary of the quadrilateral cell are mapped onto the boundary of the unit square, such that the ratio of their Euclidean distances in $\mathbb{R}^3$ is preserved. Then, the remaining degrees of freedom are determined by minimizing $E_{\text{harm}}[h]$. Since this term is quadratic with respect to the unknowns, the minimum is achieved when the gradient (2.5) vanishes,

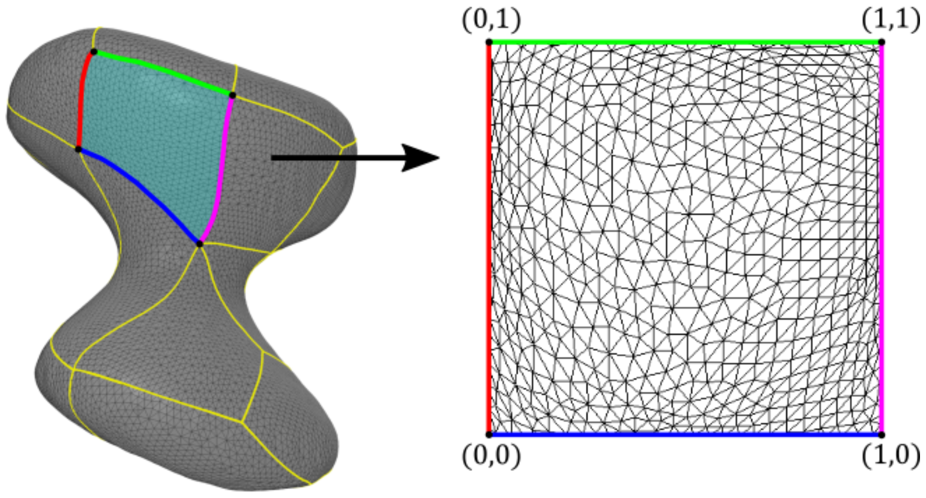


Figure 2.6: *Parameterization of a single quad-cell via a harmonic map. The boundary of the cell is mapped onto the boundary of the unit square. Then, the parameter values of inner vertices are determined by minimizing the energy E_{harm}.*

i.e., $L\overline{h} = 0$. Note that some entries of $\overline{h}$ are already set by the boundary mapping. Without loss of generality, let the indices of the data be ordered such that

$$\overline{h} = \begin{bmatrix} \overline{h}_{\mathrm{i}} \\ \overline{h}_{\mathrm{b}} \end{bmatrix}, L = [L_{\mathrm{i}}\ L_{\mathrm{b}}]$$

with $\overline{h}_{\mathrm{b}}$ denoting the (fixed) parameter values on the boundary and $\overline{h}_{\mathrm{i}}$ the (unknown) interior parameterization. Then, the values of $\overline{h}_{\mathrm{i}}$ can be found by solving

$$\begin{aligned} L\overline{h} &= [L_{\mathrm{i}}\ L_{\mathrm{b}}] \begin{bmatrix} \overline{h}_{\mathrm{i}} \\ \overline{h}_{\mathrm{b}} \end{bmatrix} \\ &= L_{\mathrm{i}}\overline{h}_{\mathrm{i}} + L_{\mathrm{b}}\overline{h}_{\mathrm{b}} \\ &= 0 \\ \Leftrightarrow L_{\mathrm{i}}\overline{h}_{\mathrm{i}} &= -L_{\mathrm{b}}\overline{h}_{\mathrm{b}}. \end{aligned}$$

This procedure can be interpreted as a physical system of ideal mechanical springs [16]. The mesh vertices correspond to free particles without mass which are connected by springs representing the mesh edges. When the boundary is fixed to a planar square, the spring forces cause the free particles to be distributed within the square such that the overall system assumes a state of equilibrium, where the total forces on each interior particle sum up to zero. An example is shown in Figure 2.6.

The freely available software package CGAL [78] provides an implementation of discrete harmonic maps as well as various other mesh parameterization methods.

2.3 B-spline Approximation

This Section reviews the construction of B-spline surfaces to approximate discrete data. Equations that need to be solved are stated in a uniform notation aimed for straightforward implementation. It is assumed that the order k and a knot vector τ are given.

As input, a set of data points and their parameterization is required. Optionally, they can also have weights assigned to them, which is useful when considering a non-uniform distribution of data points. If no weights are given, they are assumed to be equal to one. For complex objects that cannot be modeled with a single surface, a partition into quadrilateral cells is required as well. The data contained in each cell is then approximated by an individual B-spline surface that has to satisfy certain continuity constraints with neighboring surfaces.

For simplicity, Subsection 2.3.1 discusses the construction of a single surface only. The concept is generalized and a global system is introduced in Subsection 2.3.2 which includes the constraints necessary for the computation of a continuous model consisting of multiple B-spline surfaces.

2.3.1 Single B-spline Surface Approximation

First, the approximation of a set of data points by a single B-spline surface is discussed. This requires a given set of n_p points $p_i \subset \mathbb{R}^3$, $i = 1, ..., n_p$ with assigned parameter values $(u_i, v_i) \in [0, 1]^2$ and weights $w_i \in \mathbb{R}$.

The shortest distance between a data point and a B-spline surface is the solution of the nonlinear minimization problem

$$\min_{(u,v)\in[0,1]^2} \|S(u, v) - p_i\|. \tag{2.6}$$

However, the given parameter values (u_i, v_i) express where the closest point on the B-spline surface is expected. Hence, the *approximately closest point* $S(u_i, v_i)$ is used to estimate the distance between the point and the surface, i.e.,

$$\min_{(u,v)\in[0,1]^2} \|S(u, v) - p_i\| \approx \|S(u_i, v_i) - p_i\|. \tag{2.7}$$

Using the approximate shortest distances, the least squares error of the surface with respect to the input data is denoted by

$$E_{\text{LS}} = \frac{1}{2} \sum_{i=1}^{n_p} w_i \left\|S\left(u_i, v_i\right) - p_i\right\|^2. \tag{2.8}$$

An equivalent notation can be used to simplify this term. Let $b \in \mathbb{R}^{n_c^2 \times 3}$ be a one dimensional list containing the B-spline control points and let $b_x, b_y, b_z \in \mathbb{R}^{n_c^2}$ be the vectors containing their x, y, and z coordinates. A correlation between the list index $\bar{i}$ and the indices i, j in the original grid is established through a bidirectional mapping, e.g., $\bar{i} = in_c + j$. Using the same index mapping, let

$$N(u,v) = [N_1(u)N_1(v), \dots, N_{n_c}(u)N_{n_c}(v)] \in \mathbb{R}^{1\times n_c^2} \tag{2.9}$$

be a row vector containing the associated tensor products of basis functions. Then, Equation (2.8) becomes

$$E_{\text{LS}} = \frac{1}{2}\sum_{i=1}^{n_p} w_i \left\| N(u_i, v_i)b - p_i^T \right\|^2 .$$

Furthermore, let $N \in \mathbb{R}^{n_p \times n_c^2}$ be the matrix containing the basis function coefficients all given the parameter values u_i, v_i and let a list of all data points be denoted by p, i.e.,

$$N = \begin{bmatrix} N(u_1, v_1) \\ \vdots \\ N(u_{n_p}, v_{n_p}) \end{bmatrix}, p = \begin{bmatrix} p_1^T \\ \vdots \\ p_{n_p}^T \end{bmatrix}.$$

The least squares error can then be expressed as

$$E_{\text{LS}} = \frac{1}{2}\sum_{\sigma\in\{x,y,z\}} (Nb_\sigma - p_\sigma)^T W (Nb_\sigma - p_\sigma)$$

with the weights as diagonal matrix $W = \text{diag}(w_1, ..., w_{n_p})$. Its gradient is

$$\nabla_\sigma E_{LS} = N^T W N b_\sigma - N^T W p_\sigma .$$

Minimizing the least squares error is equivalent to solving $\nabla_\sigma E_{\text{LS}} = 0$ for all components $\sigma \in \{x, y, z\}$.

Due to the discrete nature of the data, an approximation based on only the least squares error often includes unwanted wiggles. This is compensated by penalizing the deviation from a smooth surface by a fairing term. A commonly used term is the *thin-plate energy* functional [15, 17, 30]

$$E_{\text{TP}} = \frac{1}{2}\int_0^1\int_0^1 \|\partial_{uu}S(u,v)\|^2 + 2\|\partial_{uv}S(u,v)\|^2 + \|\partial_{vv}S(u,v)\|^2 \,\mathrm{d}u\mathrm{d}v \tag{2.10}$$

which, for B-splines, can be rewritten as

$$E_{\text{TP}} = \frac{1}{2}\sum_{\sigma\in\{x,y,z\}} b_\sigma^T M b_\sigma$$

with gradient

$$\nabla_\sigma E_{\text{TP}} = M b_\sigma$$

where $M \in \mathbb{R}^{n_c^2 \times n_c^2}$ is a matrix containing the integrals of the basis functions given by

$$\begin{aligned} M = \int_0^1 \int_0^1 & (\partial_{uu} N(u,v))^T (\partial_{uu} N(u,v)) \\ & + 2(\partial_{uv} N(u,v))^T (\partial_{uv} N(u,v)) \\ & + (\partial_{vv} N(u,v))^T (\partial_{vv} N(u,v)) \mathrm{d}u \mathrm{d}v. \end{aligned} \tag{2.11}$$

The B-spline basis functions are piecewise polynomials. Hence, their integrals can either be determined analytically or by numeric integration, e.g., applying Gaussian quadrature to each knot interval $[\tau_i, \tau_{i+1})$ with a high enough degree to be exact for the corresponding polynomial [2].

To compute a smooth surface that approximates the data points, a linear combination of both terms is minimized, i.e.,

$$\text{minimize } (1-\lambda) E_{\text{LS}} + \lambda E_{\text{TP}} \tag{2.12}$$

with the parameter $\lambda \in [0, 1)$ controlling the impact of the fairing term to the resulting surface. Since (2.12) is a system of quadratic equations with respect to the B-spline control points, its optimal solution is found by solving

$$\begin{aligned} 0 &= (1-\lambda)\nabla_\sigma E_{\text{LS}} + \lambda \nabla_\sigma E_{\text{TP}} \\ &= (1-\lambda)(N^T W N b_\sigma - N^T W p_\sigma) + \lambda M b_\sigma \end{aligned}$$

for each coordinate $\sigma \in \{x, y, z\}$. It can be written as

$$((1-\lambda)(N^T W N) + \lambda M) b = (1-\lambda) N^T W p. \tag{2.13}$$

This is a linear system with respect to the control points b. Efficient solvers are implemented in various freely available code packages, e.g., Eigen [29].

Iterative Approximation

The described approach aims on minimizing the distance between data points p_i and the approximately closest points on the surface $S(u_i, v_i)$, associated to them through their parameterization. However, these are not necessarily the smallest distances between the data points and the surface. It is a common practice to improve the quality of the approximation by employing an iterative approach, alternating between the construction of the surface and an update to the parameter values.

After a surface has been constructed, the parameter values of each data point are updated, by solving

$$(u_i', v_i') = \text{argmin}_{u,v \in [0,1]} \|S(u,v) - p_i\|.$$

Approximations to these new parameter values can be found efficiently by considering low degree Taylor series [66]. However, with modern hardware, a more accurate solution can also directly be acquired by using numerical nonlinear solvers, e.g., NLopt [39].

2.3.2 Constructing a Continuous B-spline Surface Model

When constructing a model with n_s surfaces, the same principles as for a single surface apply. However, some modifications are necessary to ensure continuous transitions between neighboring surfaces. Data and control points are stored in a global list by concatenating the lists of each individual surface. The coefficient, weight, and smoothing matrices are combined to diagonal block matrices. Using

$$\widehat{b} = \begin{bmatrix} b^{(1)} \\ \vdots \\ b^{(n_s)} \end{bmatrix}, \widehat{p} = \begin{bmatrix} p^{(1)} \\ \vdots \\ p^{(n_s)} \end{bmatrix},$$

$$\widehat{N} = \begin{bmatrix} N^{(1)} & & 0 \\ & \ddots & \\ 0 & & N^{(n_s)} \end{bmatrix}, \widehat{W} = \begin{bmatrix} W^{(1)} & & 0 \\ & \ddots & \\ 0 & & W^{(n_s)} \end{bmatrix}, \widehat{M} = \begin{bmatrix} M & & 0 \\ & \ddots & \\ 0 & & M \end{bmatrix},$$

instead of the single surface terms in Equation (2.13) leads to the same result as constructing each surface individually.

Achieving C^0-continuity by Adding Constraints

To obtain C^0-continuity, i.e., a watertight model, the row of control points along a shared boundary between any two neighboring surfaces needs to be identical. This can be expressed by a number of constraints of the form $\widehat{b}_i - \widehat{b}_j = 0$ with i, j being the indices in the global control point vector of the points that need to be identical. All constraints together can be expressed by a matrix vector product: $G\widehat{b} = 0$ where G is a matrix with each row expressing one such equality constraint.

With these constraints, the optimization problem (2.12) becomes

$$\begin{aligned} &\text{minimize } (1-\lambda)\widehat{E}_{\text{LS}} + \lambda \widehat{E}_{\text{TP}} \\ &\text{s.t. } G\widehat{b} = 0. \end{aligned} \tag{2.14}$$

Using the method of Lagrange multipliers, an optimal solution of (2.14) can be found by solving

$$\left[\begin{array}{c|c} ((1-\lambda)\widehat{N}^T\widehat{W}\widehat{N}+\lambda\widehat{M} & G^T \\ \hline G & 0 \end{array}\right]\left[\begin{array}{c} \widehat{b} \\ \hline \Lambda \end{array}\right] = \left[\begin{array}{c} (1-\lambda)\widehat{N}^T\widehat{W}\widehat{p} \\ \hline 0 \end{array}\right] \tag{2.15}$$

with a vector of Lagrange multipliers Λ. This system can be implemented and solved in a straightforward way. However, adding the C^0-constraints in the form of the matrix G increases the size of the overall system.

Achieving C^0-continuity by Reduction of Variables

An alternative is to implicitly enforce C^0-continuity, by using only one variable for each set of control points that are supposed to be identical. The matrix G is not needed and the overall number of unknowns is reduced, which allows better performance when solving the system. However, an index transformation is necessary to implement this system, i.e., each index $\widehat{i}$ in the global (full) control point list $\widehat{b}$ is assigned a new index $\tilde{i}$ in the reduced list of control points $\tilde{b}$. Using a matrix H with entries $H_{\widehat{i},\tilde{i}} = 1$ and zero everywhere else, the original list of control points can be reconstructed as $\widehat{b} = H\tilde{b}$. Inserting that in into the objective function (2.14) and solving for the gradient being equal to zero leads to

$$\left((1-\lambda)H^T\widehat{N}^T\widehat{W}\widehat{N}H + \lambda H^T\widehat{M}H\right)\tilde{b} = (1-\lambda)H^T\widehat{N}^T\widehat{W}\widehat{p} \tag{2.16}$$

Achieving G^1-continuity

The index transformation can be modified to also ensure G^1-continuity, i.e., transitions between surfaces with continuous tangent planes. Here, a method by Fan and Peters [20] is used. It is designed for B-splines of order $k = 3$ with grids of $n_c = 8$ control points per row and requires two double interior knots within the knot vector. It is therefore set to

$$\tau = (0,0,0,0,\frac{1}{3},\frac{1}{3},\frac{2}{3},\frac{2}{3},1,1,1,1).$$

Given a quad-mesh, they provide equations for a smooth surface model where all B-spline control points $\widehat{b}_i$ are expressed as linear combinations of the quad-mesh vertices q. Using these equations and an appropriate indexing, the overall B-spline control points can be expressed as matrix-vector product $\widehat{b} = Vq$. The coefficients of the matrix V are explicitly stated in [20].

By considering the vertex positions q as unknowns and using V instead of the index transformation H, (2.16) can be rewritten as

$$\left((1-\lambda)V^T\widehat{N}^T\widehat{W}\widehat{N}V + \lambda V^T\widehat{M}V\right)q = (1-\lambda)V^T\widehat{N}^T\widehat{W}\widehat{p}. \tag{2.17}$$

Solving this linear system corresponds to finding optimal vertices q of a quad-mesh. Applying the scheme by Fan and Peters to these vertices yields a set of B-spline surfaces that minimize the objective function (2.14). The resulting surfaces are G^1-continuous everywhere.

Chapter 3

Fast and Memory-efficient Quantile Filter for Data in Three and Higher Dimensions

3.1 Abstract

Quantile filters are a very common tool in images processing. One of the most prominent instances is the median filter, which is primarily used for image denoising. Fast implementations accumulate a histogram within a mask around each pixel. The desired quantile for a pixel is then selected from the corresponding histogram in constant time.

When considering a d-dimensional image and a hypercube with side length $2r + 1$ as filter mask, constructing the histogram by simply reading all pixel values within the mask takes $\mathcal{O}(r^d)$ operations per pixel. This results in high computational costs for large filter masks.

For two-dimensional images, Huang et al. propose a "sliding window" approach in which a histogram is shifted pixel-by-pixel. As shown in this chapter, this approach can easily be adapted to be applied in arbitrary dimensions with $\mathcal{O}(r^{d-1})$ operations per pixel. Generalizing this method, Perreault and Hébert propose an algorithm for images of any dimension with $\mathcal{O}(1)$ operations per pixel. However, the low runtime complexity comes at the cost of high memory requirements.

This chapter presents a novel hybrid approach. It is based on a generalization of the two aforementioned methods and allows for a trade-off between computational cost and memory requirements. The proposed algorithm is analyzed with respect to runtime and space complexity.

The primary contributions of this chapter are:

- Review and complexity analysis of two existing algorithms (Huang et al., Perreault and Hébert)

- A novel hybrid approach
- A complexity analysis of memory requirements and computational cost of the presented hybrid approach

3.2 Introduction

The median filter is a popular tool in image processing. It is primarily used for image denoising or during pre-processing steps of complex algorithms. While it is mostly known for its use in two-dimensional image processing, it can be applied to data of arbitrary dimensions. The focus of this thesis are 3D images created via computed tomography.

In the context of image processing, a median filter is a special case ($p = 0.5$) of the p-quantile filter. This filter produces an image in which each pixel is assigned the p-quantile of the pixels in a predefined neighborhood within the input image. Quantile filters are used as a generalization of gray value mathematical morphology [74]. However, straightforward implementations are often slow and become impractical for large filter masks. Finding efficient solutions is highly desirable. The key feature of interest is the number of operations per pixel with respect to the size of the filter mask. Note that in this chapter, the term "pixel" refers to a coordinate-value-pair in an image of arbitrary dimension.

Several contributions in this field exist. The following key publications represent major improvements with respect to the computational complexity.

Huang et al. [35] propose a "sliding window" approach in which a histogram is shifted pixel-by-pixel. The method is designed for 2D images on which it achieves linear runtime per pixel. It can easily be extended to arbitrary dimensions as shown in Section 3.3.2.

Weiss [80] introduces an algorithm for 2D data based on computing partial histograms only. These histograms are then reused for multiple computations which results in logarithmic runtime per pixel.

Perreault and Hébert [58] present an extension of [35]. Their algorithm only needs a constant number of operations per pixel for images of any dimension. However, the low runtime complexity comes at the cost of high memory requirements.

In this chapter, a novel hybrid algorithm is presented based on a generalization of [35] and [58]. It allows for a trade-off between computational cost and memory requirements. This becomes clear when analyzing the runtime and memory requirements of the methods by Huang et al. and Perreault and Hébert, see Subsections 3.3.2 and 3.3.3. The hybrid approach is presented and analyzed in Subsection 3.3.4. Finally, Section 3.4 compares numbers for time and memory usage of the different algorithms. A key result is that, with the implementations used for this thesis, the hybrid strategy is more memory efficient and even faster than Perreault and Hébert's method for most practically relevant filter and image sizes.

3.3 Method

In order to keep used symbols and variables intuitive, the notation in this chapter deviates from the general notation throughout the thesis. It instead aims for consistency with defaults in the field of image processing.

Let $f : \Omega \to \{0, 1, \ldots, 2^b - 1\}$ denote a b-bit integer-valued d-dimensional image on a regular, cubical grid $\Omega \subset \mathbb{Z}^d$. The maximum size of the image in any of its dimensions is denoted by n. Furthermore, let M be a d-dimensional filter mask centered in the spatial coordinate $x \in \Omega$. The filter masks used here are d-dimensional hypercubes with side length $2r + 1$, where r is called radius.

Note that the primary focus of this work is data of dimension $d = 3$. Computed tomography typically produces gray value images with $b = 8$ or $b = 16$. Common values for n are 1024 or 2048. State-of-the-art hardware can produce even larger images.

Here, the algorithmic complexity and memory requirements are analyzed with respect to the mask radius r, data dimension d, and data size n. All considered algorithms require an initialization step that has a higher algorithmic complexity than the remaining algorithm. In practice, the resulting impact on the total runtime is neglectable. Therefore, the initialization is omitted in the complexity analysis.

3.3.1 Quantile Computation

A basic operation is the computation of the p-quantile Q_p, with $p \in [0, 1]$. This operation is done the same way by all algorithms analyzed here. It requires a discrete histogram $H(M)$ of the pixel values inside the filter mask M. This histogram consists of 2^b integer-valued entries. The p-quantile can be found by traversing the histogram, starting at the first entry, and summing up the individual values until the sum reaches the required percentage p of the total number of pixel within the filter mask. The index of the entry at which this procedure stops is the desired p-quantile. This approach requires at most $2^b - 1$ additions and comparisons. Hence, in terms of the variables considered for the complexity analysis, computing Q_p from $H(M)$ requires $\mathcal{O}(1)$ time.

Even though this step requires constant time, it has a strong impact on the total runtime. Perreault and Hébert proposed two optimizations: (1) "Multi-level histograms" storing $H(M)$ at multiple resolutions. This allows for the computation of Q_p using a lower number of operations on average. It was previously proposed by Alparone et al. [1]. (2) "Conditional kernel updating" moves the mask M but updates the highest histogram resolution only for entries needed for the computation of Q_p. A detailed description is given in [58].

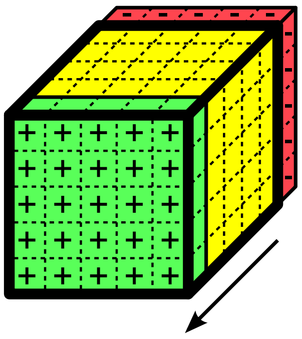


Figure 3.1: *Sliding window in 3D. The histogram in a cubical mask is shifted by one pixel by subtracting the gray values in one plane in negative scanning direction from and adding the gray values in one plane to the histogram.*

3.3.2 Algorithm by Huang et al.

The algorithm by Huang et al. is based on a "sliding window" approach. The filter mask M is moved through the image f in a given scan direction. For two filter masks M_1 and M_2 around neighboring pixel, the respective histograms $H(M_1)$ and $H(M_2)$ differ in two $(d-1)$-dimensional facets, see Figure 3.1. This allows for the computation of $H(M_2)$ from $H(M_1)$ by updating only the histogram entries corresponding to the pixels within these two facets, i.e., for each pixel added to (removed from) the mask, the histogram entry corresponding to its value is increased (decreased) by one.

In [35], the authors present the algorithm for $d = 2$ and state its time complexity as $\mathcal{O}(n^2 r)$. The following lemmas generalize this result for the d-dimensional case.

Lemma 1. *The algorithm by Huang et al. has a time complexity of $\mathcal{O}(n^d r^{d-1})$ for filtering a d-dimensional image f*

Proof. The filter mask M is shifted to the next pixel by only considering the differences in the next and last $(d-1)$-dimensional facet in scan direction, see Figure 3.1. This results in $(2r+1)^{d-1}$ additions to and subtractions from the histogram $H(M)$. As shown in Subsection 3.3.1, the computation of Q_p is $\mathcal{O}(1)$. Hence, the algorithm performs $\mathcal{O}(r^{d-1})$ operations per pixel, i.e., $\mathcal{O}(n^d r^{d-1})$ for the entire image. After reaching the end of a scan line, a new histogram has to be constructed to start processing the next line. This step causes $\mathcal{O}(r^d)$ operations, but can be neglected for large values of n. □

Lemma 2. *The algorithm by Huang et al. has a space complexity of $\mathcal{O}(1)$ for filtering a d-dimensional image f*

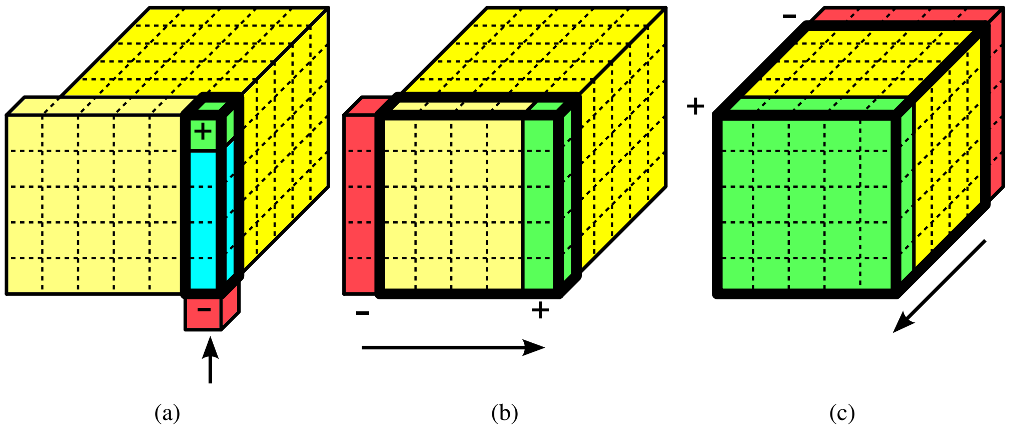


Figure 3.2: *Recursive extension of the "sliding window" approach. In addition to the histogram of the cubical filter mask, the algorithm of Perreault and Hébert also stores histograms for 1D columns which are moved by only considering differences in two pixels (a). The histograms of these columns are used to move 2D slices (b), which are then used to move the actual filter mask (c).*

Proof. During execution, the method holds exactly one histogram $H(M)$ in memory, which consists of 2^b integer-valued entries. Adding to and subtracting from the histogram does not require any further memory. $\square$

3.3.3 Algorithm by Perreault and Hébert

Perreault and Hébert extend the algorithm of Huang et al. by propagating the concept of the sliding window recursively down to the lowest dimension $d = 1$. For $d = 3$, this is illustrated in Figure 3.2. As explained in Subsection 3.3.2, the cubical filter mask can be moved by one pixel by only considering the pixel values in two 2D slices. However, when the neighboring line is processed, the filter mask is moved using almost the same slices. In fact, each slice differs from the slice of the last line by only two 1D columns. Hence, a histogram is stored for each slice along a line. The filter mask is then moved by adding the histogram of the next slice in scan direction and subtracting the histogram of the last slice. After a line is processed, all slices are moved to the next line by only considering the differences in two columns per slice. The same principle applies to the columns as well, i.e., histograms are stored for each column within a plane. When the algorithm moves to the next plane of the image, each column is moved by adding the next pixel above it and subtracting the last pixel on the bottom.

Note that the addition of histograms requires at most 2^b operations and is therefore $\mathcal{O}(1)$. Moving the column histograms requires only two operations and is also

n	Huang et al.	Perreault and Hébert	Hybrid Approach
500	2 kB	0.5 GB	1 MB
2 000	2 kB	8 GB	4 MB
4 000	2 kB	31 GB	8 MB

Table 3.1: *Memory requirements for quantile filtering of an $n \times n \times n$ image with $r = 3$, $b = 8$, and using 64-bit integers for all histogram entries. The method by Huang et al. has neglectable memory requirements while the algorithm by Perreault and Hébert quickly becomes impractical. The proposed hybrid approach is a viable solution, even for large images.*

$\mathcal{O}(1)$. This means that only a fixed number of operations of time complexity $\mathcal{O}(1)$ are performed per pixel.

Lemma 3. *[58] Perreault and Hébert's algorithm has a time complexity of $\mathcal{O}(n^d)$ for filtering a d-dimensional image f, independent of r.*

For $d = 3$, this algorithm requires that n histograms of slices along the scanning line and n^2 columns within the entire scanning plane need to be held in memory during the entire process. The extension of the algorithm to higher dimensions is straightforward and does not increase the computational complexity, as stated in [58]. However, the memory requirements grow with a power of $d - 1$.

Lemma 4. *[58] Perreault and Hébert's algorithm has a space complexity of $\mathcal{O}(n^{d-1})$ while filtering a d-dimensional image f, independent of r.*

The proofs of Lemmas 3 and 4 follow from the proofs of the corresponding propositions 5 and 6 for the novel hybrid algorithm, see Subsection 3.3.4.

3.3.4 Hybrid Algorithm

For 2D images, the memory requirements of $\mathcal{O}(n)$ that arise when applying Perreault and Hébert's algorithm can easily be met by modern hardware. However, for high-resolution 3D images, the algorithm requires up to several Gigabyte for typical CT-image sizes, see Table 3.1. For higher dimensions, the amount of required memory becomes even larger, making the algorithm unfeasible for most image sizes. The method by Huang et al. is also not a viable option for high-dimensional data since its runtime grows exponentially with d, which is stated in Lemma 1.

A solution to the memory problem was proposed in [58]. By only processing chunks of the images, the memory requirements can be reduced. However, this introduces computational overhead and also becomes impractical for larger images.

This chapter presents a novel hybrid algorithm that addresses these issue by offering a compromise between runtime and memory requirements for dimensions $d > 2$.

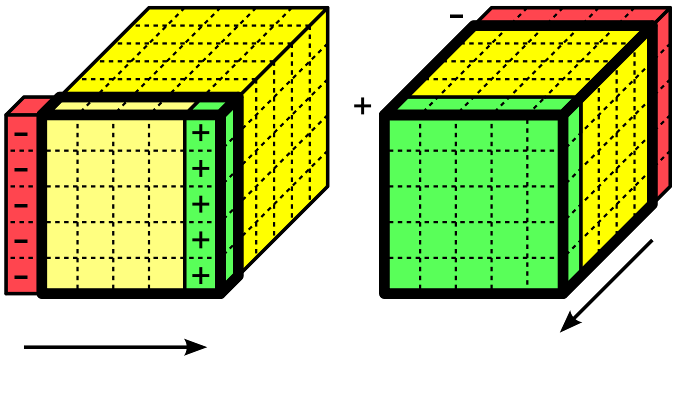


Figure 3.3: *Illustration of the novel hybrid algorithm in 3D with $c = 1$. The histograms of 2D slices are moved by considering the individual pixels in the last and next column along their path (a). The filter mask is then moved by simply adding and subtracting the histograms of the last and next slice in scan direction (b).*

Like the method by Perreault and Hébert, the algorithm moves the filter mask through the image by adding and subtracting $(d-1)$-dimensional facets. The histograms of these facets are also stored and moved orthogonal to the scan direction by adding and subtracting histograms of $(d-2)$-dimensional facets. This recursion is continued as described by Perreault and Hébert, but stops in dimension $d-c$, with $0 \leq c < d$. The histograms of the facets in this direction are moved using the method by Huang et al. which requires $\mathcal{O}(r^{d-1-c})$ operations, see Figure 3.3. This is computationally more expensive then the full recursion, but reduces the memory consumption exponentially. The parameter c of this method interpolates between the algorithms by Huang et al. $(c = 0)$ and Perreault and Hébert $(c = d-1)$.

Proposition 5. *The hybrid algorithm with recursion-depth parameter $0 \leq c < d$ has a time complexity of $\mathcal{O}(n^d r^{d-1-c})$ for filtering a d-dimensional image f.*

Proof. The computation of the p-quantile is done once per pixel. As shown in Subsection 3.3.1, this operation is $\mathcal{O}(1)$. In addition, the filter mask and exactly one facet of each dimension between $d-1$ and $d-c$ are moved by one pixel. The addition of two histograms requires 2^b additions, which is of complexity $\mathcal{O}(1)$ in terms of the variables considered here. Therefore, moving the filter mask through the image and performing updates to the corresponding histogram by adding and subtracting histograms of $(d-1)$-dimensional facets is of complexity $\mathcal{O}(1)$. This argumentation can recursively be applied to moving the lower-dimensional facets as well. The recursion stops at dimension $d-c$. At this point, the "sliding window" approach by Huang et al. is used to move the $(d-c)$-dimensional facet by one pixel. According to Lemma 1, this causes $\mathcal{O}(r^{d-1-c})$ operations. This leads to a total time complexity of $\mathcal{O}(n^d r^{d-1-c})$ for the full image. □

Proposition 6. *The hybrid algorithm with recursion-depth parameter $0 \leq c < d$ has a space complexity of $\mathcal{O}(n^c)$ for filtering a d-dimensional image f.*

Proof. Maintaining the histogram of the filter mask requires $\mathcal{O}(1)$ memory. To move the mask via adding and subtracting $(d-1)$-dimensional facets, one facet-histogram needs to be held in memory for each pixel along the entire scan line. This requires $\mathcal{O}(n)$ memory. Following the recursion of the algorithm, a set of $(d-2)$-dimensional facet-histograms needs to be held in memory along the entire path of each $(d-1)$-dimensional facet, causing a total memory requirement of $\mathcal{O}(n^2)$. The recursion stops at depth c at which the method by Huang et al. is used to move the $(d-c)$-dimensional facets. According to Lemma 2, this requires $\mathcal{O}(1)$ memory. Hence, the total memory requirements of the algorithm are

$$\sum_{i=0}^{c} \mathcal{O}(n^i) = \mathcal{O}(n^c).$$

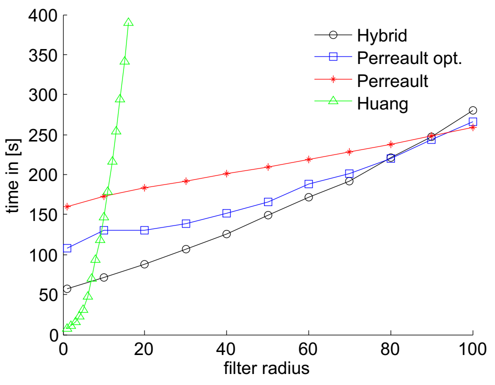
□

Edge Treatment

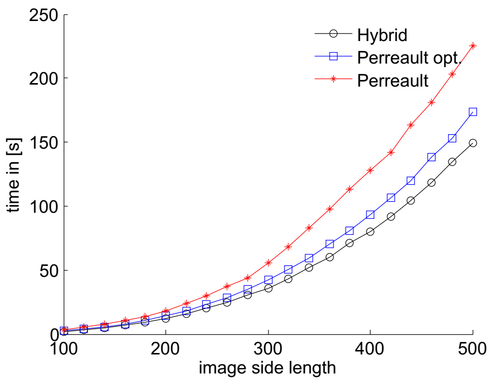
As stated in [58], storing and updating facet-histograms at image boundaries adds a non-constant term to the algorithmic complexity. This term is neglectable in terms of the asymptotic analysis, as it considers the case $n \to \infty$. In practice, however, it does have an effect on the runtime of the algorithm. The implementation, used within the context of this thesis, stores and maintains additional r histograms beyond the domain Ω in each direction. Values outside of the image are defined using reflective edge treatment. Therefore, the true space complexity of this specific implementation is $\mathcal{O}((n+2r)^c)$. However, for practically relevant cases with $r \ll n$, this reduces back to $\mathcal{O}(n^c)$.

3.4 Results

All methods mentioned in Section 3.3 were implemented in `C++` and runtimes were measured. The experiments were performed under 64-bit Linux on one core of an Intel Xeon X5675 CPU at 3 GHz. All implementations were single-threaded. However, parallelization techniques discussed in [35] and [58] are applicable to all analyzed methods. Tests were performed on 8-bit 3D image data. For the application of the hybrid algorithm, the parameter $c = 1$ was chosen. For more accurate measurements



(a) *Runtimes on a* $500 \times 500 \times 500$ *image.*



(b) *Runtimes for* $r = 50$.

Figure 3.4: *Runtimes in 3D, depending on* r *and* n.

of the algorithm's performance, chunks with side length $n + 2r$ were cut out from larger images. Only the inner n^3 pixels were processed to avoid additional overhead from edge effects. The presented measurements are the minimum runtime among three applications for each parameter setting. The optimizations, described in Subsection 3.3.1, for the algorithm by Perreault and Hébert were implemented as well, but measured separately (denoted by "Perreault opt." in Figure 3.4), as they do not affect the computational complexity itself.

Note that using large quantities of memory causes additional run time, e.g., because of cache misses. This means that even though the computational complexity improves for $c > 0$, the overhead caused by initialization and managing increasing memory leads to an increase in the measured runtime. Such an effect can be observed in Figure 3.4a. The method by Huang et al. is fastest for small values of r. Perreault and Hébert's algorithm gains a significant speed-up when using the additional optimizations, but computational time increases with r, even though its theoretical time complexity is $\mathcal{O}(1)$. The hybrid algorithm outperforms both methods for filter sizes $r < 80$. For a fixed value of $r = 50$, Figure 3.4b shows how the algorithms scale when the image size increases. The method by Huang et al. is left out, as it is no longer viable for the demonstrated filter size.

3.4.1 Conclusions

This chapter presented an algorithm for quantile filtering of data in three or more dimensions. It is a generalization of the methods by Huang et al. and Perreault and Hébert. For 3D images, it allows for a hybrid strategy that offers a trade-off between computational cost and memory requirements. In higher-dimensional images, the amount of possible hybrid strategies increases by the number of additional dimensions. This allows the filter to be tuned for optimal performance on the available hardware.

The presented algorithm performed faster than an implementation of [58] for filter masks with $r < 80$. For those sizes, the constant time complexity of [58] did not compensate for the practical overhead caused by higher memory consumption.

3.5 Core Reference

- D. Mosbach, H. Hagen, M. Godehardt, and O. Wirjadi: Fast and memory-efficient quantile filter for data in three and higher dimensions. *IEEE International Conference on Image Processing (ICIP)*, pp. 2928-2932, (2014).

Chapter 4

Topologically Robust B-spline Reconstruction of Fibers from 3D Images

4.1 Abstract

The micro-structure of wood-based insulation materials is analyzed to gain insight into how features on microscopic scales influence macroscopic thermal conductivity. Three-dimensional image data obtained by micro-computed tomography reveals a complex structure formed by cellulose fibers. To study the effect of geometry changes, simple B-spline representations of these fibers are highly desirable.

A straightforward solution is to extract a triangulated isosurface from the 3D image and partition it into quadrilateral macro-cells with disk-like topology. For each cell, a B-spline surface is constructed by minimizing a least squares error term. However, the physical processing of the material affects the structure of the fibers. The resulting changes in surface topology cause difficulties for the quadrilateral partitioning. Image processing tools can solve these topological issues, but they also impact geometry.

This chapter presents a novel approach that splits geometry and topology processing of the data. It allows for topological simplification while still preserving the geometry of a scanned object. Established B-spline approximation methods are used to create a model. Presented results show that smooth and accurate models can be created for a variety of challenging data sets.

This chapter presents the following contributions:

- A novel approach for data processing that allows for stable B-spline approximations of objects given as 3D images.
- A specific pipeline to construct B-spline surface models of (damaged) cellulose

fibers.

- Verification of the method by constructing B-spline models consisting of a low number of surfaces to represent several fibers given as high-resolution data sets.

4.2 Introduction

An important part in researching material properties is the generation of models representing microscopic structures which can be used for visualization and simulation purposes. How these models are obtained depends strongly on the size and complexity of the samples, the initial data acquisition, and the target application.

This chapter addresses the model generation of cellulose fibers. A cellulose fiber can be seen as a hollow tube which is open on both ends. The empty space inside is called lumen. Meaningful models of such structures are helpful to better understand and assess mechanical properties of a material on a macroscopic scale [3]. For example, this insight allows researchers to establish a relation between the micro-structure and physical properties of paper [47]. Another example is an analysis of sound-dampening properties of wood-based acoustic insulation material with respect to its tortuosity - a parameter that helps linking physical, acoustic, and morphological properties of a material [61].

The motivating research of this chapter is the investigation of heat conductivity properties of wood-based thermal insulation materials. The micro-structure of these materials consists of a complex system of cellulose fibers which occur in chunks, chips, and as individuals. To conduct simulations, B-spline surface models of such fibers are desired. These models should consist of a low number of continuously connected surfaces and include the inner and outer surfaces of the fiber wall. The main feature of interest is the shape of the fibers as well as the contained lumen. This means that small holes in the wall as well as roughness along the surface can be neglected and do not need to be present in the final model.

The initial data is a scan of a material sample given as 16-bit gray value image which is obtained by high-resolution synchrotron computed tomography. For denoising, a $5 \times 5 \times 5$ median filter is applied after which the image is binarized with a global gray value threshold. Individual fibers are then extracted from this image based on a local shape criterion [13].

However, the physical processing of the material causes damage to the micro-structure which leads to cuts and holes in the fiber walls as well as increased surface roughness in some areas. As Figure 4.1 shows, these effects complicate the topological structure of the surface by adding numerous small tunnels, holes, and cavities.

As described in Chapter 2, a straightforward solution to create a B-spline model of those fibers is to extract the isosurface of such a fiber in the form of a triangulated mesh. In order to compute a continuous B-spline model, this mesh needs to be partitioned

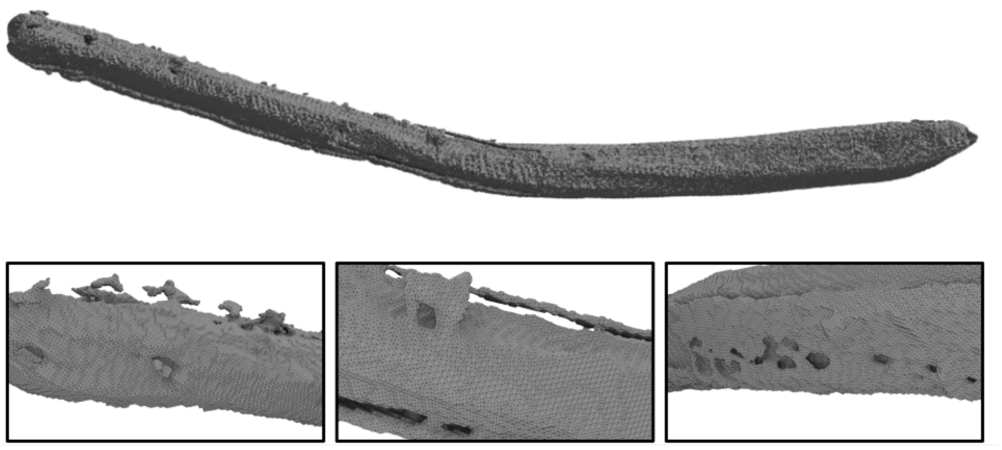


Figure 4.1: *Example of an extracted fiber. While the overall geometric impression appears to be smooth, close-ups show artifacts affecting the local topology on the surface.*

into a collection of quadrilateral macro-cells. Each macro-cell is approximated by a B-spline surface. Due to the non-trivial topological situation on the fiber's surface, most approaches to compute such a quadrilateral decomposition either fail or produce a very large amount of small macro-cells. Standard image denoising operations can solve the topological problems, but might also introduce a high geometric error to the resulting surface.

The contribution of this chapter is a new approach that splits the data processing pipeline into geometry and topology processing. A strongly cleaned and repaired image is used to create the mesh of larger quadrilateral faces capturing the topology of the idealized cellulose fiber. This quad-mesh is then used to partition the original data, extracted from the unprocessed image, by assigning each data point to the nearest quad-face. Processing the data this way provides the necessary input for established B-spline approximation methods. The resulting B-spline models have fiber-like topology and closely approximate the geometry of the original data.

4.3 Related Work

While no methods exist to extract isosurfaces directly in the form of B-splines, a variety of research has been published concerning the construction of B-spline approximations of discrete data given as point clouds or surface meshes. Such an explicit representation can easily be obtained from a 3D image by contouring methods like Marching Cubes.

Regular tensor product B-spline surfaces can only represent structures with disk-like topology. To construct a model of complex objects, the data needs to be decomposed into quadrilateral cells. The data points in each cell are approximated by a single

B-spline surface. The key challenge is to ensure certain continuity constraints along boundaries. While a watertight model, i.e., C^0-continuous transitions along shared boundaries, is almost always a requirement, it is often desired to also have at least tangent plane continuity, i.e., G^1-continuous transitions.

Eck and Hoppe [17] provide an in-depth description of the overall pipeline starting with a point cloud. Their first step is to establish a topological structure of the data by generating a fine surface mesh. Using a Voronoi-like tessellation and its dual Delaunay-like complex, the mesh is subdivided into a set of quadrilateral cells where each cell has a disk-like topology. Those cells are parameterized using harmonic maps. A B-spline approximation using a modification of Peters' scheme [59] is then computed to minimize the least squares error between the surfaces and the data points.

Gregorski et al. [27] propose a method to construct a set of B-spline surfaces approximating a given point cloud. They decompose their data into quadrangular cells based on a so-called strip tree which provides an adaptive subdivision of the data, similar to a quad-tree, into small boxes such that the points contained in each box can be approximated by B-spline surfaces with low error. Control points of neighboring surfaces are then adjusted during a post-processing step to ensure C^1-continuous transitions. Since they do not establish a topological structure of the surface, their approach works best on mostly flat objects.

Given a triangulated surface mesh and a manually defined subdivision into quadrilateral cells, Krishnamurthy and Levoy [42] remesh each cell to obtain a regular sampling of the data. This implicitly defines a parameterization and reduces the complexity of the B-spline approximation. However, to apply it to complex objects, an automatization of the quadrilateral decomposition is necessary.

Yoo [83] describes an approach to construct a B-spline model of human bones given as point cloud or sequence of CT images. First, the input data is used to define an implicit surface on which a fine quad-mesh is constructed. Each quad and the normal vectors along its boundaries are then interpolated by a B-spline surface.

Yoshihara et al. [84] capture the topology of a given point cloud by constructing an implicit function and applying a level set method. At the cost of geometric accuracy in noisy regions, this allows for a stable processing of difficult data. The object's surface is approximated by a Catmull-Clark subdivision surface and the corresponding control points are used to form a fine quad-mesh which is interpolated by B-spline surfaces.

Lin et al. [45] present a method to create a smooth B-spline model to approximate a mesh. The quadrilateral decomposition is done manually. After constructing a curve network representing the boundaries, the data points in each cell are approximated by a bi-quintic Bézier surface that interpolates the boundary curves. The resulting surface model is made G^1-continuous by also interpolating pre-computed normal vectors along the boundaries.

Zhao et al. [85] introduce an iterative approach, allowing individual surfaces to use differing knot vectors. Given a point cloud and a partition into quads, their approach

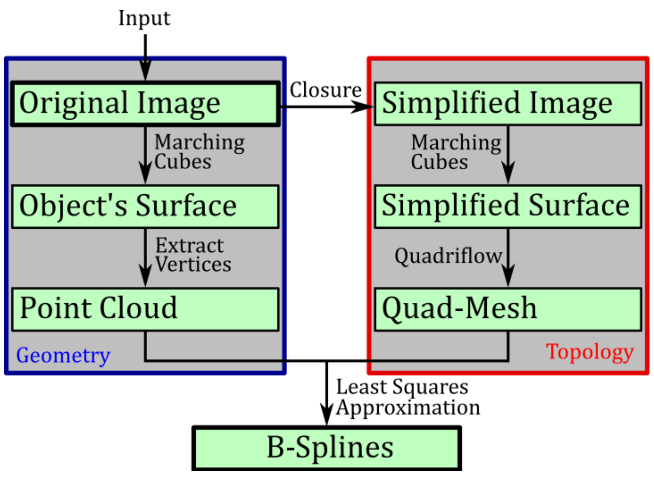


Figure 4.2: *Pipeline separating geometry and topology processing. To partition the data into quadrilateral cells, a morphological closure operation is applied to the input image. The isosurface of the image is extracted and re-meshed into a quad-mesh defining the topological structure of the B-spline model. The approximation is computed with respect to the data points extracted from the isosurface of the original mesh.*

constructs an initial set of B-spline surfaces to closely approximate the data and then ensures approximate G^1-continuity in a numerical post-processing step.

Fan and Peters [20] provide a construction scheme for bi-cubic B-splines with exactly two double inner knots. Equations for all control points are explicitly given as linear combinations of the surrounding vertices of the quad-mesh, but the method can also be modified to approximate a set of discrete data points.

While a large variety of methods for many scenarios exists, they are primarily designed for structures that allow for a decomposition into a relatively small number of quadrilateral macro-cells. However, this chapter deals with rough fiber structures with complicated surface topology. A straightforward decomposition of the data into disk-like quadrilateral cells would require a very high number of small cells. Automatically removing these effects during a pre-processing step strongly affects the geometry of the resulting B-splines. Hence, an alternative solution is introduced here.

4.4 Method

The key aspect of the proposed method is splitting geometry and topology processing, see Figure 4.2. After processing the data this way, standard methods for B-spline approximation can be applied, e.g., as described in Chapter 2.3.

To compensate for small holes and the rough surface structure in the data, a topolog-

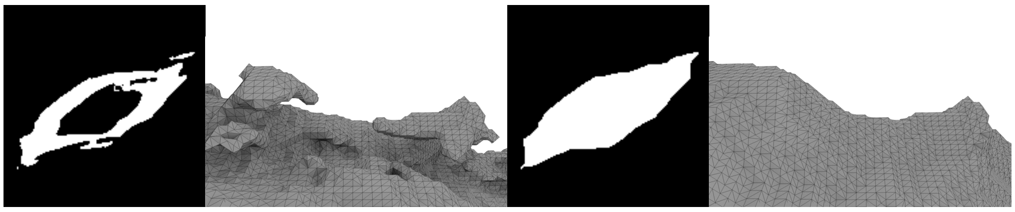


Figure 4.3: *A morphological closure operation simplifies the topology of the object. A slice of the 3D volume data (a) and a close-up of the corresponding region in the isosurface (b) highlight regions hard to handle for subsequent processing. Applying the closure operation simplifies the surface structure, as seen in (c) and (d).*

ical simplification is employed. This is done through the application of a morphological closure to the volume image, i.e., performing a sequential dilation and erosion on the object represented by the image's foreground [57]. The size of the filter mask needs to be chosen sufficiently large to fill the entire structure. Efficient implementations for large filter masks are available [52]. This operation preserves the original outer boundary in smooth regions and absorbs any outside material into the fiber's surface, see Figure 4.3. The inside of the fiber and holes in its wall are closed. The resulting object has the topology of a cylinder. Hence, applying Marching Cubes leads to a surface mesh with a topological structure that is well-suited to be partitioned into large and evenly sized macro-cells. A re-meshing into a small number of quadrilateral cells is performed on this mesh using a freely available implementation of the Quadriflow algorithm [34]. The resulting quad-mesh is a "rough geometric approximation" of the data, but it provides a suitable topological structure for a B-spline model.

For the geometry-focused part of the pipeline, the original image is considered. As with the closed image, a triangulated isosurface is extracted using Marching Cubes. The vertices of this triangulation provide the geometry data for the B-spline approximation. To associate them with the topological structure of the quad-mesh, each point is assigned to the closest quad-face. An initial parameterization is obtained by projecting the points onto their associated faces. By mapping the corners of a quadrilateral to the corners of the unit square, a correspondence is established between the position of the projected point in the quad-cell and a tuple in (u, v)-parameter space, where $(u, v) \in [0, 1]^2$.

Cellulose fibers are hollow. To reconstruct a fiber wall and the contained lumen, the inside and outside of the isosurface representing the wall are considered separately. The topology is the same in both cases. Hence, the same quad-mesh can be used for both approximations. However, each data point needs to be classified as belonging to either the interior or exterior surface of the fiber wall. The points can be classified by applying a Euclidean Distance Transform to the closed image, assigning to each voxel the distance to the closest point of the isosurface [25]. As the morphological closure still mostly preserves the outer surface, data points with low distances (here, ≤ 2 voxels) are considered to belong to the outside surface of the wall, while data points with larger distance values are considered to belong to the inside. Once the classification is performed, separate surface models are constructed, one for each set of data points.

The quad-mesh obtained from the simplified image is closed. Since the layout produced by the Quadriflow algorithm is sensitive to sharp features, the actual open ends of the fibers align well with individual quad-faces. To reproduce the open structure of fibers, these quads are removed and the open space between inside and outside B-spline surfaces is closed via linear interpolation between the boundary curves.

Image processing operations, including closure, Euclidean Distance Transform and isosurface extraction with Marching Cubes, are performed with MAVI [24]. The projection of the data points onto the quad-mesh is performed with the freely available software package libigl [36].

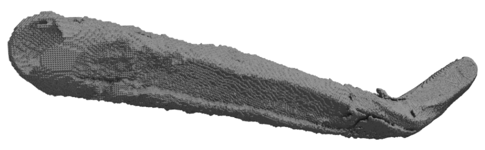
4.5 Results

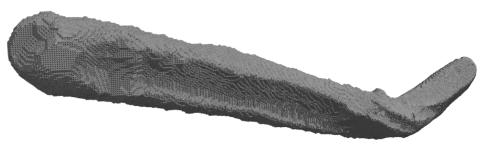
The presented method was applied to a number of isolated fibers. Figure 4.4 shows an example, including intermediate steps. Further results are shown in Figure 4.5. Even though many complicated topological situations arise in the data, the algorithm produces stable results, see Figure 4.6. The overall shape and important characteristics, like length and bending, are well-preserved. Loose noise-like structures occurring in regions with high roughness are smoothed, while larger material parts are absorbed resulting in only a small distortion of the surface. Holes in the fiber walls are closed, and the topological structure of the surface does not impact the stability of the quadrilateral partitioning.

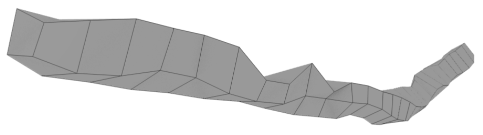
Each resulting B-spline surface is of order $k = 3$ and is defined by 8×8 control points. The surface model is constructed with the G^1-continuous approach by Fan and Peters, as described in Subsection 2.3.2, which defines the knot vector as $\tau = (0, 0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1, 1, 1, 1)$. The computation is done via three iterations of surface approximation and parameter optimization, with decreasing smoothness parameters $\lambda_1 = 0.9, \lambda_2 = 0.5, \lambda_3 = 0.1$.

4.5.1 Conclusions

This chapter presented a new approach that reconstructs high-resolution cellulose fibers from 3D images with a low number of B-spline surfaces. Due to the production process of the material, the micro-structure contains regions with complex topology that are


(a) *Original surface (*553 748 *triangles)*


(b) *Simplified surface*


(c) *Quad-mesh (*118 *quads)*

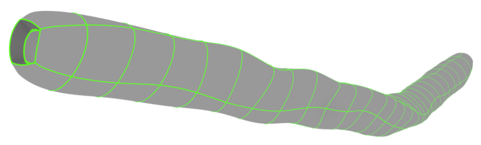
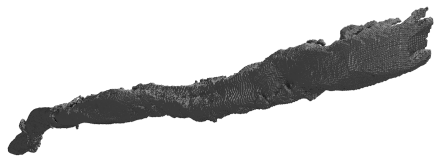
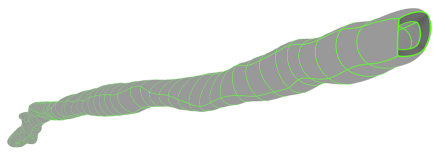
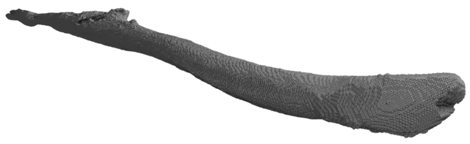
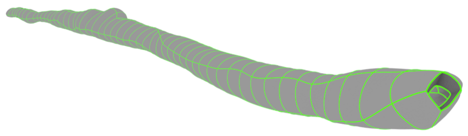

(d) *B-spline model (*236 *B-spline surfaces)*

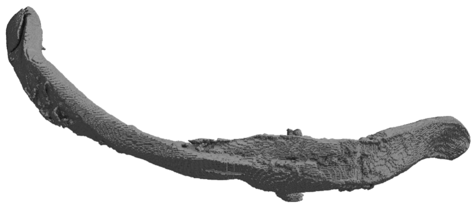
Figure 4.4: *Main processing steps. (a) shows an isosurface of the original image. After applying a morphological closure operation, the topology is simplified (b). A quad-mesh is computed for this surface (c). Using the geometry of the initial data and the topology of the quad-mesh, a B-spline model is constructed (d).*

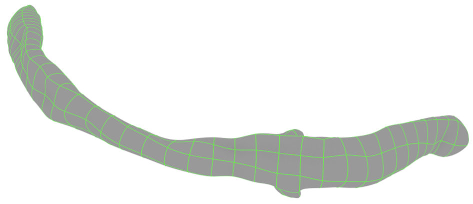

(a) *Isosurface (*1 203 844 *triangles)*

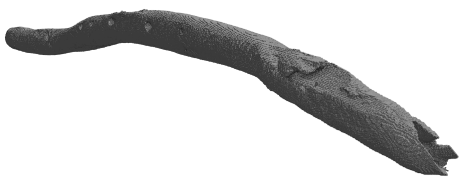

(b) *B-spline model (*676 *B-spline surfaces)*


(c) *Isosurface (*423 628 *triangles)*


(d) *B-spline model (*432 *B-spline surfaces)*


(e) *Isosurface (*710 808 *triangles)*


(f) *B-spline model (*580 *B-spline surfaces)*


(g) *Isosurface (*601 040 *triangles)*

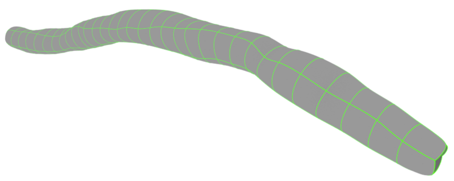

(h) *B-spline model (*268 *B-spline surfaces)*

Figure 4.5: *Additional results. Isosurfaces of the exact image data (a), (c), (e), (g) and their B-spline approximations (b), (d), (f), (h).*

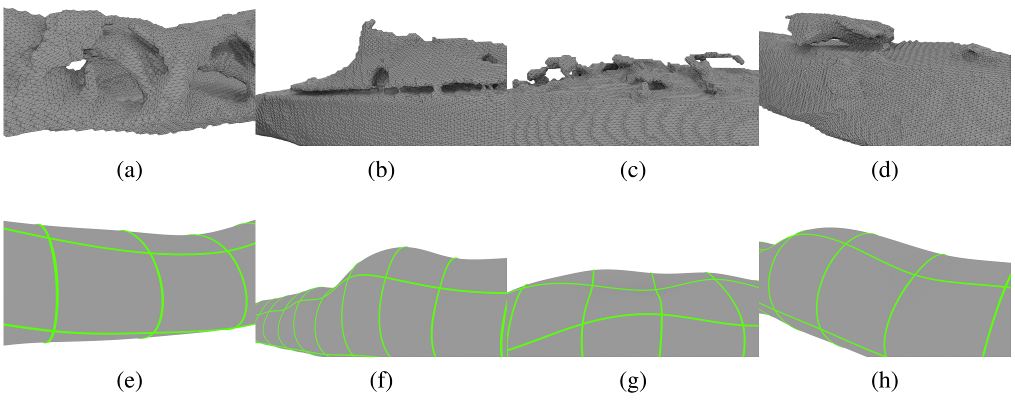


Figure 4.6: *Examples of data artifacts (a)-(d) and the B-spline approximations in those areas (e)-(h).*

problematic for established methods. By processing geometry and topology separately, these difficulties are overcome. This allows for stable reconstructions of the idealized structure while still approximating the unaltered geometry of the input data.

After the data is processed according to the proposed pipeline, a B-spline surface model can be generated by using established B-spline methods.

The presented results show how the method can handle challenging situations in images. For example, surface roughness has almost no impact. Even when larger chunks of material are present near the surface of a fiber, its wall is still reconstructed in a topologically correct way, and surplus material is treated as a local deformation of the surface. The triangulated meshes extracted from the high-resolution input images contain $423\,628$ to $1\,203\,844$ triangles, and they are modeled with 236 to 676 B-spline surfaces.

4.6 Core Reference

- D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen: Topologically robust B-spline reconstruction of fibers from 3D images. *to be published in Topological Methods in Visualization: Theory, Software and Applications.*

Chapter 5

A Local Approach for Computing Smooth B-spline Surfaces on Arbitrary Quadrilateral Base Meshes

5.1 Abstract

A new method for approximating surface data of arbitrary topology by a model of smoothly connected B-spline surfaces is presented. Most of the existing solutions for this problem use constructions with limited degrees of freedom or they address smoothness between surfaces in a post-processing step, often leading to undesirable surface behavior in proximity of the boundaries.

The contribution of this chapter is the design of a local method for the approximation process. A smooth B-spline surface approximation is computed without imposing restrictions on the topology of a quadrilateral base mesh defining the individual B-spline surfaces, the used B-spline knot vectors, or the number of B-spline control points. Exact tangent plane continuity can generally not be achieved for a set of B-spline surfaces for an arbitrary underlying quadrilateral base mesh. The proposed method generates a set of B-spline surfaces leading to a nearly tangent plane continuous surface approximation. The presented examples demonstrate that B-spline approximations can be generated with differences of normal vectors along shared boundary curves of less than one degree. The approach can also be adapted to locally utilize other approximation methods leading to higher orders of continuity.

Key contributions of this chapter are:

- Presentation of a novel local algorithm to construct approximately G^1-continuous

B-spline surface models

- Meaningful methods to visualize and evaluate the quality of a B-spline surface model
- Quality analysis of models obtained by applying the proposed method in comparison with C^0- and G^1-continuous models
- Complexity analysis of the algorithm's computational cost and memory requirements

5.2 Introduction

Approximating large quantities of data points by a relatively small number of B-spline surfaces is an approach commonly used to reduce data size, re-sample objects, or to create a model for simulations, analysis, or reverse engineering. As discussed in Chapter 2, it is fairly easy to construct a C^0-continuous B-spline surface approximation for a set of given data points. However, achieving higher orders of continuity in a general setting is still a challenging task. To obtain smooth and visually pleasing models, it is desirable to generate B-spline approximations with G^1-continuity, i.e., approximations that have continuous tangent planes everywhere.

This problem has been studied quite intensively, and a wealth of literature is available on the topic. Most existing solutions produce very good results, but introduce constraints, e.g., to the order of the B-splines or the type of knot vector, which defines the parameterization of the surface. When the approximation problem must be tackled in a general setting, several established methods have problems, e.g., in case of large or incomplete data.

This chapter proposes a local algorithmic framework for the computation of smooth B-spline models to approximate discrete data. The presented approach is suitable for B-splines of any order and with arbitrary knot vectors. This allows for the construction of a model with low complexity surfaces defined by a small number of control points. The computation of a single surface only requires the data in a small neighborhood to be available. Due to the locality of the approach, it is easy to handle large amounts of data and to parallelly compute individual parts of the result. Most of the existing methods can be localized as well with this method. Its viability is demonstrated by using a computationally more expensive approach for approximate G^1-continuity. Additionally, visualizations and error metrics are discussed to evaluate and compare the quality of a B-spline approximation.

The main idea of the proposed method is to construct smooth approximations of subsets of the data. First, optimal B-spline surfaces are constructed around each corner in the quad-mesh. The control points, necessary for smooth transitions around that corner, are fixed for the subsequent steps. Then, a spline approximation is computed around each boundary curve. Only those control points are kept which are relevant for the smooth transition along that boundary. These steps are equivalent to defining a curve network with tangent information. Finally, the remaining free control points are chosen to approximate the interior of each cell in an optimal way. The main advantage of this approach is that it only requires data in local neighborhoods, which leads to several small systems of equations that need to be solved instead of one large global system.

The method is used to model isosurfaces of binary volume images in the form of B-spline surfaces. These isosurfaces are the result of applying the Marching Cubes algorithm to a number of synthetic and application-driven data sets. To compensate for the discrete nature of the data, a Laplacian smoothing is applied to the resulting triangulation.

5.3 Related Work

The construction of smooth B-spline models to approximate or interpolate discrete data is a topic that has been well-researched over the past decades. While B-splines are fairly easy to understand and use, computing a G^1-continuous model of B-spline surfaces is not a trivial task.

Peters and Fan [60] discuss the complexity of constructing smooth piecewise polynomial surfaces and derive essential conditions that need to be met in order to construct a G^1-connected model of B-splines. A central statement is that a G^1-continuous B-spline model of arbitrary topology with bi-cubic B-spline surfaces requires knot vectors to have at least two internal double knots. Therefore, for a knot vector with only single interior knots it is not guaranteed that there always exists an exact G^1-continuous B-spline model.

However, a variety of methods exist to solve specialized instances. Some accept certain restrictions on the data or B-splines, others aim only for approximate G^1-continuity.

Milroy et al. [49] present a stitching approach by defining G^1-errors as difference between normal vectors on a fixed number of points along boundary curves and then minimizing this term. This approach yields good results but is sensitive to the size of the model due to the complexity of the nonlinear constraints.

Lai and Ueng [43] show that even G^2-continuity can be achieved fairly easily when simplifying the continuity constraints to colinearity of derivatives along surface boundaries. Since this leads to singularities when multiple surfaces meet in one point, the method is best suited for constructing pairs or stripes of surfaces.

According to Shi et al. [72] a construction of a G^1-continuous B-spline model with single interior knots requires at least bi-quintic splines. For those, they provide explicit formulas for the computation of the control points. To deal with irregular corner points, they project all outgoing tangents onto the plane defined by an averaging normal vector.

Lin et al. [45] approximate data with bi-quintic Bézier surfaces. They compute a global curve network and generate normal information along those curves. Then, they compute each surface individually such that it interpolates the boundary curves and the normal vectors along those curves to obtain G^1-continuity.

Yoo [83] presents an approach that utilizes a global implicit surface to generate a quad-mesh in addition to sample points that are then interpolated by one B-spline surface per quad. Smoothness between individual B-splines is obtained by also interpolating the normal vectors of the implicit surface along the surface boundaries.

Zhao et al. [85] introduce a numerical approach for point cloud approximation. First, they approximate the data using a set of B-spline surfaces with no continuity constraints, even allowing different knot vectors. Then, the control points are adjusted by numerical methods to also obtain approximately C^0 and G^1-continuous transitions.

Closely related to the task of approximating data is the interpolation problem, where the vertices of a given (quad-)mesh are to be interpolated by a number of smooth surface patches. These approaches need to deal with the same continuity constraints and can be adapted to also allow data approximation. Most of these methods express the control points of the B-spline as linear combinations of the mesh vertices. Eck and Hoppe [17] show how to modify such an interpolation method and use it within the context of a least squares approximation.

Hahmann et al. [31] describe a localized method to interpolate quad-meshes by using 2×2 bi-cubic Bézier patches per quad. Their algorithm consists of three steps, calculating control points around corners and edges first to ensure G^1-continuity there, before defining the interior control points. However, this method cannot be applied to arbitrary meshes which is then addressed by a more general approach by Bonneau and Hahmann [7] using a similar idea of a localized construction to obtain 2×2 bi-quartic Bézier patches per quad.

Yoshihara et al. [84] show how to make use of an interpolation method for data approximation. They approximate the data with a Catmull-Clark subdivision surface. The control structure is then projected onto its limit surface to obtain a dense quad-mesh. Vertices of that mesh are interpolated using an approach by Fan and Peters [19] which computes a model of 3×3 bi-cubic Bézier patches per quad.

Fan and Peters also present an improved version of that algorithm [20] where they construct either a single bi-cubic Bézier patch or a B-spline surface with at most two inner double knots per quad, depending on the valence of the surrounding vertices. This construction even provides C^2-continuity along regular edges, i.e., edges that connect vertices with valence 4.

A more formal approach to the problem of constructing smooth splines has been proposed by Mourrain et al. [55]. They provide a detailed analysis and in-depth theoretical background on the dimensionality and bases of the space of smoothly connected splines. While they look at general splines defined over arbitrary topologies, Blidia et al. [6] provide a similarly thorough investigation of spline spaces on quad-meshes with 4-split elements.

Most of the above mentioned methods can be used to generate very good, smooth B-spline surfaces with certain restrictions, e.g., the spline order is often fixed and by using Bézier patches they already define the knot vector and number of control points. The approach proposed in this chapter aims to provide a more general solution for data approximation and does not impose restrictions on the topology of the quad-mesh, the order of the B-splines, or the used knot vectors.

5.4 Method

In this chapter, it is assumed that the input is given in the form of a triangular surface mesh. Furthermore, a quad-mesh, partitioning the data into individual cells, and a parameterization of each quad-cell are required. Sections 2.1 and 2.2 cover these steps when starting with only a mesh.

5.4.1 Nonlinear Constraints

Existing methods that achieve G^1-continuity have only limited degrees of freedom when control points are computed and impose constraints to knot vector and order, e.g., [20, 72]. This section discusses pointwise G^1-constraints that do not have any of these restrictions for the cost of only approximate G^1-continuity.

Without loss of generality, let S_a and S_b be two B-spline surfaces meeting at $S_a(1,t) = S_b(0,t), t \in [0,1]$. Similar to [49], the G^1-error along the shared boundary curve is defined as

$$r_{a,b}(t) = \left\| \frac{n_a(1,t)}{\|n_a(1,t)\|} - \frac{n_b(0,t)}{\|n_b(0,t)\|} \right\|, \tag{5.1}$$

where n_a and n_b are the normal vectors of the two surfaces. Figure 5.1 provides an illustration of the G^1-error at a point on the shared boundary between two surfaces.

The B-spline basis functions and their derivatives are piecewise polynomial functions. Hence, it can be expected that the G^1-error between two points on the boundary with $r_{a,b}(t_1) = r_{a,b}(t_2) = 0$ is sufficiently small as long as $|t_1 - t_2|$ is small. Enforcing G^1-constraints of the form $r_{a,b}(t_i) = 0$ with $t_i = i/(n_{G^1} - 1)$ in n_{G^1} discrete points along shared boundary leads a low overall G^1-error on that boundary.

A set of B-spline surfaces with approximate G^1-continuity can be found by adding these constraints to the C^0-continuous system, given by 2.14. Let Γ be the set of all index tuples (a,b) of surfaces that are supposed to be joined with G^1-continuity. Then,

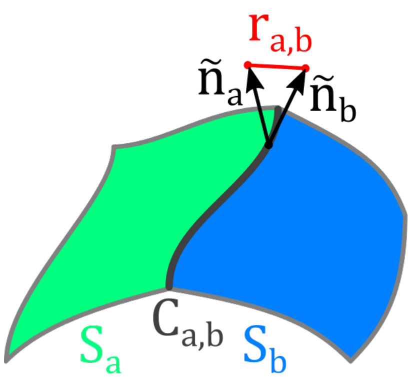


Figure 5.1: *The G^1-error $r_{a,b}$ at a point on the boundary of two B-spline surfaces is defined as the length of the difference vector between the unit-normal vectors $\tilde{n}_a$ and $\tilde{n}_b$.*

the following problem has to be solved:

$$\begin{aligned} &\text{minimize } (1-\lambda)\widehat{E}_{\text{LS}} + \lambda\widehat{E}_{\text{TP}} && (5.2)\\ &\quad\text{s.t. } G\widehat{b} = 0 \\ &\qquad r_{a,b}(t_i) = 0 \qquad\qquad \forall(a,b)\in\Gamma, i = 1,...,n_{G^1}. \end{aligned}$$

The G^1-constraints can be reformulated as $(r_{a,b}(t_i))^2 = 0$. This allows for a more efficient implementation as it negates the outer square root introduced by taking the norm in Equation (5.1).

As stated in [60], a perfect G^1-continuous construction may not always be possible in a general setting. Furthermore, nonlinear constraints can be hard to handle for some numerical solvers. This can be addressed by reformulating (5.2) into

$$\begin{aligned} &\text{minimize } (1-\lambda')\left((1-\lambda)\widehat{E}_{\text{LS}} + \lambda\widehat{E}_{\text{TP}}\right) + \lambda' \sum_{(a,b)\in\Gamma}\sum_{i=1}^{n_{G^1}} r_{a,b}(t_i) && (5.3)\\ &\quad\text{s.t. } G\widehat{b} = 0. \end{aligned}$$

This does not anymore guarantee G^1-continuity in any point, but allows for a more robust implementation.

Note that due to the nonlinearity of the G^1-constraints, such a system needs to be solved with nonlinear solvers, e.g., NLopt [39]. The computational complexity is far higher than for linear systems, which makes this approach impractical on a global scale. However, Section 5.4.2 discusses a local construction scheme in which only small neighborhoods are considered.

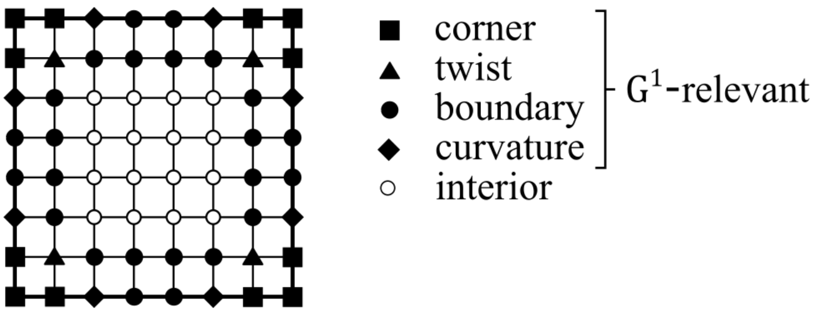


Figure 5.2: *Control points and G^1-continuity. The figure shows the control points relevant for ensuring G^1-continuity of a B-spline control mesh with 8×8 control points.*

5.4.2 Local Multi-step Algorithm

The first derivatives of B-spline surfaces along their boundaries only depend on two rows of control points, i.e., $b_{i,j}$, with one index being in $\{1, 2, n_c - 1, n_c\}$. Hence, inner control points can be chosen freely without affecting G^1-continuity. Control points that have an influence on the derivatives along a surface boundary are called *G^1-relevant control points* for that boundary. At corner points of a surface where two boundary curves meet, both tangents in u and v direction are defined by only three control points, i.e., the control point in that corner as well as the neighboring control point in each direction. The first inner control points in each corner, the so-called twist points, play a special role. They do not have an impact on the normal vector in the corner itself but along both boundaries in a neighborhood around the corner; therefore, they must be chosen to ensure compatibility with both neighboring surfaces. These classifications are illustrated in Figure 5.2. As shown in Section 5.5, using these points alone already produces good results. If the goal is exact G^1-continuity, it is also necessary to consider the so-called curvature points, i.e., the second control point from each corner on the boundary, as they are also a part the G^1-compatibility constraints discussed in [60].

The main idea of the local algorithm is similar to other approaches [7, 20, 31, 72] that construct a global result by performing various local computations in subsequent phases. Here, instead of computing only individual control points in a phase, entire surfaces are computed to approximate small subsets of the data. Relevant control points of these surfaces are used in the final result, while the rest is discarded and re-computed in the next step. The individual phases are:

1. *Corner Phase*
   For each corner of the quad-mesh, a B-spline approximation of the incident cells is computed while ensuring G^1-continuity around that corner. The G^1-relevant control points around the corner are stored for the final result. Without loss of generality, let the control points of the surrounding B-spline surfaces be aligned

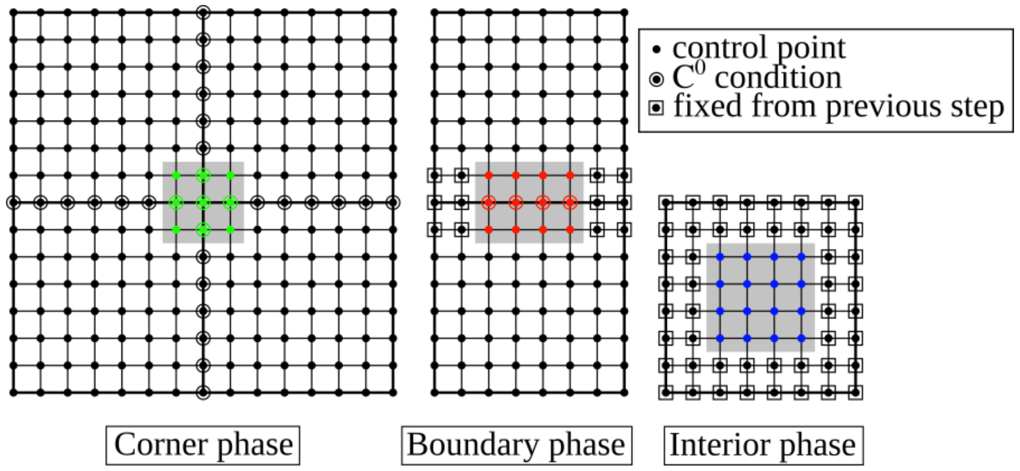


Figure 5.3: *Phases of control point computation. The individual images highlight the control points used in subsequent phases.*

such that $b_{0,0}$ is associated with the corner currently being considered. Then, the points $b_{0,0}$, $b_{0,1}$, $b_{1,0}$, and $b_{1,1}$ of all these surfaces are kept.

2. *Boundary Phase*
   For each boundary between any two quad-cells, a G^1-continuous B-spline approximation of those cells is constructed. The G^1-relevant control points around the end points of the boundary are set as the result of the last phase (corner phase). All other control points can be chosen freely again. The G^1-relevant control points along the shared boundary are kept for the final result.

3. *Interior Phase*
   For each cell of the quad-mesh, a final B-spline approximation is performed, considering all G^1-relevant control points already known from the previous two phases. The interior control points are used to finalize the global result.

The specific spline approximation algorithms used in the individual phases can be chosen freely. Here, the objective function and constraints from Section 5.4.1 are used to demonstrate the viability of this approach. Section 5.3 provides a discussion of other viable algorithms. These methods can also be used in this local framework. To assess the quality of the resulting surface approximation, where only approximate G^1-continuity is achieved, evaluation criteria are described in Subsection 5.4.3.

A B-spline surface computed by the interior phase is the final approximation of the corresponding quad-cell. To compute a surface in this phase the results of the corner and boundary phases affecting that cell are needed. Only the data of the neighboring quad-cells needs to be known at this time. This means that the individual phases do not

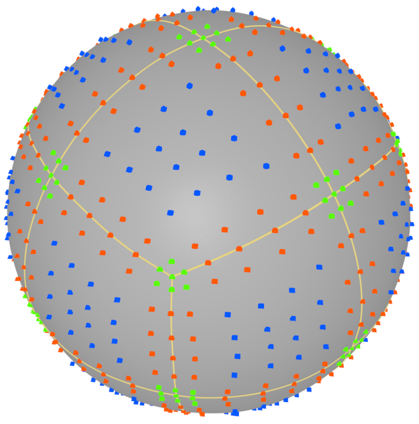


Figure 5.4: *Final B-spline control meshes. The figure shows the contributions of control points produced by the three phases. Control point colors indicate the phases that generated them, see Figure 5.3.*

need to be completed for the entire quad-mesh before starting surface computation in the final phase. This fact makes it possible to apply this approach even when only parts of the data can be loaded into memory at once.

5.4.3 Quality Evaluation

Depending on the application it might suffice if the constructed B-spline model passes a visual inspection. However, for a formal evaluation, well-defined quality criteria are desirable.

A good error measure to evaluate a spline approximation is the distance to the original object/surface. Here, the *approximation error* of each data point is defined as the distance to the closest point on the B-spline surface, given by (2.6). Due to the iterative scheme described in Section 2.3.1, it can be expected to find the closest surface point approximately by evaluating the B-spline at the parameter values of the data point, i.e., by evaluating (2.7). To obtain the most accurate result for evaluation purposes, it is also possible to compute the closest surface points by numerically solving the nonlinear Equation (2.6) which yields optimal parameter values (u, v). The distances are measured between the B-splines and the given data points, which are the vertices of the input mesh. When these data points are obtained via a more complex pipeline, pre-processing steps can add additional error which has to be considered separately.

The approximation error of the whole surface can either be analyzed via statistical measures of the individual errors or qualitatively by plotting the error values, color-coded onto the B-spline surfaces, see Figure 5.5a. When the goal is to smooth out irregularities in the original data, the plot will show higher errors in these irregular regions. To understand where data points are above or below the approximating B-splines, a signed error value is computed. The sign is defined by the scalar product between the error vector of a data point and the normal vector at its closest point on the B-spline surface.

In order to establish a statistical measure for overall approximation quality, the root mean square (RMS) error is computed, given by

$$E_{\text{RMS}} = \sqrt{\frac{1}{n_p}\sum_{i=1}^{n_p} d_i^2},$$

with n_p being the total number of data points and d_i denoting the approximation error of the i-th data point. Furthermore, the maximum error is measured as reference. To enable a meaningful comparison between data sets of different sizes, the error is expressed as a percentage of the "diameter" of the data set, defined as the largest distance between any two data points, see [17, 84].

The input mesh might already exhibit undesirable geometric behavior, e.g., extensive variation of normal vectors resulting from an isosurfacing method, like the Marching Cubes method. These effects have an impact on statistical quality measures and color plots. Since they are typically a byproduct of an earlier step in the data processing pipeline, they generally do not reflect the proper geometry of an original object. A good spline approximation scheme eliminates or greatly reduces such undesirable irregularities, but, numerically, this smoothing effect impacts error computation. Ultimately, a user must interpret quantitative error values and compare results with the input.

In addition to the actual approximation error, smoothness is evaluated in the form of tangent plane continuity. In a surface model composed of B-spline surfaces with non-degenerate control points and knot vectors, the only regions where discontinuities in first derivatives can occur are along the boundary curves between individual surfaces. Based on Equation (5.1), the normal vectors of two adjacent surfaces along their shared boundary curve are considered. The G^1*-error* is defined as the angle between these vectors. These values are computed along all surface boundaries. The result is then color-coded and visualized on the surface enabling an interactive visual assessment of the result, see Figure 5.5b.

To calculate meaningful measures that summarize the G^1-error of the entire model via a single number, the average G^1-error is computed by integrating the G^1-error of the entire surface model and dividing it by the total length of all boundary curves. Let $C_{a,b}(t)$ be the shared boundary curve between B-spline surfaces S_a and S_b. Since $C_{a,b}(t)$ is usually not parameterized by arc length, the parameter interval $[0, 1]$ can get distorted in $\mathbb{R}^3$. Therefore, simply computing a G^1-error in the parameter domain by using the integral

$$\int_0^1 r_{a,b}(t)dt$$

does not generally yield an acceptable result for the error along the curve. Hence, the local errors are integrated along the curve in $\mathbb{R}^3$. The definition of $r_{a,b}(t)$ is formally extended to the 3D setting by

$$r_{a,b}(C_{a,b}(t)) = r_{a,b}(t).$$

Then, the integral along a curve parameterized over $[0, 1]$ is reformulated using

$$\int_{C_{a,b}} g(C_{a,b})ds = \int_0^1 r_{a,b}(C_{a,b}(t))\|\dot{C}_{a,b}(t)\|dt.$$

The result is normalized through division by the length of all boundary curves

$$L = \sum_{(a,b)\in\Gamma} \int_0^1 \|\dot{C}(t)\|dt.$$

Hence, the average G^1-error of the surface model is

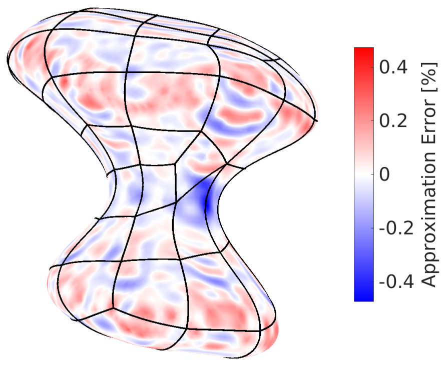
$$E_{G^1} = \frac{1}{L} \sum_{(a,b)\in\Gamma} \int_0^1 r_{a,b}(t) \left\|\dot{C}_{a,b}(t)\right\| dt.$$

As an upper error bound, the maximum G^1-error among all points on the surface model is computed as well.

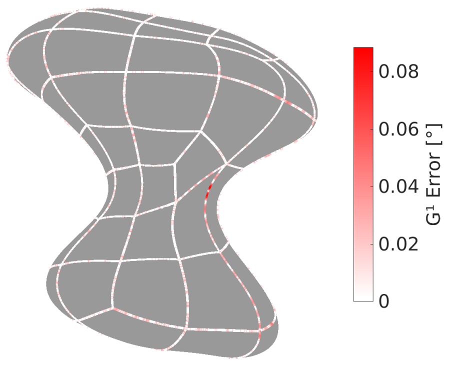
5.5 Results

This section covers the results of applying the proposed method to several synthetic and real-world data sets. To evaluate the quality of the resulting B-spline models, the approximation and G^1-errors are measured as described in Section 5.4.3. The B-spline models are of order $k = 3$, have 6×6 control points per surface, and use a uniform knot vector with multiple end knots, i.e., $\tau = (0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1)$. The experiments have shown that the choice of $\lambda = 0.05$ for the smoothness term leads to good results.

The results are compared to models that are only C^0-continuous. These models are obtained by minimizing (5.2) without G^1-constraints. Since no deformations are imposed by the G^1-continuous transitions, the weight of the smoothness term is reduced to $\lambda = 0.01$.



(a) *Approximation error shows surface areas above or below original data.*



(b) G^1*-error shows how much normal vectors differ on surface boundaries.*

Figure 5.5: *Error visualization. Error values mapped onto the surface support a qualitative evaluation of an approximation.*

Furthermore, the results are also compared to the G^1-exact construction by Fan and Peters [20], see Section 2.3.2. Due to the restrictions imposed by exact G^1-continuity, the model uses the required minimal number of 8×8 control points per B-spline surface and the knot vector $\tau = (0,0,0,0,1/3,1/3,2/3,2/3,1,1,1,1)$. Since the resulting surface model is G^1-continuous by construction, its G^1-error is zero everywhere.

The initial meshes were obtained by using the Marching Cubes algorithm to extract isosurfaces from given binary 3D volume images. A Laplacian smoothing operation was applied to compensate for discretization effects and artifacts resulting from the piecewise linear interpolation underlying the Marching Cubes method. Quad-meshes were constructed using the method by Eck and Hoppe [17], see Section 2.1. The test data consists of several synthetically generated models with different geometry and topology and two real-world, application-driven data sets. The first real-world data set is a stone particle used in composite materials. It has simple topology. A researcher's main interest in this case is the geometry of the surface. The second real-world data set consists of two steel fibers found in fiber-reinforced concrete. The fibers' surfaces are quite smooth, and the primary interest here is the overall shape and connectivity of objects. Both data sets were given as 3D images, generated via scanning the objects with micro-computed tomography at the Fraunhofer Institute for Industrial Mathematics. Figures 5.6-5.8 show the initial meshes, the resulting B-spline approximations (including the subdivisions into individual surfaces), and color plots visualizing approximation error.

The results provided in Table 5.1 show that the proposed method provides a good trade-off between an accurate reconstruction of the data and tangent plane continuity along shared boundaries. Approximation error values compare well to those of the

	Approximation Errors					
	C^0-Method		**Local G^1-Method**		G^1-Method	
Object	RMS[%]	Max[%]	RMS[%]	Max[%]	RMS[%]	Max[%]
Sphere	0.004	0.025	0.006	0.032	0.048	0.359
Cube	0.092	1.330	0.154	1.883	1.050	6.176
Spring	0.013	0.110	0.015	0.127	0.219	1.191
Torus	0.006	0.046	0.008	0.055	0.074	0.418
As. Torus	0.007	0.074	0.005	0.034	0.060	0.399
Tanglecube	0.007	0.080	0.007	0.077	0.168	1.306
Stone	0.041	0.737	0.066	0.891	0.430	2.445
Fibers	0.002	0.018	0.001	0.031	0.024	0.361

(a)

	G^1-Errors					
	C^0-Method		**Local G^1-Method**		G^1-Method	
Object	Avg[°]	Max[°]	Avg[°]	Max[°]	Avg[°]	Max[°]
Sphere	0.522	2.305	0.006	0.048	0	0
Cube	3.795	83.955	0.009	0.112	0	0
Spring	3.132	30.640	0.007	0.108	0	0
Torus	1.271	9.482	0.005	0.059	0	0
As. Torus	2.786	11.181	0.005	0.091	0	0
Tanglecube	3.436	21.076	0.011	0.207	0	0
Stone	7.000	84.278	0.007	0.099	0	0
Fibers	5.133	31.278	0.016	0.292	0	0

(b)

Table 5.1: *Approximation (a) and G^1-error values (b) for test data sets of a C^0-continuous approximation method (left), the novel local method (middle), and the G^1-continuous method by Fan and Peters (right).*

models with only C^0-continuity. The comparison also shows that even though the G^1-continuous models consist of more control points, satisfying the restrictions needed for exact G^1-continuity severely limits the available degrees of freedom when approximating a dense set of points. As a consequence, the RMS error is about a factor of 10 higher than for the presented local method.

5.5.1 Complexity

When computing control points in a local neighborhood around corners and boundaries of the quad-mesh, B-spline surfaces are constructed for all incident quad-cells, even though only a subset of the control points is contributing to the final result. The benefit of this approach becomes evident when considering the computational complexity of the algorithm. Assume that $t(n)$ describes the computational cost needed to perform a G^1-continuous approximation with n B-spline surfaces. Let V, E, F be the number of corners (vertices), boundaries (edges), and cells (faces) of the quad-mesh.

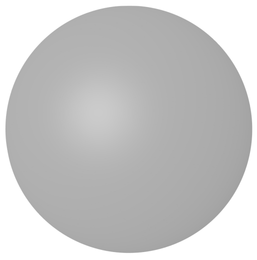
In the quad-meshes used in this chapter, no corner had a valence above 8. In theory that number could be reduced even more by applying optimization algorithms to generate a more even distribution of valence values. Let n_v be the highest valence of any corner in the quad-mesh. The largest systems that has to be solved during the B-spline computation consists of at most n_v surfaces, i.e., the computational cost for the first phase is at most $V \cdot t(n_v)$.

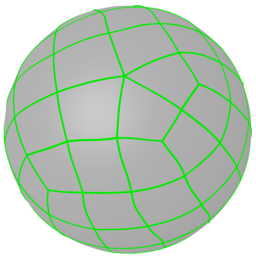
During the boundary phase only pairs of neighboring quad-cells are considered. Assuming the potential speed-up resulting from having already a number of fixed control points from the first phase is neglected, the computational cost of the boundary phase is $E \cdot t(2)$.

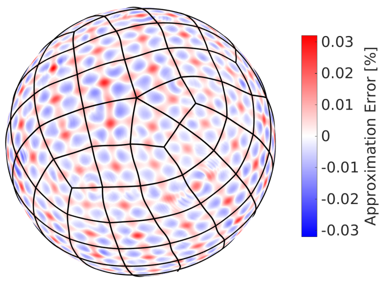
Finally, in the interior phase, every G^1-relevant vertex is considered to be a given constant. Hence, a solution must be found for only $(n_c - 4)^2$ unknown control points. To provide a more intuitive upper bound of the method, the computational cost of the final phase is simply expressed as $F \cdot t(1)$.

This leads to a total computational cost of at most $V \cdot t(n_v) + E \cdot t(2) + F \cdot t(1)$. Solving a global system would have a cost of $t(F)$. If it were possible to define a system that scales linearly, i.e., $t(F) = F \cdot t(1)$, it would be computationally cheaper to solve the system globally. Unfortunately, to the author's knowledge, such an approach does not exist yet. Usually, $t(n)$ is at least in $\Omega(n \log n)$ or worse. Note that the same argument also holds for memory requirements. The total amount of memory required to represent the systems of equations for several small systems is by far less than the amount required for a global system.

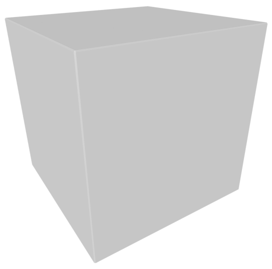
Besides having a lower runtime and memory complexity, the presented approach can also make use of parallel computing. In practice, this means that the computation can be sped up at the cost of higher memory requirements, allowing the system to adapt its performance to the available hardware.

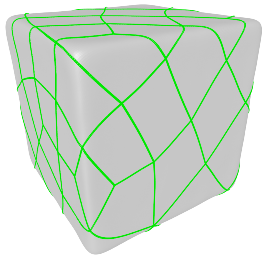

(a) *Triangle mesh (*753 774 *vertices)*

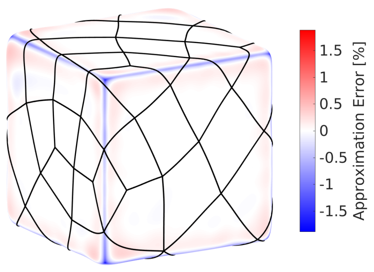

(b) *B-spline model (*130 *surfaces)*



(c) *Approximation error*

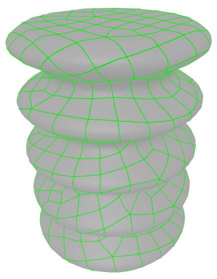

(d) *Triangle mesh (*48 600 *vertices)*

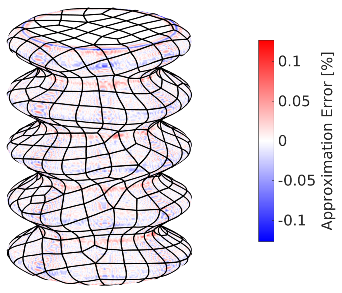

(e) *B-spline model (*60 *surfaces)*



(f) *Approximation error*

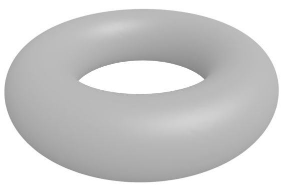
(g) *Triangle mesh (*861 858 *vertices)*

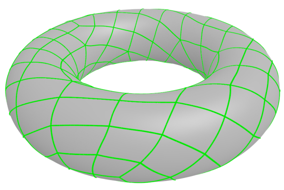

(h) *B-spline model (*538 *surfaces)*

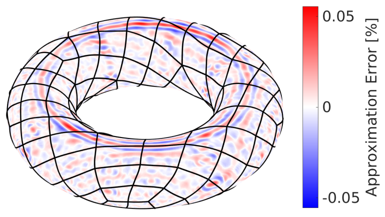


(i) *Approximation error*

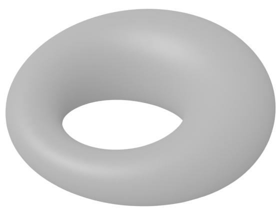
Figure 5.6: *Sphere (top), cube (middle), and spring (bottom). These examples share a simple topological situation. However, their geometric complexity varies. The sphere has constant curvature while the cube consists of several flat surfaces. The spring consists of regions of strongly varying curvature. Errors are highest close to edges and regions of changing curvature.*

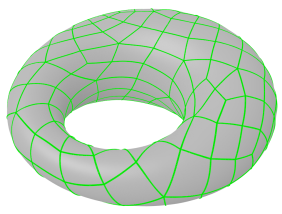

(a) *Triangle mesh (*426 224 *vertices)*

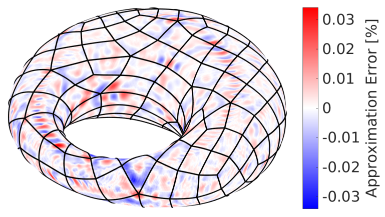

(b) *B-spline model (*180 *surfaces)*



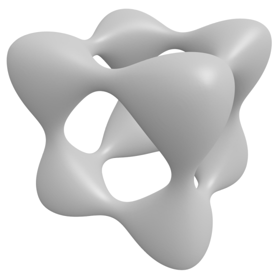
(c) *Approximation error*

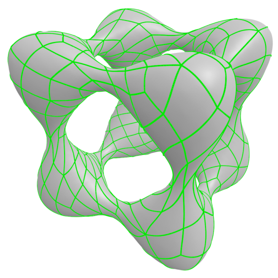

(d) *Triangle mesh (*151 954 *vertices)*

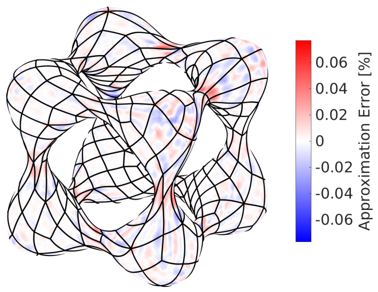

(e) *B-spline model (*248 *surfaces)*



(f) *Approximation error*

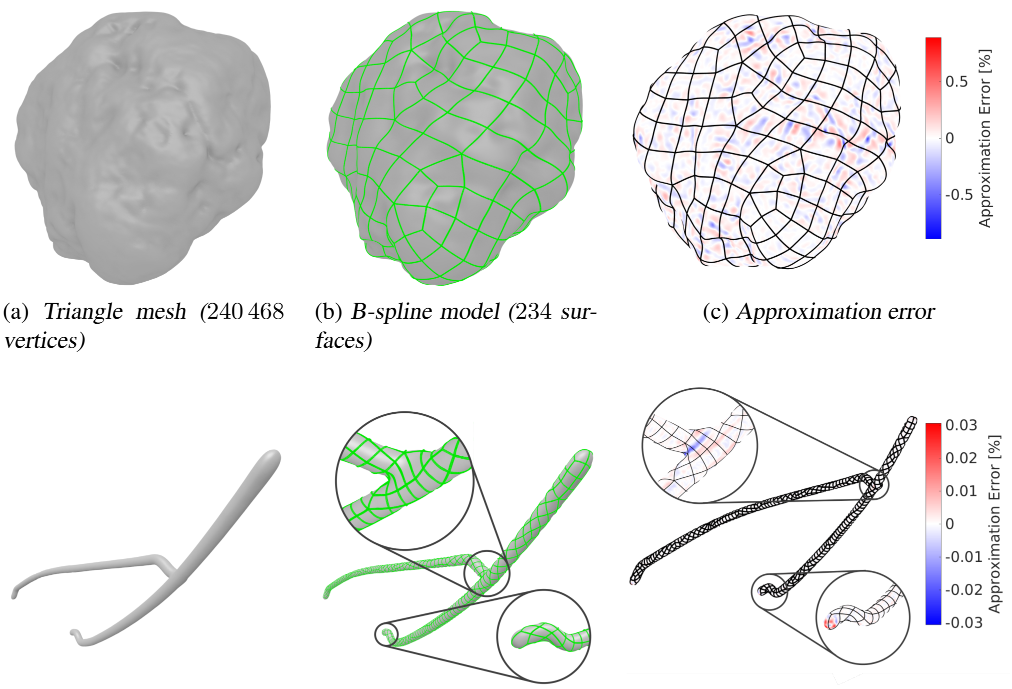

(g) *Triangle mesh (*383 718 *vertices)*


(h) *B-spline model (*452 *surfaces)*



(i) *Approximation error*

Figure 5.7: *Torus (top), asymmetric torus (middle), and tanglecube (bottom). Even with increasing topological complexity, the proposed method still produces high-quality results. Errors are highest in regions where structures transition from strong to weak surface bending.*



(a) *Triangle mesh (*240 468 *vertices)*

(b) *B-spline model (*234 *surfaces)*

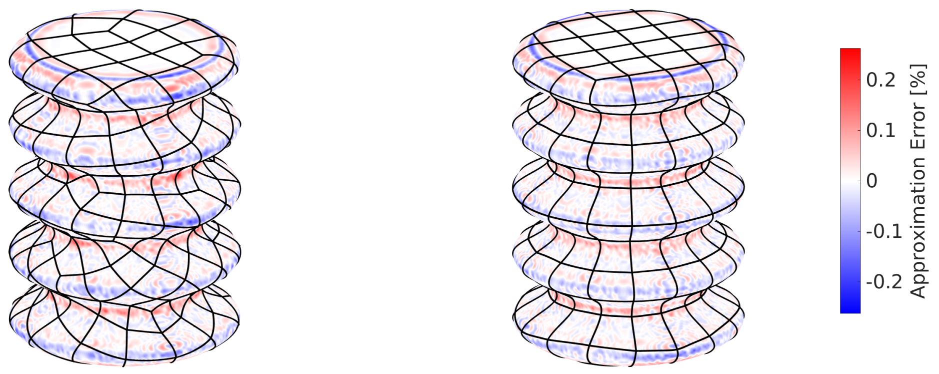
(c) *Approximation error*

(d) *Triangle mesh (*439 464 *vertices)*

(e) *B-spline model (*718 *surfaces)*

(f) *Approximation error*

Figure 5.8: *Stone (top) and fibers (bottom). Sharp features are smoothed out. The overall approximation error is still low, and important shape characteristics are well-preserved.*



(a) *No. of quads:* 258
RMS error: 0.037%
Max error: 0.262%
Avg G^1: 0.014°
Max G^1: 0.171°

(b) *No. of quads:* 224
RMS error: 0.033%
Max error: 0.205%
Avg G^1: 0.005°
Max G^1: 0.079°

Figure 5.9: *Impact of quad-mesh. (a) Quad-mesh created automatically; (b) quad-mesh generated manually with cell boundaries in principal curvature directions.*

5.5.2 Impact of the Quad-mesh

The error visualizations indicate that the quality of the approximation depends on the quadrilateral partition of the data. Strong variations in curvature or multiple fine details within a single surface lead to a higher approximation error. This behavior suggests that a partition with a limited amount of surface complexity per cell is desirable.

Figure 5.9 provides a comparison between an automatically generated quad-mesh and one that has been manually constructed. In both cases, each B-spline surface is defined by 6×6 control points. The manually constructed quad-mesh is designed in such a way that individual surfaces only contain curvature situations that can be handled easily. Even though the automatically generated quad-mesh has a higher number of surfaces, the quality of the B-spline approximation is worse than the approximation based on the manually created quad-mesh.

5.5.3 Conclusions

This chapter presented an approach for surface approximation that works locally. Given a point set and a quadrilateral base mesh, the algorithm computes a smooth B-spline surface model that closely approximates the input. The direct comparison shows that it offers a good trade-off between a purely C^0-continuous algorithm and an exact G^1-continuous scheme. The method is demonstrated using nonlinear pointwise constraints. Approximation specific steps and continuity constraints can easily be exchanged by other algorithms. The main contribution is the local approach, making it possible to locally use other approximation methods that usually could not be applied due to data size or memory limitation.

In addition, meaningful ways to analyze the resulting B-spline approximation quality were discussed. Statistical measures are computed which can be used to trigger a re-computation with different parameters to improve an approximation. The colored surface quality visualizations are effective to ensure that desired approximation quality criteria are met.

Due to the locality of the algorithm, not all data needs to be present in memory simultaneously. For each B-spline surface to be computed, the method only requires the data of the incident cells. Thus, it is possible to merge computation results produced during different phases. This enables the processing of data sub-sets prior to progressing to other data parts and thereby minimizes expensive I/O operations.

The presented algorithm does not rely on a specific choice of continuity constraints. Any G^1-continuous method can be used for the local surface construction. The method can utilize B-splines of any order and it can be generalized to arbitrary dimensions. Knot vectors do not necessarily need to consist of uniformly spaced knot values. However, for a quad-mesh of general topology, they must be equal for all B-spline surfaces and all dimensions.

5.6 Core Reference

- D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen: A local approach for computing smooth B-spline surfaces on arbitrary quadrilateral base meshes. *to be published in ASME Journal of Computing and Information Science in Engineering*.

Chapter 6

Feature Driven Viewpoint Placement for Model Based Surface Inspection

6.1 Abstract

The goal of visual surface inspection is to analyze an object's surface and detect defects by looking at it from different angles. Developments over the past years have made it possible to partially automate this process. Inspection systems use robots to move cameras and obtain pictures that are evaluated by image processing algorithms.

However, the task of automation is intrinsically tied to the application and the environment of the inspection. Setting up these systems or adapting them to new models is primarily done manually by a trained expert. A key challenge is to define camera viewpoints, i.e., position and orientation of the camera, from which the images are taken. The number of viewpoints should be as low as possible while still guaranteeing an inspection of the desired quality. System engineers define and evaluate configurations that are improved based on a time-consuming trial and error process leading to a sufficient, but not necessarily optimal, configuration. With the availability of 3D surface models, the system configuration can be done virtually.

This chapter presents a new, scalable, approach to determine a small number of well-placed camera viewpoints for optical surface inspection planning. The initial model is approximated by B-spline surfaces. A variety of meaningful geometric feature functionals is defined and used for an adaptive, non-uniform surface sampling that is sparse in geometrically low-complexity areas and dense in regions of higher complexity. This sampling is accomplished by subdividing the B-spline surface model into segments for which the selected feature functional falls bellow a user-defined threshold. The presented approach is applicable to solid objects with a given 3D surface model. It

makes camera viewpoint generation independent of the resolution of the triangle mesh and it improves previous contributions with respect to the number of viewpoints and their relevance.

Results from applying the method to objects of varying geometric and topological complexity are shown and analyzed. Computational cost of the algorithm is low and the resulting distribution of viewpoint candidates can easily be steered by a low number of meaningful parameters. Furthermore, an in-depth analysis of the feature functionals is provided, highlighting their respective strengths and making suggestions for favorable use-cases.

The main contributions of this chapter can be summarized as follows:

- A novel algorithm to compute viewpoint candidates for visual surface inspection.
- Introduction of various geometric feature functionals on which the non-uniform surface sampling is based.
- Validation of the presented approach by applying it to sample objects of varying complexity.
- Analysis and comparison of the introduced feature functionals with respect to computational cost, intuition, and stability.

6.2 Introduction

With the abundance of various sensing methods, almost every produced object undergoes a quality assurance process. Because of their availability and application versatility, optical sensors are commonly utilized, both as a support, as well as a primary sensing device. When it comes to surface quality inspection, they are especially useful as there are many advanced machine vision tools available (e.g. [8, 23, 40]). Moreover, they enable the possibility of inspecting both optical and spatial properties of the object. Over the years, significant effort has been put into inspection process automation. An in-depth analysis of the topic and various ways to automate machine vision tasks have already been covered by literature [4].

One widely known, but rarely emphasized or discussed, fact is that the overall visual inspection automation is inextricably connected to the application and the environment of the inspection, thus requiring an expert approach at the point of image acquisition system design and hardware configuration. The design of the system affects the acquired images and consequently enables or hinders the machine vision algorithms analyzing the given images.

The expert approach is required when a decision concerning the hardware placement is needed. The system design is an iterative process consisting of four steps:

assessment of the product geometry and its reflection properties, *construction* of a prototype configuration, *verification* using several machine vision algorithms and *configuration adjustment*. It is a process which works well, given a low geometric complexity of the surface. An increase in complexity makes the system design more tedious and harder to evaluate in terms of inspection completeness and accuracy, thus raising the need of developing an automated design (planning) tool. With general shift of the industry towards more flexible production lines producing small size batches of geometrically strongly varying objects, this challenge becomes more relevant. For example, additive manufacturing (3D printing) is one important application, where no adaptive inspection solution yet exists.

During the assessment phase, an engineer will observe the product from various angles, learning about the object's characteristics (features) such as visible or potentially occluded surfaces (geometry) and light response behaviour. Based on the observed features and own experience, the engineer will infer the initial acquisition system configuration, relative to the inspected object, and introduce further refinement. Even for an expert, this process is often based on trial and error and requires many iterations and tests. As a consequence, setting up an inspection system manually can take days or even weeks until it meets the required quality criteria. The described process can be translated into more formal steps: *1) read object features*, *2) consider potential configurations*, *3) choose the most promising configuration* (based on object features and expert knowledge). The first two steps belong to object space exploration and the third to optimization. Once a set of optimal viewpoints is determined, the physical system prototype has to be configured. Cameras must be placed either statically or dynamically using a manipulator. In case of a manipulator, an additional path planning step is required for viewpoint traversal. Figure 6.1 illustrates this pipeline.

Visibility of a defect and probability of detection depend on the relative angles between camera, surface, and the illumination [26]. An image of a free form object is likely to exhibit a large variation of those angles, making it challenging to heuristically ensure every part of the object is inspected from a proper position.

In this chapter, the object space exploration is considered to be the main challenge. An approach to solving it is proposed based on a 3D model of the object, using its geometric features as viewpoint planning criteria. The obtained viewpoint candidate set is derived directly from the specific object features which are to be inspected. It has a controllable distribution for areas of varying complexity. The impact of the viewpoint candidate sets is further evaluated using a next best view greedy algorithm.

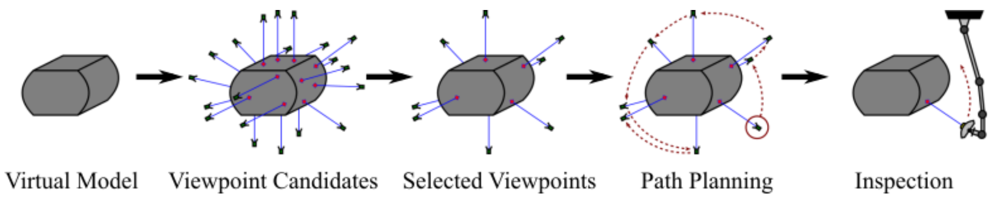


Figure 6.1: *Overall pipeline. A set of viewpoint candidates gets produced by placing virtual cameras and directing them at certain pivot points on the objects surface. From these candidates, a lower number of optimal viewpoints get selected so that the object is still sufficiently covered. Then, a camera path is planned considering the degrees of freedom of the used inspection system. Finally, the physical inspection system carries out the inspection by moving the camera along the computed trajectory and taking pictures at the selected viewpoints.*

6.3 Related Work

The need for automated inspection planning tools has been recognized for over 30 years and a significant amount of effort has been put into researching it. Early work of Cowan and Kovesi [10] present an analytical approach, suggesting that for each surface polygon, a 3D solution space should be formed by solving inspection requirements posed as analytic constraints. The optimal solution is then reached through intersection of all obtained solution spaces. Their work focuses on a low complexity surface patch, since such an approach is computationally expensive.

Sakane et. al. [67, 68] use uniformly and adaptively tessellated spheres for solution space discretization and further evaluate the sphere facets by two criteria: reliability for recovering the surface normal vectors and detectability of the surface by camera and illuminators. The size of the viewpoint candidate sets is briefly commented on, stating that the desired number of viewpoint candidates should be manually chosen to balance the trade-off between planning complexity and accurate inspection representation.

Tarabanis et. al. contribute to the inspection planning field through their work on the MVP (machine vision planner) system [76, 75], where the use of viewpoint candidate sets is recognized as the *generate-and-test* approach to inspection planning. The survey considers objects containing only polyhedral features and the authors state a clear need for the development of planning algorithms capable of handling complex surfaces. They are inclined towards the *synthesis approach*, stating that the viewpoint candidate approach might have the following drawbacks: computational costs of the combinatorial search over finely tessellated (spherical) parameter spaces, use of a tessellated sphere for viewpoint visibility evaluation, and overall scalability of the approach.

Tarbox and Gottschlich [77], too, use a densely sampled viewing sphere, explicitly stating they have no apriori preference on viewpoint placement. The camera viewing distance d is fixed to an arbitrary number and the viewpoints are uniformly distributed over the sphere's surface. It is assumed that the visible parts of the object are always within the acceptable depth of field. The main incentive for using dense sampling is to increase the likelihood of the discovery and coverage of regions which are difficult to sense by introducing redundancy. For meshes containing $1\,000$ - $2\,500$ vertices, they produce a set of $15\,000$ viewpoint candidates which is then used for the evaluation of the proposed optimization algorithms. They stress discrete surface representation as a major drawback of the sensor planning algorithms because the allowed shapes of the surfaces are restricted.

Sheng et al. [70, 71] focus on inspecting objects consisting of surfaces with prominent differences in orientation or having low degree of waviness. They take the *flat patch growing* approach by first splitting the surface into sub-patches, calculating the bounding box of each sub-patch and finally placing the viewpoint candidate at a fixed distance along the normal of the largest bounding box side.

Prieto et al. [65] discretize a 3D model described by NURBS into voxels, split the object into sub surfaces representing simple patches, and, for each patch, find a viewpoint which can be used on the surface in a sweeping manner (also known as view swaths). While such an approach works well on simple objects, it struggles when applied on free form or highly complex surfaces.

Scott [69] suggests an alternative model-based approach using geometric information of the object. A viewpoint candidate is created for each vertex of the mesh, which is stated to be the optimal point for surface description. Due to the high resolution and geometric complexity of some discrete models, the first step is to perform a curvature sensitive resampling. After an additional decimation step, the final viewpoint candidate set is created. A good decimation level is heuristically determined to be 32 times lower than the target model. The method achieves good results for a set covering approach (e.g., a model of ca. $36\,000$ vertices is resampled to 702 vertices and decimated to 72 viewpoint candidates). The introduced geometry-based exploration principles provide the foundation for the feature driven approach presented in this chapter.

In his work on underwater inspection path planning, Englot [18] creates a viewpoint candidate set using random sampling of the mesh primitives, where each primitive is used as a pivot point for a viewpoint. The viewpoints are generated until each facet of the model has reached an arbitrary level of redundancy (number of different viewpoint candidates from which a facet can be observed), resulting in ca. $2\,500$ and $1\,300$ viewpoint candidates from approximately $131\,000$ and $107\,000$ vertex meshes respectively.

Gronle and Osten [28] agree with [69] on generating one viewpoint candidate per mesh vertex. Instead of sampling down the triangulation, they reduce the cardinality of their viewpoint candidate sets using an octree. Each leaf contains a group of neighboring candidates from which random viewpoints get selected while others with similar orientation get deleted.

Jing [38] employs a mesh dilation by a maximum depth field within which a camera can operate. Such an approach resembles the tesselated sphere, but with higher correlation to the model geometry. The viewpoint candidates are obtained by sampling the dilated surface until a predefined number of viewpoint candidates is reached. Their orientation is calculated based on the distance d to the original object primitives, where each primitive has an attraction force of $1/d^2$.

Beside discrete solutions, recent work by Mavrinac et al. [48] and Mohammadikaji et al. [50] shows a rise in continuous optimization, which does not require the generation of a viewpoint candidate set. It is a possible alternative to inspection planning and enables the discovery of a truly optimal solution at the cost of higher computational requirements. Due to the different nature of the general approach, an in-depth comparison is beyond the scope of this thesis.

The viewpoint candidate set generation itself has not received much attention as a solemn research focus so far. While it is present in the available literature, various approaches and their impact to the selected final viewpoints are not discussed. Researchers mostly approach it in a way to obtain any kind of reasonable search space before proceeding onto the optimization which is the focus of their work. The obtained viewpoint candidate set is assumed to be dense enough to contain the optimal viewpoints and is thus deemed adequate. While such approaches offer a rather clean shortcut to tackling the optimization problem, they also inherently impose an optimization handicap due to the fact that viewpoint candidates are mostly not generated based on geometric features of the model. The object space exploration approaches used so far are reasonable and good in terms of implementation simplicity. However, they also do not revise different possibilities or modifications which could produce both more meaningful and application-specific results, as well as reduce cardinality of the generated viewpoint candidate set. Existing solutions have not considered uneven object space sampling by concentrating the viewpoint candidates in areas which might appear challenging during the inspection based on various criteria. Such a modification is crucial due to the fact that the optimization problem at hand is a set covering problem and thus NP-hard. As such, it requires a combinatorial approach to solve it. This work aims to decrease the combinatorial complexity challenge by providing an adaptive method for feature driven placement of viewpoint candidates. The results are meaningful and application-specific viewpoint sets of suitable size for the subsequent optimization process.

6.4 Method

The novel strategy, presented in this section, finds viewpoint candidates by subdividing a continuous model of parametric surfaces into smaller segments, leading to an adaptive and feature-based sampling of the object.

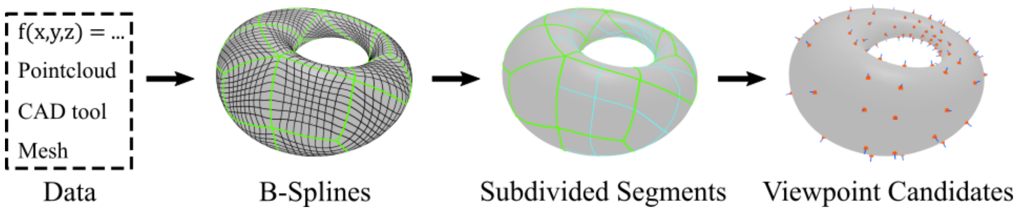


Figure 6.2: *Generating viewpoint candidates. Discrete data is approximated by a B-spline model. Surfaces are subdivided according to a user-defined geometric feature functional (e.g., curvature). Viewpoint candidate pivot points are placed at the centers of each subdivided segment. The cameras are positioned on the line defined by the normal vectors of the surface.*

Here, a *viewpoint candidate* is defined to be a pair of 3D vectors, one of them describing the camera position in world coordinates, the other being the view direction of the camera.

The approach uses a parametric surface representation, in particular it is designed for (but not limited to) objects described by B-spline surfaces. While those surfaces are often an intermediate product of the virtual design process, they are rarely available to inspection system engineers. Hence, it is assumed that only a triangulated surface mesh of the object of interest is given. This can either be a digitally designed model or a result of a 3D reconstruction of a physical object. Due to the use of lasers, 3D scanning tools produce output in the form of point clouds; a conversion to a triangle mesh can easily be achieved by a variety of freely available tools and algorithms, e.g., [9]. If the model is given only in such a discrete format, the first step is to compute a network of continuous B-spline surfaces approximating the original data.

Using B-splines makes the subsequent steps fully independent of the resolution of the input mesh and supports the computation of analytically describable measurements on the surfaces without undesirable discretization effects. Given a model, the user needs to specify a so-called *feature functional* E that measures the relevance of a surface region with respect to certain application-specific features. It should be designed in such a way that it has high values for regions that need to be densely covered by viewpoint candidates and low values in less relevant regions. Let S be a parametric surface, parameterized over a rectangular region $\Omega_0 = [u_{\text{min}}, u_{\text{max}}] \times [v_{\text{min}}, v_{\text{max}}] \subset \mathbb{R}^2$. A point on the surface is denoted by $S(u, v) \in \mathbb{R}^3$ and the notation $S(\Omega) = \{S(u, v) | (u, v) \in \Omega\}$ is used to describe the segment of the surface corresponding to the region $\Omega \subseteq \Omega_0$. The value of the feature functional for the entire surface is denoted by $E(S)$ and the value of the segment corresponding to a parameter region Ω by $E(S, \Omega)$.

In order for the algorithm to converge, the values of the $E(S, \Omega)$ need to converge to zero as Ω converges to a point. To obtain a stable subdivision behavior with respect to the chosen termination threshold, the value of the feature functional computed for a part of a region needs to be smaller than the value computed for the region itself, i.e., for two parameter regions $\Omega, \Omega' \subseteq \Omega_0$, it is required that $E(S, \Omega') < E(S, \Omega)$ if $\Omega' \subset \Omega$.

For functionals that are defined as integrals over some function f of the form

$$E(S, \Omega) = \iint_\Omega f(S, u, v) \mathrm{d}u \mathrm{d}v,$$

it follows that

$$E(S, \Omega) = E(S, \Omega') + E(S, \Omega'')$$

when $\Omega' \cup \Omega'' = \Omega$ and $\Omega' \cap \Omega'' = \emptyset$. In this case, E is said to be *additive*, which means that partitioning a surface segment leads to a partitioning of the evaluated values. This leads to a monotonic decrease when the surface is subdivided. When the function f has a regular distribution of values, the value of E will decrease approximately exponentially with respect to the depth of the subdivision.

The chosen measurement is used to steer the placement and distribution of viewpoint candidates by recursively subdividing regions with high values until a predefined threshold is met. Once the surface network is subdivided, a viewpoint candidate is computed for each segment of the subdivided model.

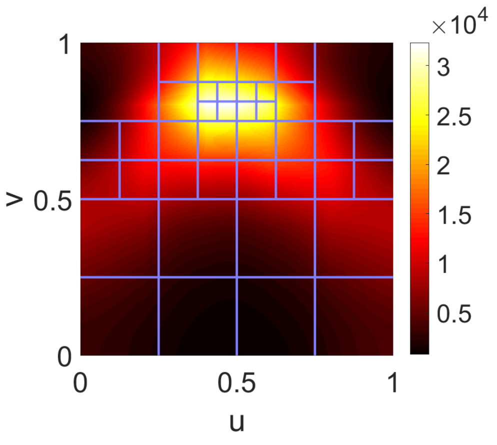
Assuming that the initial input is a triangle mesh model, the individual steps can be summarized as follows:

1. Generate a representation of the object with a finite number of B-spline surfaces.
2. Subdivide the B-spline surfaces with respect to a user-defined functional $E(S, \Omega)$ and an associated threshold value t.
3. Place viewpoint candidates in $\mathbb{R}^3$ based on the subdivision of the model.

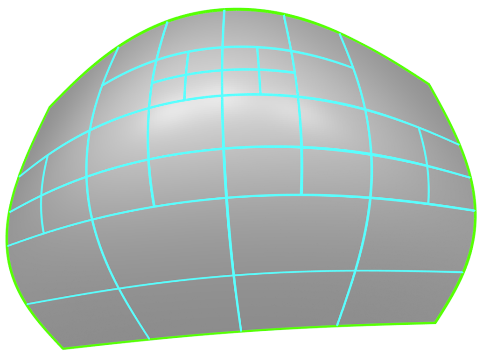
Figure 6.2 illustrates this process. The generation of the B-spline surface model is described in detail in Chapter 2. Subsection 6.4.1 covers the subdivision of such a model. The computation of the viewpoint candidates is explained in Subsection 6.4.2. Subsection 6.4.3 introduces a variety of possible and practical feature functionals. A method to select optimal viewpoints from the list of candidates is discussed in Subsection 6.4.4.

6.4.1 Subdivision of the Model

This section describes the feature-based subdivision approach for a given B-spline model consisting of n_s surfaces $S_i : \Omega_0 \to \mathbb{R}^3, i = 1, ..., n_s$ defined over a rectangular parameter region $\Omega_0 \subset \mathbb{R}^2$. Let $E(S, \Omega)$ be the given functional that measures how prominent a feature of interest occurs within the segment of S restricted to $\Omega \subset \Omega_0$.



(a) *Subdivision in parameter space*



(b) *B-spline surface in* $\mathbb{R}^3$

Figure 6.3: *Subdivision of a single B-spline surface using thin-plate energy as feature functional. Plotting the integrand value of Equation (2.10) in the parameter space* $\Omega_0 = [0, 1] \times [0, 1]$ *illustrates how the approach leads to a deeper subdivision in areas with high values (a). (b) shows the surface* $S(\Omega_0)$ *and its subdivision into segments in* $\mathbb{R}^3$.

The goal is to subdivide each surface into segments, such that E, evaluated for any of these segments, is below a given threshold t. Since the presented recursive approach considers each surface individually, it can be applied in parallel to all B-spline surfaces of the model.

The B-spline surfaces are subdivided in parameter space. The subdivision method is inspired by the work of Nouanesengsy et al. concerning analysis-driven refinement [56]. Initially, for each B-spline surface S, the given functional E is evaluated for the entire surface. If its value is larger than t, the parameter domain is subdivided into four equal-size rectangles. The procedure continues recursively for each subregion. Once the value of E evaluated for a segment falls below t, the corresponding parameter region does not get subdivided any further. Algorithm 1 summarizes this procedure. Figure 6.3 demonstrates the subdivision of a single surface based on thin-plate energy.

Choosing the Subdivision Threshold

Finding a good subdivision threshold value depends on the understanding of the inspection requirements and intuition behind the used subdivision criteria. For example, a measurement that computes a maximum incidence angle for a segment is easily un-

Algorithm 1 Recursive Subdivision

```
function SUBDIVIDESURFACE($S, \Omega, E, t$)
    if $E(S, \Omega) > t$ then
        $(\Omega_1, \Omega_2, \Omega_3, \Omega_4) \leftarrow$ SPLIT($\Omega$)
        return $\bigcup_{i=1}^{4}$ SUBDIVIDESURFACE($S, \Omega_i, E, t$)
    else
        return $\{S(\Omega)\}$
    end if
end function

function SPLIT($\Omega = [u_0, u_1] \times [v_0, v_1]$)
    $u_c \leftarrow \frac{u_0+u_1}{2}$
    $v_c \leftarrow \frac{v_0+v_1}{2}$
    $\Omega_1 \leftarrow [u_0, u_c] \times [v_0, v_c]$
    $\Omega_2 \leftarrow [u_c, u_1] \times [v_0, v_c]$
    $\Omega_3 \leftarrow [u_0, u_c] \times [v_c, v_1]$
    $\Omega_4 \leftarrow [u_c, u_1] \times [v_c, v_1]$
    return $(\Omega_1, \Omega_2, \Omega_3, \Omega_4)$
end function
```

derstood, while an integral of principal curvatures requires the user to have specialized training in geometry in order to interpret the resulting values.

Defining the subdivision threshold manually requires technical knowledge about the expected values for differently shaped surfaces. This can be mitigated by evaluating the feature functionals for all individual surfaces and calculating the average value

$$t_{\text{avg}} = \frac{1}{n_s} \sum_{i=1}^{n_s} E(S). \tag{6.1}$$

The user defines a percentage of the average to control the amount of viewpoint candidates. For example, choosing $t = 0.5 t_{\text{avg}}$ leads to a subdivision of the surface network until the values of each patch are below half of the overall average feature value. Alternatively, if the feature functional has the additivity property, it is possible to express the threshold as a percentage of the total value over the entire object.

A suitable subdivision threshold can also be chosen interactively in real time. In this scenario, it is necessary to pre-compute the subdivision for a small threshold t_1, e.g., a lower bound for the range of considered thresholds. Depending on the complexity of the feature functional, computing this fine subdivision can be time-consuming. However, when all intermediate steps and evaluated measurements are stored in a tree structure, the tree can be cut back in real time for any threshold $t_2 > t_1$. This makes it possible for the system engineer to adjust a slider in a graphical user interface and interact with a visual representation of the subdivided model to determine the best threshold for a specific application scenario.

Modifications

If the used feature functional is additive, it allows for a potential speed-up of the process by starting with a fine subdivision of all surfaces. The subdivision tree is then computed backwards, i.e., by merging individual surface segments as long as the value of the merged patch stays below the subdivision threshold. The computational speed-up is due to the fact that E only has to be evaluated for the deepest subdivision level. The values of the combined segments can be obtained by adding the values of the individual segments.

Furthermore, the algorithm can be extended to handle multiple functionals describing different features of interest, e.g., both curvature and surface area. For this purpose, the feature functional and threshold parameter must be replaced by a list of functional-threshold pairs. All functionals are evaluated, and when one exceeds its corresponding threshold the evaluated segment is subdivided.

Iterative Subdivision

The goal of the recursive subdivision is to generate a sufficient sampling of the model with respect to the features of interest. Alternatively, a global approach guided by a desired number of viewpoint candidates can be chosen. This approach becomes necessary when external constraints limit the amount of allowed viewpoint candidates.

Let t_n be a given number of maximum viewpoints for the entire model. Instead of individually subdividing each surface until the measurement reaches the subdivision threshold, the feature functional is evaluated for all surface segments and the segment where it is maximal is subdivided. This procedure is repeated until the desired amount of segments is reached. Algorithm 2 summarizes this approach.

Algorithm 2 Iterative Subdivision

```
function SUBDIVIDEMODEL($\{S_1, S_2, ..., S_{n_s}\}, E, t_n$)
    $\mathcal{S} \leftarrow \{S_1, S_2, ..., S_{n_s}\}$
    while $|\mathcal{S}| < t_n$ do
        $(S', \Omega') \leftarrow \underset{S,\Omega : S(\Omega) \in \mathcal{S}}{\text{argmax}} \; E(S)$
        $(\Omega_1, \Omega_2, \Omega_3, \Omega_4) \leftarrow$ SPLIT($\Omega'$)
        $\mathcal{S} \leftarrow \mathcal{S} \setminus S'(\Omega') \cup \{S'(\Omega_1), S'(\Omega_2), S'(\Omega_3), S'(\Omega_4)\}$
    end while
end function
```

6.4.2 Computing the Viewpoint Candidates

After the subdivision phase, one viewpoint candidate is placed per surface segment of the subdivided model. First, a pivot point is chosen, i.e., the point that determines the camera viewing direction. Then, the actual camera position is calculated via a user-defined distance d. This distance is manually determined by a system engineer according to the inspection set-up requirements. That way, the viewpoint candidate generation approach assures compliance with the camera's depth-of-field constraints at generation time.

The approximated center of the segment is used as a pivot point. It can be obtained by evaluating the surface at the center point of its respective parameter domain. The camera position is computed by moving away from the surface in the direction of the surface normal, given by

$$n(u,v) = \frac{\partial_u S(u,v) \times \partial_v S(u,v)}{\|\partial_u S(u,v) \times \partial_v S(u,v)\|}.$$

The position of the camera is defined as

$$C^d_{S,\Omega} = S(u_c, v_c) + n(u_c, v_c) \cdot d, \tag{6.2}$$

where (u_c, v_c) is the center of the parameter domain Ω. For a rectangular region $\Omega = [u_0, u_1] \times [v_0, v_1]$, it is given by $u_c = (u_0 + u_1)/2$ and $v_c = (v_0 + v_1)/2$. The view direction of the camera is the reversed normal vector. Figure 6.4 illustrates this concept.

6.4.3 Feature Functionals

The feature functional E is essential for the distribution of the viewpoint candidates. In order to produce high quality results, tailored to specific applications, it needs to be defined in a way that identifies areas of interest by assigning them high values while at the same time giving low values to areas that do not contain the desired features. It is up to the system engineer to either choose or define a functional applicable to their use case. A good functional should be intuitive and easy to compute.

As discussed in Section 6.4, the values of $E(S, \Omega)$ need to converge towards zero as Ω converges to a single point to ensure termination of the subdivision. The following subsections introduce a number of fundamental feature functionals that satisfy this property. They are suitable candidates for a variety of shapes, as they highlight common geometric features of interest. The modularity of the algorithm allows it to be easily extended with custom functionals as long as they also satisfy the convergence property.

Some of the presented feature functionals are defined as integrals. For implementation purposes, those can be evaluated numerically. Trapezoidal rules have proven to

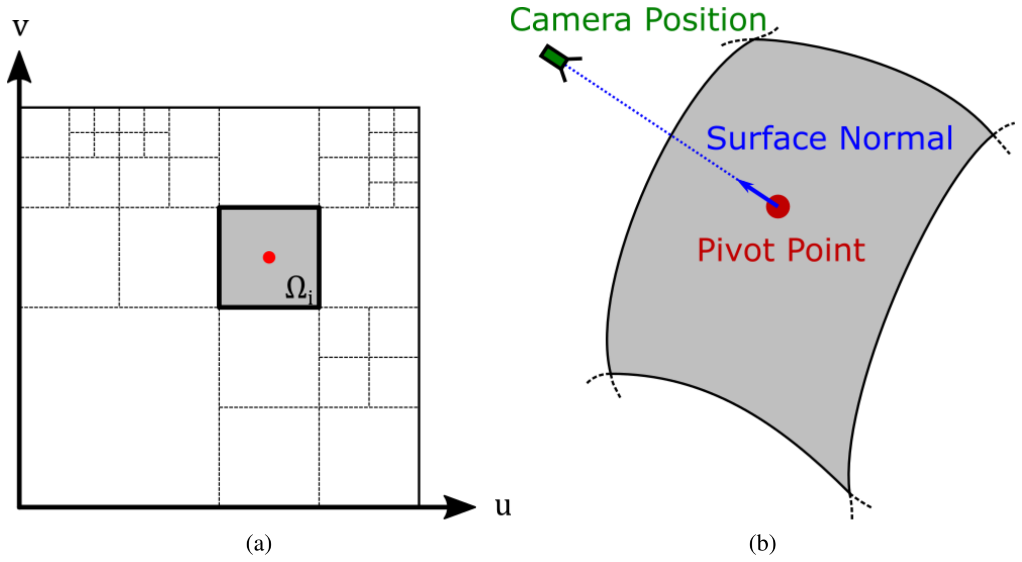


Figure 6.4: *Computing the viewpoint candidate for a surface segment. (a) illustrates a subdivision of a surface in parameter space. The image $S(\Omega_i)$ of the highlighted region is shown in (b). Evaluating the surface at the center of Ω_i yields the pivot point for this segment. The camera position is obtained by moving away in the direction of the surface normal at that point.*

be an adequate method, as for a sufficiently dense sampling of the surface, the error becomes neglectably small [2].

Curvature

In the context of surface inspection, the placement of viewpoint candidates should be focused around highly curved parts of the object. The high surface variation in those regions usually requires an inspection from several directions, while flat regions can be covered with a single or only a few viewpoints.

Commonly used terms to describe curvature are the *principal curvatures* κ_1 and κ_2. At any given point on the surface, these are the normal curvature values in the directions in which the surface has its lowest and highest bending, respectively. Combined, they define the mean curvature $H = (\kappa_1 + \kappa_2)/2$ and the Gaussian curvature $K = \kappa_1\kappa_2$. More detailed information on differential geometry and curvature can be found in literature [12].

Individually, these terms are not suited to separate flat and curved regions, as both can become zero in non-flat regions, i.e., if one of the principal curvatures is zero while the other has a value other than zero (Gaussian curvature) or if $\kappa_1 = -\kappa_2 \neq 0$ (mean curvature). This can be resolved by directly considering the principal curvatures. The surface integral given by

$$E_\kappa(S) = \iint_S \kappa_1^2 + \kappa_2^2 \,\mathrm{d}S \tag{6.3}$$

is a meaningful measurement to describe the total bending of a surface S [30]. It is equal to zero if and only if S is a flat surface. To compute the integral with respect to a parameter region Ω, it can be rewritten as

$$E_\kappa(S, \Omega) = \iint_\Omega (\kappa_1^2(u,v) + \kappa_2^2(u,v)) \left\| \partial_u S(u,v) \times \partial_v S(u,v) \right\| \mathrm{d}u\mathrm{d}v. \tag{6.4}$$

B-splines are parametric surfaces with well-defined first and second derivatives everywhere. Hence, it is possible to analytically evaluate κ_1 and κ_2 at any point and numerically compute the entire integral.

For implementation purposes, it is useful to express the integrand of Equation (6.4) in terms of Gaussian and mean curvature:

$$\kappa_1^2 + \kappa_2^2 = 4H^2 - 2K.$$

Evaluating this term requires less operations, as H and K can be obtained directly from the coefficients of the first and second fundamental forms which are straightforward to compute at any point of a B-spline surface [12].

Thin-plate Energy

Computing the integral of the principle curvatures numerically yields precise and intuitive results at the price of high computational cost. However, it can be sped up immensely by approximating Equation (6.3) using the thin-plate energy, as described in Section 2.3.

This is done by generalizing the thin-plate energy term, see Equation (2.10), to arbitrary parameter regions Ω. The resulting feature functional for a B-spline surface S is defined as

$$\begin{aligned} E_{\mathrm{TP}}(S, \Omega) &= \iint_\Omega \left\| \partial_{uu} S(u,v) \right\|^2 + 2\left\| \partial_{uv} S(u,v) \right\|^2 + \left\| \partial_{vv} S(u,v) \right\|^2 \mathrm{d}u\mathrm{d}v \\ &= \sum_{\sigma \in \{x,y,z\}} b_\sigma^T M_\Omega b_\sigma. \end{aligned} \tag{6.5}$$

Analogously to Equation (2.11), the matrix $M_\Omega \in \mathbb{R}^{n_c^2 \times n_c^2}$ is given by

$$\begin{aligned} M_\Omega = \iint_\Omega & (\partial_{uu} N(u,v))^T (\partial_{uu} N(u,v)) \\ & + 2(\partial_{uv} N(u,v))^T (\partial_{uv} N(u,v)) \\ & + (\partial_{vv} N(u,v))^T (\partial_{vv} N(u,v)) \mathrm{d}u\mathrm{d}v. \end{aligned}$$

After computing M_Ω, evaluating $E_{\text{TP}}(S,\Omega)$ requires the calculation of computationally inexpensive matrix-vector products.

The coefficients of the matrix M_Ω only depend on the used knot vector τ, the order k of the B-spline and the evaluated parameter region Ω. Assuming that all B-spline surfaces in the model have the same order and knot vector, M_Ω only needs to be computed once per subregion $\Omega \subseteq \Omega_0$ that is considered during a subdivision step. The matrices for individual subregions can be precomputed and saved up to a predefined maximum subdivision depth allowing for them to be reused on different models with same order and knot vector.

Maximum Deviation From Average Normal

An alternative approach to obtain a subdivision into mostly flat surface segments is to aim at a low variation of normal vectors within each patch. Finding the maximum angle between normal vectors at any two arbitrary surface points is a non trivial task. A more practical solution is to calculate the average normal of a surface segment and compute the maximum angle between any normal vector on the segment and this average normal. This is a modification to the approach used by Sheng et al.[70] who apply this concept in the discrete case of triangulated surfaces.

In the continuous setting of B-splines, the average normal vector of a surface segment is given by

$$\tilde{n}_{\text{avg}}(S,\Omega) = \frac{\iint_\Omega n(u,v)\,\|\partial_u S(u,v) \times \partial_v S(u,v)\|\,\mathrm{d}u\mathrm{d}v}{\iint_\Omega \|\partial_u S(u,v) \times \partial_v S(u,v)\|\,\mathrm{d}u\mathrm{d}v}$$

with normalization

$$n_{\text{avg}}(S,\Omega) = \frac{\tilde{n}_{\text{avg}}(S,\Omega)}{\|\tilde{n}_{\text{avg}}(S,\Omega)\|}.$$

The functional to measure maximum normal deviation is then defined as

$$E_{\text{ND}}(S,\Omega) = \max_{(u,v)\in\Omega} \cos^{-1}\left(n_{\text{avg}}(S,\Omega)\cdot n(u,v)\right).$$

Maximum Incidence Angle

While this work focuses on geometric properties of the target objects, it can also utilize additional knowledge from material science. Some properties like surface roughness or reflection behavior require a restriction of the angle under which the surface of the object is inspected. This is achieved by introducing a functional that measures the maximum deviation between the surface normal at any point of the segment and a ray that originates from a camera placed according to Equation (6.2) at distance d:

$$E_{\text{IA}}(S,\Omega) = \max_{(u,v)\in\Omega} \cos^{-1}\left(n(u,v)\cdot\frac{C^d_{S,\Omega} - S(u,v)}{\|C^d_{S,\Omega} - S(u,v)\|}\right).$$

More complex constraints like a range between minimum and maximum angle or deviation from an optimal angle can be modeled in a similar way.

Area

Surface area itself can also be used as subdivision criterion. It makes sure that at least one viewpoint candidate gets generated per predefined maximum amount of area. This is especially useful in cases where features of interest cannot be expressed by simple mathematical functions or in a prototyping process where specific feature functionals are not yet determined. Moreover, inspection pipeline characteristics such as a narrow camera frustum might impose limiting factors on the amount of the surface that can be processed from a single viewpoint. A subdivision by area will always ensure a certain sampling resolution of the surface.

The surface area is given by

$$E_A(S) = \iint_S 1 \mathrm{d}S. \tag{6.6}$$

To obtain the feature functional for arbitrary segments, (6.6) is rewritten as

$$E_A(S, \Omega) = \iint_\Omega \|\partial_u S(u,v) \times \partial_v S(u,v)\| \,\mathrm{d}u\mathrm{d}v.$$

For complex objects, this functional can be used in combination with other geometric feature functionals like curvature to control sampling rate sparsity in flat regions.

Region of Interest

Engineers and inspectors working with production lines can usually predict where defects will occur based on their experience. Even if the inspection is already automated, this kind of knowledge can be obtained by analyzing the distribution of previously detected defects.

If this information is available, the application scientist can define a region of interest (ROI), either directly by using a brush tool on the surface or freely in 3D world coordinates (e.g., by defining a bounding sphere or bounding box) to highlight regions of the object at which the viewpoint candidate distribution needs to be more intense. With

$$f_{\text{ROI}}(x,y,z) = \begin{cases} 1, & \text{if } (x,y,z) \in \text{ROI} \\ 0, & \text{otherwise} \end{cases},$$

the ROI-functional is defined as the surface area that is inside the region of interest:

$$E_{\text{ROI}}(S, \Omega) = \iint_\Omega f_{\text{ROI}}\left(S(u,v)\right) \|\partial_u S(u,v) \times \partial_v S(u,v)\| \,\mathrm{d}u\mathrm{d}v$$

Besides highlighting regions where errors are expected, this functional can also be used to focus on critical parts of the object in which surface defects have a higher significance.

6.4.4 Selecting Final Viewpoints

After a set of viewpoint candidates is computed, a list of optimal viewpoints needs to be selected with the requirement of keeping the number as low as possible while covering the desired parts of the object. For that purpose, the initial mesh model of the object is used to evaluate the viewpoints. Let F be a set containing the n_F surface primitives (triangles) of the mesh and let a set of viewpoint candidates $V = \{v_1, ..., v_{n_{\text{vpc}}}\}$ be given. The goal is to find a set of optimal viewpoints $O \subseteq V$ which sufficiently covers the object. To represent the coverage relation between viewpoints and the faces of the mesh, a visibility matrix $A \in \{0,1\}^{n_F \times n_{\text{vpc}}}$ is constructed, with coefficients $a_{ij} = 1$ if and only if viewpoint v_j covers the i-th triangle $f_i \in F$.

Since all sets are discrete and finite, the task is equivalent to the Set Coverage Problem [73] and can be written as

$$\begin{aligned} &\min \sum_{i=1}^{n_{\text{vpc}}} c_i x_i, \quad c_i = \frac{1}{\sum_{i=1}^{n_F} a_{ij}} \\ &\text{s.t. } Ax \geq b^{(F)}. \end{aligned} \tag{6.7}$$

The binary variables x_i indicate whether the i-th viewpoint candidate is selected, i.e., $O = \{v_i \in V | x_i = 1\}$. The vector $b^{(F)} \in \{0,1\}^{n_F}$ controls which surface primitives need to be covered by the inspection system. Using $b^{(F)} = (1, ..., 1)^T$ is equivalent to requiring full coverage.

The next best view approach is frequently used to solve this problem [28, 38, 69]. To evaluate the viewpoint candidate sets reasonably, the same approach is used here. That way, comparable results are obtained and the strengths of the presented approach can be discussed. The visibility matrix is built using ray tracing to create the correlation between the primitives and the viewpoints which can observe them. The starting viewpoint is chosen by finding the triangle covered by the least amount of viewpoints. From the set of viewpoints that observe this triangle, the viewpoint which observes the largest part of the object is chosen. Further viewpoints are chosen iteratively by always choosing the viewpoint that observes the most uncovered triangles. The process is repeated until all the primitives are covered or there are no more viewpoints available.

6.5 Results

The presented method is applied to models of varying complexity to demonstrate the viability of the approach as well as to highlight the differences between the feature functionals introduced in Subsection 6.4.3. To enable a meaningful comparison, the subdivision thresholds are chosen to generate approximately the same magnitude of viewpoint candidates.

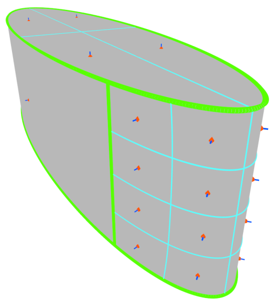
Each result picture shows the subdivided models including glyphs to mark pivot points and normal directions. When using integral-based functionals, the used subdivision threshold t is given in relation to the average threshold t_{avg} as defined by Equation (6.1). For the measurements that are evaluating angles, the threshold is provided as fixed value in degrees. The number of viewpoint candidates generated by the recursive subdivision is denoted by n_{vpc}. All shown B-spline models are of order $k = 3$ and use uniform knot vectors with multiple ends, see Equation (2.1).

Figure 6.5 shows the results from applying the recursive subdivision to a cylinder geometry (stretched in one dimension). Top and bottom are each realized by a single B-spline surface while the mantle is modeled by four surfaces. Due to the stretching in x-direction, two of these surfaces are relatively flat, while the other two surfaces are highly bent. The respective sizes of the object in x, y, and z directions are 80, 20, and 20 units in world coordinates.

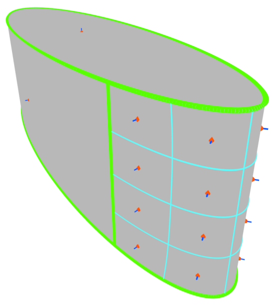
The spring model shown in Figure 6.6 is geometrically more difficult. Its minimum bounding box has length and width equal to 15, and height equal to 20 world units. Like the cylinder, it consists of one individual circular surface at the top and bottom, respectively. Instead of a smooth mantle, it consists of more complex surfaces with high variation in curvature along its side. This part is divided into four slices along the circumference, each being split into three individual B-spline surfaces along its height. As long as the goal of the viewpoint placement is only unconstrained coverage, this model can still easily be processed by a human inspector, while the curvature behavior along the mantle introduces difficulties to most automatic methods. However, once constraints, e.g., coverage under restricted angles, are factored in, the complexity of this task increases drastically for human operators, necessitating an automated solution.

In Figure 6.7, the recursive subdivision is demonstrated on a geometrically and topologically more challenging object. The data itself had been obtained from a 3D volumetric image and the B-spline representation with 452 surfaces was computed fully automatically using the surface reconstruction approach discussed in Chapter 2. Dimensions of the object are 72 world units in x, y, and z direction respectively. Setting up an inspection system for this model manually would take considerable effort. In this example the boundaries of the individual B-spline surfaces are not aligned with any features of the object. Hence, some surfaces can cover both curved and flat parts. The curvature-based criteria are able to handle this scenario especially well, leading to a dense placement of viewpoint candidates around highly bent areas.

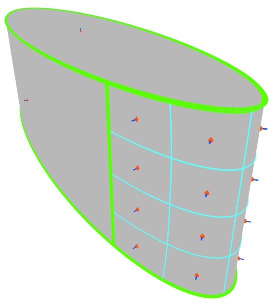
In some cases, several types of geometric features are relevant and need to be considered for the placement of viewpoint candidates. As discussed in Subsection 6.4.1, the recursive subdivision approach can evaluate several functionals and compare them with their respective thresholds at each step. Figure 6.8 demonstrates this modification



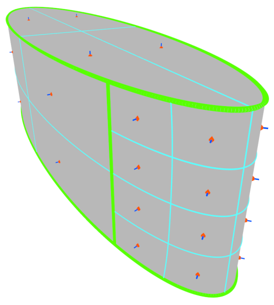
(a) *Thin-plate energy*
$t = 0.4t_{\text{avg}}, n_{\text{vpc}} = 42$



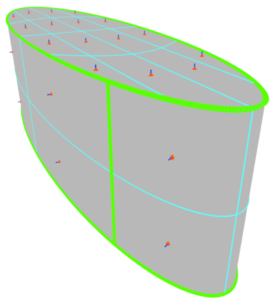
(b) *Curvature*
$t = 0.4t_{\text{avg}}, n_{\text{vpc}} = 36$



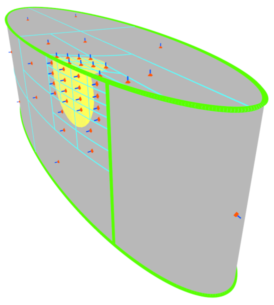
(c) *Max. normal deviation*
$t = 45°, n_{\text{vpc}} = 36$



(d) *Max. incidence angle*
$t = 60°, d = 20, n_{\text{vpc}} = 48$

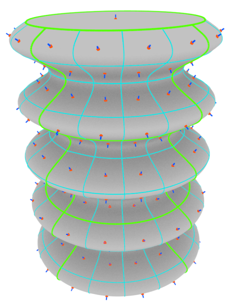


(e) *Area*
$t = 0.3t_{\text{avg}}, n_{\text{vpc}} = 48$

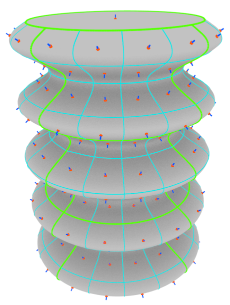


(f) *Region of interest*
$t = 0.5t_{\text{avg}}, n_{\text{vpc}} = 45$

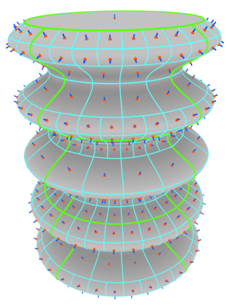
Figure 6.5: *Application of the recursive subdivision to a cylinder model. The curvature-based functionals (a), (b), (c) yield a higher sampling where surface bending is high. However, using the thin-plate energy also leads to a subdivision of the top and bottom surfaces. Likewise, the incidence angle (d) produces the desired sampling of the curved areas while also subdividing flat regions to provide sufficient coverage under the given angle constraints. The subdivision by area (e) provides a good overall coverage while the region of interest (f) places the viewpoint candidates primarily on the highlighted region.*



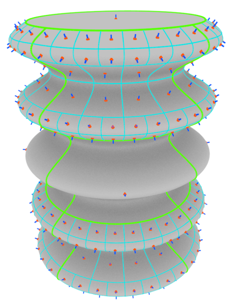
(a) *Thin-plate energy*
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 194$



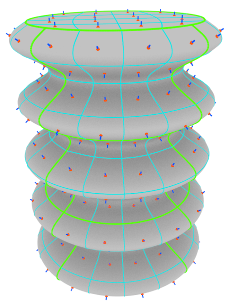
(b) *Curvature*
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 194$



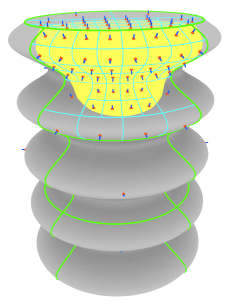
(c) *Max. normal deviation*
$t = 70^\circ, n_{\text{vpc}} = 482$



(d) *Max. incidence angle*
$t = 85^\circ, d = 20, n_{\text{vpc}} = 326$

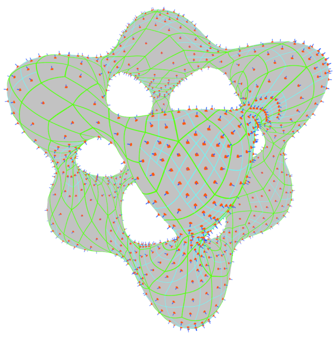


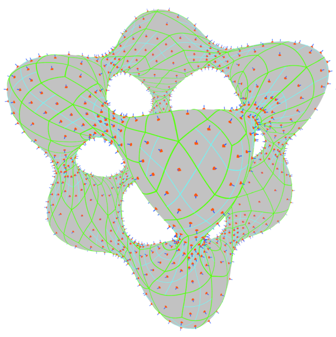
(e) *Area*
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 224$

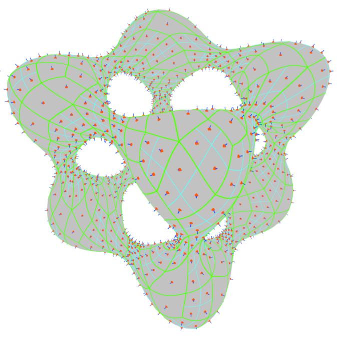


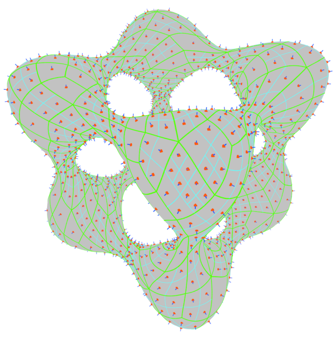
(f) *Region of interest*
$t = 0.5t_{\text{avg}}, n_{\text{vpc}} = 86$

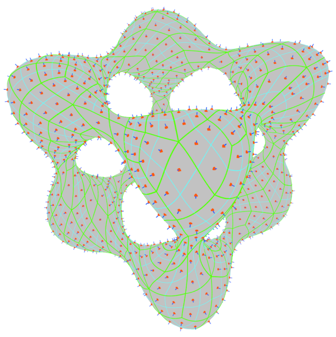
Figure 6.6: *Application of the recursive subdivision approach to the spring model. The subdivisions for thin-plate energy (a) and curvature (b) are identical for equivalent thresholds. However, the normal deviation (c) shows less stable results. While the subdivision for the given threshold is finer than for the other curvature-based measurements, choosing a slightly higher threshold would cause no subdivision at all. Using the incidence angle (d) is also less stable as the middle part of the object does not get subdivided at all. As in the case of the cylinder model, the area measurement (e) leads to a evenly distributed sampling while the region of interest (f) subdivides only the highlighted part of the model.*


(a) *Thin-plate energy*
$t = 0.75 t_{\text{avg}}, n_{\text{vpc}} = 1\,400$


(b) *Curvature*
$t = 0.75 t_{\text{avg}}, n_{\text{vpc}} = 1\,325$


(c) *Max. normal deviation*
$t = 30^\circ, n_{\text{vpc}} = 1\,571$


(d) *Max. incidence angle*
$t = 35^\circ, d = 130, n_{\text{vpc}} = 1\,379$


(e) *Area*
$t = 0.75 t_{\text{avg}}, n_{\text{vpc}} = 1\,406$

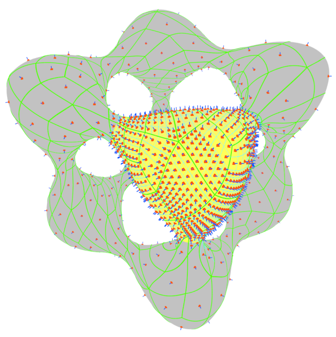
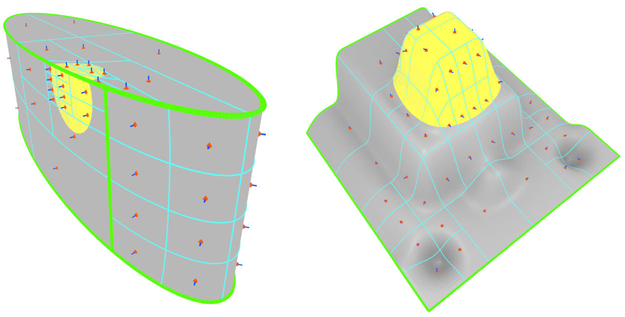

(f) *Region of interest*
$t = t_{\text{avg}}, n_{\text{vpc}} = 1\,361$

Figure 6.7: *When applying the recursive subdivision to more complex models, differences between used functionals become more distinctively visible. Using the thin-plate energy (a) leads to a higher refinement on curved regions, but also on some of the flat regions as they still contain planar distortions. Measuring curvature directly (b) provides a more accurate placement of viewpoint candidates around the thin and highly curved parts of the object, while still moderately refining the outer surfaces with medium bending. Similar results are obtained using the normal deviation (c) and incidence angle (d) functionals. A regular sampling is obtained when considering surface area (e). The region of interest functional (f) leads to a high intensity of viewpoint candidates in the highlighted part.*



(a)
$t^{(TP)} = 0.4t_{\text{avg}}^{(TP)}$
$t^{(ROI)} = 0.8t_{\text{avg}}^{(ROI)}$
$n_{vpc} = 66$

(b)
$t^{(TP)} = 0.05t_{\text{avg}}^{(TP)}$
$t^{(ROI)} = 0.1t_{\text{avg}}^{(ROI)}$
$n_{vpc} = 55$

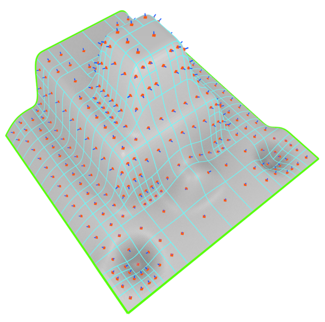
Figure 6.8: *Combining feature functionals for the recursive subdivision. Both models are subdivided using both thin-plate energy (TP) and region of interest functional (ROI), ensuring an adequate overall coverage based on surface bending while also focusing on a highlighted region.*

on selected examples.

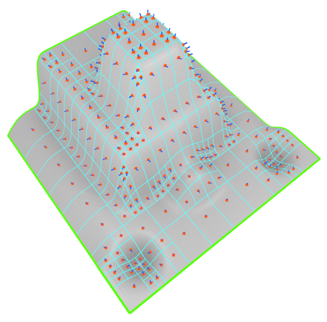
Figure 6.9 shows results of the iterative subdivision (Algorithm 2) applied to a single B-spline surface. This model is 19 world units wide and long while having a height of 11.8 units. It combines a variety of curvature situations within a single surface. In contrast to the recursive approach that focuses on coverage of the object with respect to given objectives, the iterative approach gives the user direct control over the number of viewpoint candidates. Furthermore, it allows for an intuitive comparison of the used feature functionals, as similarities and differences can easily be spotted in the visualizations.

6.5.1 Viewpoint Set Optimization

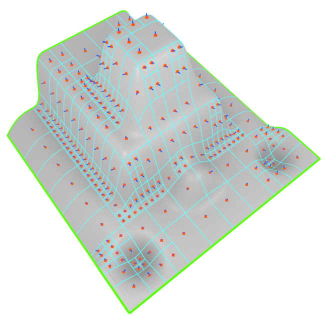
This section briefly presents the result of the subsequent optimization step where a small number of viewpoints is chosen to solve the set covering problem (6.7). The viewpoint candidate sets obtained by applying the recursive subdivision with different feature functionals are processed with the next best view approach as explained in Subsection 6.4.4. The results are given in Table 6.1. While the time for the viewpoint candidate generation is only dependent on the properties of the B-splines, the selection step utilizes the original mesh for the raytracing and verification. Hence, the times depend on the number of triangles as well as the number of viewpoint candidates. The machine used to compute these results is equipped with an Intel(R) Core(TM) i7-4790



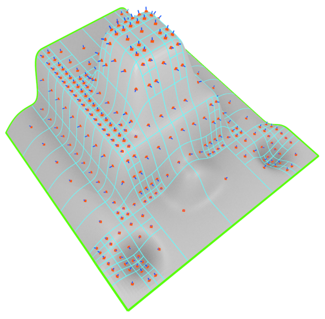
(a) *Thin-plate energy*



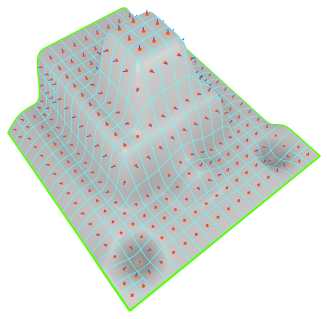
(b) *Curvature*



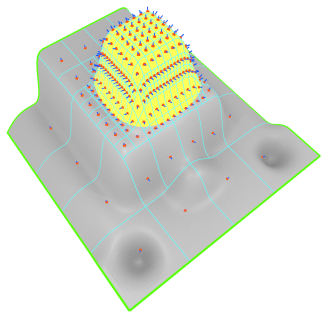
(c) *Max. normal deviation*



(d) *Max. incidence angle*



(e) *Area*



(f) *Region of Interest*

Figure 6.9: *Applying the iterative subdivision constrained by the number of total viewpoints highlights the differences between the presented feature functionals. The number of iterations is limited to* 101 *leading to* 304 *viewpoint candidates in total. The curvature-based measurements (a), (b), and (c), as well as the incidence angle functional (d) focus the subdivision primarily on curved regions. Small differences can be observed on those areas with only moderate bending as the individual functionals prioritize them differently. The area measurement (e) provides a regular, feature independent sampling. Using the region of interest (f), subdivisions occur only in segments that contain parts of the highlighted region.*

CPU @ 3.60GHz and uses 16 gigabyte of DDR3 RAM.

6.5.2 Feature Functional Evaluation

As the presented results demonstrate, a good overall functional that provides satisfying results for all types of geometries is difficult to define. However, the fundamental functionals used here have proven to effectively focus the viewpoint candidate placement around the desired features. When a system engineer has a good idea of what characteristics the viewpoint candidates should be focused on, the recursive subdivision will provide the desired result.

The thin-plate energy provides good results for most situations. It is also the fastest of the presented functionals and can be evaluated in real time, even for larger objects. As a consequence it allows the operator to find a good subdivision threshold interactively, e.g., by adjusting a slider and observing the changes in the visualization. Using the thin-plate energy, however, can also cause subdivisions in some flat areas. While the subsequent viewpoint optimization for inspection purposes is an NP-hard problem, a few additional viewpoints are usually not an issue. This effect is further analyzed in Subsection 6.5.3.

Using curvature as subdivision criterion produces dense sampling of highly curved regions without subdividing flat areas at all. Because it requires a high amount of elementary computations, it cannot be evaluated in real time. The bending at any point on the surface segment contributes to the evaluated value leading to slight averaging effects, i.e., singular occurrences of a high surface bending will not necessarily trigger a subdivision of a mostly flat surface piece if the threshold is chosen high enough.

When this effect is not desired, the deviation from the average normal can provide better results. This functional is more susceptible to local spikes, leading to very high subdivision levels around them. However, the subdivision threshold should be modified carefully, as minor changes can drastically increase or reduce the amount of generated viewpoint candidates. Subsection 6.5.5 analyzes this behavior in more detail.

The maximum incidence angle functional is more practically relevant, as it ensures that the entire segment assigned to a viewpoint can be inspected under a certain angle. However, as Figure 6.6d shows, the intermediate viewpoint placement evaluation during individual steps can lead to strong fluctuations of the subdivision depth.

Using subdivision by surface area provides regular sampling. It ensures that no segment is larger than the given threshold. It does not highlight any particular properties of the object on its own. However, it is a good supplement for any other criteria that focus on high subdivisions around specific features, e.g., the region of interest.

In fact, the region of interest functional is the most interactive of the presented subdivision criteria. It allows users to highlight important features manually without the need to formally define and implement a measurement. On more complex models, the features of interest and the regions they are occurring in might be detected using a

Viewpoint Candidate Generation				Viewpoint Set Optimization		
Feature Functional	Threshold	SD [s]	n_{vpc}	n_{opt}	RT [s]	NBV [s]
Thin-plate	$0.4t_{\mathrm{avg}}$	0.001	**42**	**16**	49.553	0.522
Curvature	$0.4t_{\mathrm{avg}}$	12.518	**36**	**10**	41.187	0.427
Normal Deviation	$45°$	2.848	**36**	**10**	41.323	0.424
Incidence Angle	$60°$	4.317	**48**	**22**	56.680	0.579
Area	$0.3t_{\mathrm{avg}}$	2.950	**48**	**24**	61.570	1.031

(a) *Cylinder (*$21\,800$ *mesh primitives,* 6 *surfaces,* 16×16 *control points).*

Viewpoint Candidate Generation				Viewpoint Set Optimization		
Feature Functional	Threshold	SD [s]	n_{vpc}	n_{opt}	RT [s]	NBV [s]
Thin-plate	$0.2t_{\mathrm{avg}}$	0.005	**194**	**13**	267.68	189.91
Curvature	$0.2t_{\mathrm{avg}}$	79.921	**194**	**13**	269.43	181.01
Normal Deviation	$70°$	20.835	**482**	**9**	647.22	610.15
Incidence Angle	$85°$	25.189	**326**	**9**	408.50	359.29
Area	$0.2t_{\mathrm{avg}}$	17.253	**224**	**13**	312.95	205.01

(b) *Spring (*$51\,800$ *mesh primitives,* 14 *surfaces,* 20×20 *control points).*

Viewpoint Candidate Generation				Viewpoint Set Optimization		
Feature Functional	Threshold	SD [s]	n_{vpc}	n_{opt}	RT [s]	NBV [s]
Thin-plate	$0.75t_{\mathrm{avg}}$	0.008	**1 400**	**36**	2 614.64	249.02
Curvature	$0.75t_{\mathrm{avg}}$	16.696	**1 325**	**32**	2 556.78	270.60
Normal Deviation	$30°$	2.134	**1 571**	**32**	3 103.06	289.86
Incidence Angle	$35°$	3.021	**1 379**	**33**	1 771.99	234.97
Area	$0.75t_{\mathrm{avg}}$	1.883	**1 406**	**38**	2 613.49	268.04

(c) *Tanglecube (*$82\,672$ *mesh primitives,* 452 *surfaces,* 6×6 *control points).*

Table 6.1: *Results of the viewpoint candidate generation based on the recursive subdivision for different feature functionals and set optimization with goal of maximum coverage. The computational time is caused primarily by three key steps of the process which are measured individually: Application of the recursive subdivision algorithm (SD), computation of the visibility matrix via raytracing (RT), and performing the next best view algorithm (NBV). The number of viewpoint candidates is given by* n_{vpc} *while* n_{opt} *denotes the number of viewpoints chosen during the selection step.*

Functional	Intuition of Values	Computational Cost	Meeting Design Objectives	Additivity
Thin-plate Energy	$--$	$+++$	$+$	yes
Curvature	$-$	$--$	$++$	yes
Normal Deviation	$++$	$-$	$++$	no
Incidence Angle	$++$	$-$	$++$	no
Area	$+$	$-$	$++$	yes
Region of Interest	$+$	$-$	$+++$	yes

Table 6.2: *Qualitative overview over different aspects of the presented feature functionals. The intuition of values describes how easy it is for a human operator to understand the values obtained by evaluating the given functional. Computational cost is important when the system has to be set up interactively or close to real time. How well a functional meets the design objective determines to which extend the results and the user's expectations will align. Fulfilling the additivity property leads to a stable behavior with respect to modifications of the subdivision threshold.*

domain specialized tool.

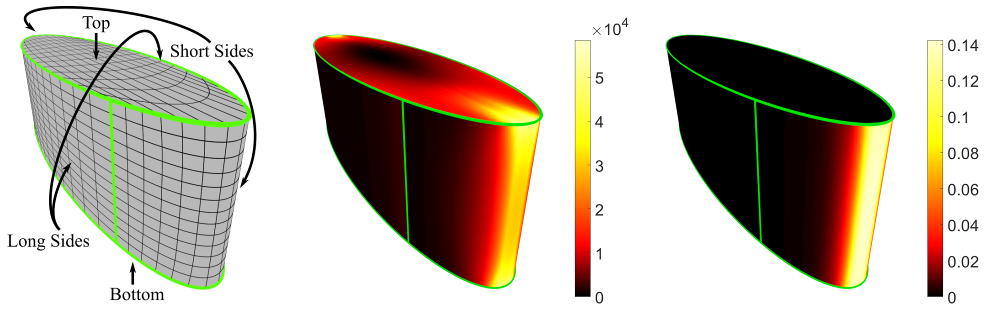
In general, all of the curvature related functionals lead to a good overall viewpoint candidate placement and ensure that the object is covered from all angles with a relatively low amount of viewpoints. They automatically summarize the object's geometric complexity into meaningful values freeing the user of the need to specify complex properties manually. The remaining measurements are tailored for specific situations and do not necessarily aim at full coverage of the object in highly bent areas. This can be compensated by combining them with one of the curvature-based criteria.

Table 6.2 provides a summarized overview over the strengths and weaknesses of the discussed criteria. The following subsections will further discuss individual aspects.

6.5.3 Curvature Computation Approach

While the thin-plate energy is a widely used functional in variational design to approximate surface bending, it is important to keep in mind that it has been motivated by the physics of deformations. Therefore, it does not accurately model the actual geometric properties of the object. A simple example can be constructed by considering a parametric surface S° with rectangular parameter region $\Omega^\circ = [u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}]$ modeling a flat disk. It is clear that $\kappa_1 = \kappa_2 = 0$ everywhere which means that there is no geometric curvature. Hence, $E_\kappa(S^\circ, \Omega^\circ) = 0$. However, the mapping from a rectangular region to a circular object has to include planar distortions, i.e., second derivatives do not vanish everywhere, implying that $E_{\text{TP}}(S^\circ, \Omega^\circ) \neq 0$.

Figure 6.10b demonstrates this effect. It shows the symmetrical cylinder model consisting of 6 individual B-spline surfaces with a colorplot of the thin-plate energy



(a) *Model with isolines* (b) *Thin-plate energy* (c) *Curvature*

Figure 6.10: *Analysis of the stretched cylinder. (a) shows the model with isolines corresponding to the individual knots τ_i of the B-spline surfaces. Planar distortion can be observed on the top surface, caused by forcing a quadrilateral surface into an elliptic shape. As consequence the thin-plate energy (b) does not vanish. (c) shows that the curvature values conform with the expected result.*

FF	Top/Bottom (2×)	Long Sides (2×)	Short Sides (2×)
$E_{\text{TP}}(S)$	18.83%	2.95%	28.22%
$E_\kappa(S)$	0%	0.41%	49.59%

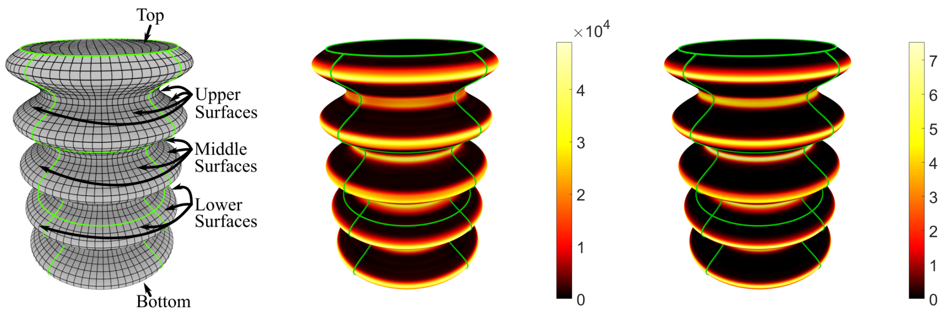
Table 6.3: *Distribution of feature functional (FF) values on the cylinder model shown in Figure 6.10. While the curvature functional shows the desired distribution, using the thin-plate energy will cause a subdivision of the flat surfaces on the top and bottom before considering the slightly curved side surfaces.*

integrand

$$\|\partial_{uu}S(u,v)\|^2 + 2\,\|\partial_{uv}S(u,v)\|^2 + \|\partial_{vv}S(u,v)\|^2$$

on the surface. Since the top and bottom surfaces have ellipsoidal shapes, their second derivatives do not vanish - especially close to the curved boundaries. At the same time, the surfaces on the side with almost no bending are described by a close to uniformly distributed grid of B-spline control points. Hence, there is only little thin-plate energy. Applying one of the presented subdivision approaches leads to a subdivision of the top and bottom before considering these two side surfaces. However, as Table 6.3 shows, the most energy is located at the highly curved surfaces at the ends, which means that the design objective (i.e., subdivision of highly curved regions) is still met.

The exact curvature measured by $\kappa_1^2 + \kappa_2^2$ is shown in Figure 6.10c. Like the thin-plate energy, it has a low value on the long side surfaces and a high value on the strongly curved ends. However, it vanishes on both the top and the bottom, which is in alignment with the expected result when aiming for a curvature-based subdivision.



(a) *Model with isolines* (b) *Thin-plate energy* (c) *Curvature*

Figure 6.11: *Analysis of the spring model. The isolines (a) of the B-splines indicate less distortion in planar regions than for the cylinder in Figure 6.10. Although this still causes low nonzero values for the thin-plate energy there, the color plot (b) shows that they can be neglected. The distribution is almost identical to the curvature shown in (c).*

The cylinder model has a relatively simple curvature situation (at least one principal curvature is zero at any point) causing the differences to be distinctively visible. In fact, the value distributions are more aligned when considering a more complex example.

Figure 6.11 shows the thin-plate energy and curvature functional values for the spring model. They are summarized in Table 6.4, highlighting that there are still planar distortions causing nonzero values of thin-plate energy on the flat top and bottom. However, compared to the high values occurring on the sides, they are almost neglectable. A qualitative comparison of the color plots of the integrands of $E_{\text{TP}}(S)$ (Figure 6.11b) and $E_\kappa(S)$ (Figure 6.11c) on the surface leads to the conclusion that both functionals behave similarly for this model.

For a curvature-based subdivision, these comparisons indicate that using $E_\kappa(S)$ yields more accurate results as it always exactly meets the design objective. On more complex models, the difference becomes less significant. While it can lead to some additional viewpoint candidates in unintended regions, $E_{\text{TP}}(S)$ still prioritizes highly bent regions leading to the indented subdivision there. However, evaluating $E_\kappa(S)$ requires a large number of operations, while $E_{\text{TP}}(S)$ can be computed by simply evaluating the matrix-vector product in Equation (6.5). Hence, at the cost of some accuracy, thin-plate energy can be used in real time, allowing a user to interactively modify the subdivision threshold. On the other hand, if the viewpoint candidates do not need to be generated interactively and time is not a critical factor, using the curvature functional is preferable.

FF	Top/Bottom ($2\times$)	Lower S. ($4\times$)	Middle S. ($4\times$)	Upper S. ($4\times$)
$E_{\text{TP}}(S)$	0.27%	8.46%	8.16%	8.25%
$E_{\kappa}(S)$	0	8.71%	7.65%	8.64%

Table 6.4: *Feature functional (FF) values on individual surfaces of the spring model. The partition into top/bottom, lower surfaces, middle surfaces, and upper surfaces is consistent with Figure 6.11. The distribution of values of thin-plate energy and curvature mostly coincide, as the low thin-plate energy values on the top and bottom surface are almost neglectable.*

6.5.4 Intuition Behind Feature Functional Values

An important aspect for the usability of any method is how easy it is for the operator to understand and control the parameters. While the principles behind the feature functionals can be understood fairly easily, choosing a good termination threshold requires some knowledge of the expected values.

To a user, the integral values evaluated for the thin-plate energy or curvature are just numbers with no intuitive meaning. Instead of defining a fixed number as threshold, it is easier to normalize those values and set the threshold to a percentage of the total or average measurement. This is possible since all presented integral measurements can be summed up to a "total" value for the entire object.

The measurements for area and the region of interest are somewhat more intuitive. If the operator has a feeling for the dimensions of the objects, it is possible to define a fixed number of maximum area for the subdivision. However, since those measurements are also integral-based, the approach of using an average value also proves viable here.

While a threshold for the maximum angle-based measurements could also be chosen in relation to their average, it is actually more intuitive to set a fixed number for those functionals. The evaluated values are angles, which might already be explicitly determined by external constraints. Even if not, it is fairly intuitive for a user to understand the meaning of a maximum angle when setting up the parameters for the system.

6.5.5 Stability

To apply the recursive subdivision, a termination threshold needs to be chosen. Unless already given by some external constraints, a system engineer would usually determine this value experimentally.

Besides the intuition of the expected values, it is also important to be aware of the stability of the used feature functionals. In an optimal setting, small changes to the threshold value only add or remove a low amount of viewpoint candidates. However, in some cases, a small variation of the threshold can make the difference between subdividing a surface several times and no subdivision at all (e.g., Figures 6.6c and 6.6d).

Such behavior is analyzed by applying the iterative subdivision, i.e., always subdividing the surface with the highest feature value until a certain number of iterations has been done (Algorithm 2). Plotting the highest occurring feature functional value in relation to the number of iterations enables visual analysis of the stability behavior. Figure 6.12 demonstrates this approach for three individual surfaces. Each surface gets processed with 100 iterations of the subdivision approach. The plots show the behavior of all presented feature functionals except the region of interest, as it is just a restricted variety of the area measurement. Computing the recursive subdivision with a threshold value t corresponds to finding the first intersection between the graph and the horizontal line $y = t$.

Due to the partition of values when splitting a surface, the integral-based criteria show a monotonous decrease that becomes almost exponential when the surface has a regular geometry. This leads to a stable behavior when choosing a threshold, as well as a fast convergence of the recursive method.

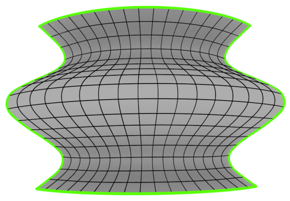
The angle-based functionals measure the maximum deviation of any normal vector from the average normal or a ray originating at the camera, respectively. However, when subdividing the surface, both the average normal and the camera position, are recomputed, possibly introducing significant jumps. As a consequence, sudden increases in the evaluated maximum deviation can occur. It is clear that these jumps do not occur anymore when the surface gets subdivided into sufficiently small segments. However, for surfaces with high frequency wiggles (e.g., Figure 6.12d), it will take many iterations until this criterion is met. For smoother surfaces, these functionals converge almost with the same rate as the integral-based measurements (see Figure 6.12a).

6.5.6 Conclusions

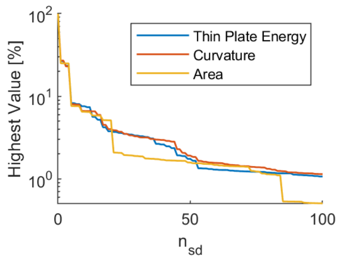
The presented method provides an adaptive and intuitive solution for object space exploration by generating desirable viewpoint candidates in sufficient numbers. It allows skilled system engineers to factor in expert knowledge, while at the same time producing viable results when used with the default settings. The experiments have shown that the thin-plate energy functional leads to well-distributed viewpoint candidates requiring little computational effort.

As the results show, objects can be fully covered with a small number viewpoint candidates. Thus, the subsequent NP-hard problem of selecting optimal viewpoints has to be solved only for a small input. The subdivision threshold, and with it the density of the surface sampling, can be chosen interactively. By pre-computing the subdivision to a certain depth, the operator can modify the threshold and evaluate the impact on the results in real time.

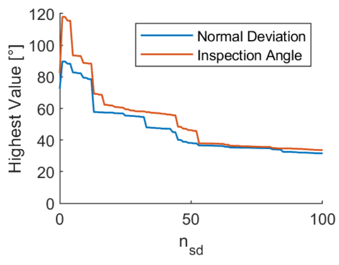
The use of B-splines makes the approach independent of the resolution of a discrete



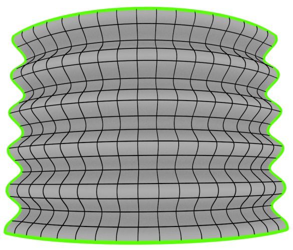
(a) *Surface with isolines*



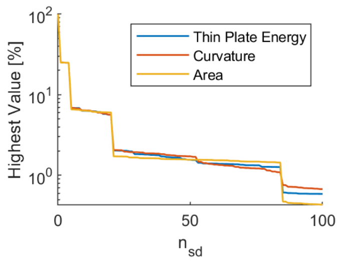
(b) *Integral-based criteria*



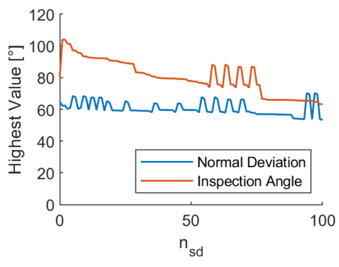
(c) *Max. angle-based criteria*



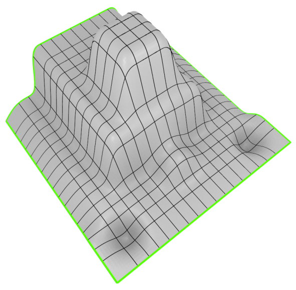
(d) *Surface with isolines*



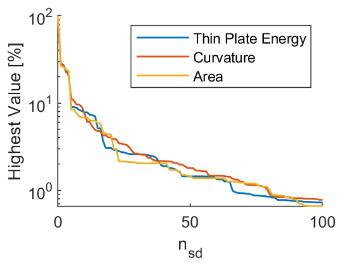
(e) *Integral-based criteria*



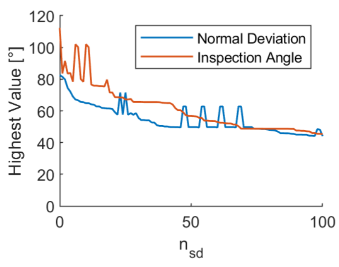
(f) *Max. angle-based criteria*



(g) *Surface with isolines*



(h) *Integral-based criteria*



(i) *Max. angle-based criteria*

Figure 6.12: *Analysis of the stability and convergence behavior of the presented functionals. Three selected surfaces (a), (d), (g) of different geometric complexity are considered. For each surface,* 100 *subdivision steps are performed. Each plot shows the highest measured value on the y-axis with respect to the number of performed subdivisions n_{sd} on the x-axis. The integral-based values (b), (e), and (h) are given as percentage of the total value measured for the entire surface, as it always equals the sum of all segments. Since the decrease is close to exponential, a logarithmic scale is used. Angle driven functionals (c), (f), (i) are given as absolute values. They can be related to the recursive subdivision by finding the first intersection between the plotted graphs and a vertical line representing the subdivision threshold $y = t$.*

representation as a single B-spline surface can model surface triangulations of various resolution. The analytic and continuously differentiable B-spline representation allows for the computation of various measurements. A user can easily factor in application-specific constraints or define new features to be focused on during inspection.

The overall pipeline for setting up inspection systems has existed for years and in many variations. Despite the desperate need to do so, full automatization for arbitrary objects has not yet been achieved. While the work presented here addresses one singular aspect of the inspection pipeline, it provides researchers and engineers with a powerful and adaptable solution, with the potential for further innovations. Continuous development of this approach will lead to smart inspection systems capable of adapting to new products within minutes, making production and inspection systems truly agile.

6.6 Core Reference

- D. Mosbach, P. Gospodnetić, M. Rauhut, B. Hamann, and H. Hagen: Feature driven viewpoint placement for model based surface inspection. *to be published in Machine Vision and Applications*.

Chapter 7

Conclusions

Gaining knowledge from raw data is an intricate process. The contributions presented in this thesis provide various improvements throughout the processing pipeline, starting with pre-processing, creating B-spline models of large and topologically challenging data, and finally using such models in an application scenario. The proposed methods improve existing solutions in terms of efficiency, hardware restrictions, and quality of the results.

The hybrid algorithm for quantile filtering, presented in Chapter 3, is a powerful tool for the processing of high-resolution 3D CT image data. Quantile filters are commonly used for gray value mathematical morphology. The median filter in particular is one of the most frequently used algorithms for image denoising. For large filter masks, existing solutions only allow to choose between high memory requirements or long runtimes. The proposed hybrid algorithm offers a trade-off by allowing to tune the performance to the available hardware, i.e., gaining additional speed, even on machines with limited memory. The presented results showed that, in 3D, the hybrid approach outperforms both reference methods for most practically relevant filter sizes.

When it comes to extracting a B-spline surface model from a volume image, even data without noise can pose a challenge. An example is the reconstruction of cellulose fibers, discussed in Chapter 4. Difficulties arise from the complex surface topology caused by the physical processing of the material. Standard approaches fail to partition the surface data into a low number of quadrilateral cells, suited for the construction of a B-spline model. This thesis proposed a solution based on splitting the geometry and topology processing into individual pipelines. The method was applied to several fibers which were extracted from scans of a wood-based insulation material. Triangulated surfaces of those fibers ranged from $423\,628$ to $1\,203\,844$ triangles. Resulting B-spline models consisted of 236 to 676 surfaces, providing a good representation while robustly dealing with holes, cuts, and chunks of excess material.

Chapter 5 introduced a novel approach for the approximation of triangular meshes with smooth B-spline surface models. Contrary to most existing solutions, the proposed method does not impose restrictions to knot vectors or spline orders. All control points are free variables which are successively computed during a local multi-phase construction process. Approximate tangent plane continuity is achieved by augmenting standard minimization problems with nonlinear terms. Through its local design, the algorithm is not hindered by the higher computational complexity that these additional terms cause. This makes the method viable for large or even incomplete data. Various measurements were discussed to evaluate the quality of B-spline models. Resulting models were shown for several objects with varying complexity. Their quality was compared with models obtained by a C^0-continuous and G^1-continuous method. This comparison indicated that the novel method improves data approximations with smooth surfaces on average by a factor of 10 at the cost of having small discontinuities in normal vectors of less than 1 degree.

An important use-case of accurate B-spline models was demonstrated in Chapter 6. Based on those models, a novel, purely virtual method was presented to setup visual inspection systems. These systems utilize robots to take pictures from pre-defined camera positions. Finding good viewpoints automatically removes the need for the time intensive manual configuration by experts. The focus of this work was the initial generation of viewpoint candidates. Through the definition of various feature functionals, the method is able to perform a non-uniform sampling of given objects with respect to user-defined criteria. Results were shown for various test objects. In all cases, a low amount of viewpoint candidates was computed using the proposed method. From those, a set of final viewpoints, optimal for the inspection system, was chosen via a greedy approach. The in-depth analysis of the feature functionals showed that thin-plate energy makes a good choice as it can be computed in real-time and provides a good overall distribution of viewpoints with a focus on highly curved regions.

Creating B-spline models from scanned data and gaining the desired insight is a complex task. Many factors impact the viability and efficiency of available strategies and solutions usually have to be tailored specifically for the target application. This thesis proposed several novel methods to deal with difficulties that can arise at various stages of this process. Besides discussing new algorithms, it also covered efficient approaches that do not require high-end hardware. Many aspects of the discussed solutions are highly flexible and can easily be modified to handle new challenges. The presented contributions improve virtual processing of real-world data. They have the potential to impact almost all scientific fields and lead to new advances.

Bibliography

[1] L. Alparone, V. Cappellini, and A. Garzelli. A coarse-to-fine algorithm for fast median filtering of image data with a huge number of levels. *Signal Processing*, 39(1-2):33–41, 1994.

[2] K. E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[3] E. Badel, C. Delisee, and J. Lux. 3D structural characterisation, deformation measurements and assessment of low-density wood fibreboard under compression: The use of X-ray microtomography. *Composites Science and Technology*, 68(7-8):1654–1663, 2008.

[4] J. Beyerer, F. P. León, and C. Frese. *Machine Vision: Automated Visual Inspection: Theory, Practice and Applications*. Springer, 2015.

[5] Blender Online Community. Blender - a 3D modelling and rendering package. `http://www.blender.org`, 2019. [Online; accessed 2019-12-30].

[6] A. Blidia, B. Mourrain, and N. Villamizar. G^1-smooth splines on quad meshes with 4-split macro-patch elements. *Computer Aided Geometric Design*, 52:106–125, 2017.

[7] G.-P. Bonneau and S. Hahmann. Flexible G^1 interpolation of quad meshes. *Graphical Models*, 76(6):669–681, 2014.

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[10] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, 1988.

[11] C. De Boor et al. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

[12] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications, 2016.

[13] D. Dobrovolskij, M. Engelhardt, A. Rack, and K. Schladitz. Shape classification for wood based insulation material. Presentation, July 2019. contributed to ICTMS 2019, Cairns (Australia).

[14] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *Acm transactions on graphics (tog)*, 25(3):1057–1066, 2006.

[15] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.

[16] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM, 1995.

[17] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 325–334. ACM, 1996.

[18] B. J. Englot. *Sampling-Based Coverage Path Planning for Complex 3D Structures*. PhD thesis, Massachusetts Institute of Technology, 9 2012.

[19] J. Fan and J. Peters. On smooth bicubic surfaces from quad meshes. In *International Symposium on Visual Computing*, pages 87–96. Springer, 2008.

[20] J. Fan and J. Peters. Smooth bi-3 spline surfaces with fewest knots. *Computer-Aided Design*, 43(2):180–187, 2011.

[21] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 5th edition, 2002.

[22] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice (2nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., USA, 1990.

[23] Fraunhofer ITWM. ToolIP – tool for image processing. `https://www.itwm.fraunhofer.de/en/departments/bv/products-and-services/toolip.html`, 2019. [Online; accessed 2019-12-30].

[24] Fraunhofer ITWM, Department of Image Processing. MAVI – modular algorithms for volume images. `http://www.mavi-3d.de`, 2005. [Online; accessed 2019-12-30].

[25] M. Godehardt, D. Mosbach, D. Roldan, and K. Schladitz. Efficient 3D erosion dilation analysis by sub-pixel EDT. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 243–255. Springer, 2019.

[26] P. Gospodnetić and F. Hirschenberger. Detection and visibility estimation of surface defects under various illumination angles using bidirectional reflectance distribution function and local binary pattern. In *Croatian Computer Vision Workshop, Year 4*, pages 9–14, 2016.

[27] B. F. Gregorski, B. Hamann, and K. I. Joy. Reconstruction of B-spline surfaces from scattered data points. In *Proceedings Computer Graphics International 2000*, pages 163–170. IEEE, 2000.

[28] M. Gronle and W. Osten. View and sensor planning for multi-sensor surface inspection. *Surface Topography: Metrology and Properties*, 4(2), 2016.

[29] G. Guennebaud, B. Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010. [Online; accessed 2019-12-30].

[30] H. Hagen and G. Schulze. Automatic smoothing with geometric surface patches. *Computer Aided Geometric Design*, 4(3):231–235, 1987.

[31] S. Hahmann, G.-P. Bonneau, and B. Caramiaux. Bicubic G^1 interpolation of irregular quad meshes using a 4-split. In *International Conference on Geometric Modeling and Processing*, pages 17–32. Springer, 2008.

[32] J. Hoschek, D. Lasser, and L. L. Schumaker. *Fundamentals of computer aided geometric design*. AK Peters, Ltd., 1993.

[33] J. Hsieh. *Computed tomography: principles, design, artifacts, and recent advances*. SPIE Bellingham, WA, 2009.

[34] J. Huang, Y. Zhou, M. Niessner, J. R. Shewchuk, and L. J. Guibas. Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum*, volume 37, pages 147–160. Wiley Online Library, 2018.

[35] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

[36] A. Jacobson, D. Panozzo, et al. libigl: A simple C++ geometry processing library. `https://libigl.github.io`, 2018. [Online; accessed 2019-12-30].

[37] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189–1, 2015.

[38] W. Jing. *Coverage Planning for Robotic Vision Applications in Complex 3D Environment*. PhD thesis, Carnegie Mellon University, 2017.

[39] S. G. Johnson. The nlopt nonlinear-optimization package. `http://github.com/stevengj/nlopt`, 2014. [Online; accessed 2019-12-30].

[40] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`, 2001–. [Online; accessed 2019-12-30].

[41] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[42] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH*, volume 96, pages 313–324, 1996.

[43] J.-Y. Lai and W.-D. Ueng. G^2 continuity for multiple surfaces fitting. *The International Journal of Advanced Manufacturing Technology*, 17(8):575–585, 2001.

[44] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, et al. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

[45] H. Lin, W. Chen, and H. Bao. Adaptive patch-based mesh fitting for reverse engineering. *Computer-Aided Design*, 39(12):1134–1142, 2007.

[46] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[47] C. Marulier, P. J. Dumont, L. Orgéas, S. R. du Roscoat, and D. Caillerie. 3D analysis of paper microstructures at the scale of fibres and bonds. *Cellulose*, 22(3):1517–1539, 2015.

[48] A. Mavrinac, X. Chen, and J. L. Alarcon-Herrera. Semiautomatic model-based view planning for active triangulation 3-D inspection systems. *IEEE/ASME Transactions on Mechatronics*, 20(2):799–811, 2015.

[49] M. J. Milroy, C. Bradley, G. W. Vickers, and D. Weir. G^1 continuity of B-spline surface patches in reverse engineering. *Computer-Aided Design*, 27(6):471–478, 1995.

[50] M. Mohammadikaji, S. Bergmann, S. Irgenfried, J. Beyerer, C. Dachsbacher, and H. Worn. Inspection planning for optimized coverage of geometrically complex surfaces. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 52–67. IEEE, 2018.

[51] D. Mosbach, P. Gospodnetić, M. Rauhut, B. Hamann, and H. Hagen. Feature driven viewpoint placement for model based surface inspection. *Machine Vision and Applications*, 2019. Submitted.

[52] D. Mosbach, H. Hagen, M. Godehardt, and O. Wirjadi. Fast and memory-efficient quantile filter for data in three and higher dimensions. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2928–2932. IEEE, 2014.

[53] D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen. A local approach for computing smooth B-spline surfaces on arbitrary quadrilateral base meshes. *ASME Journal of Computing and Information Science in Engineering*, 2019. Submitted.

[54] D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen. Topologically robust B-spline reconstruction of fibers from 3D images. *Topological Methods in Visualization: Theory, Software and Applications*, 2019. Submitted.

[55] B. Mourrain, R. Vidunas, and N. Villamizar. Dimension and bases for geometrically continuous splines on surfaces of arbitrary topology. *Computer Aided Geometric Design*, 45:108–133, 2016.

[56] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens. ADR visualization: A generalized framework for ranking large-scale scientific data using analysis-driven refinement. In *2014 ieee 4th symposium on large data analysis and visualization (ldav)*, pages 43–50. IEEE, 2014.

[57] J. Ohser and K. Schladitz. *3D images of materials structures: processing and analysis*. John Wiley & Sons, 2009.

[58] S. Perreault and P. Hébert. Median filtering in constant time. *IEEE Trans. Image Processing*, 16(9):2389–2394, 2007.

[59] J. Peters. Constructing C^1 surfaces of arbitrary topology using biquadratic and bicubic splines. In *Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design*, pages 277–293. SIAM, 1994.

[60] J. Peters and J. Fan. On the complexity of smooth spline surfaces from quad meshes. *Computer Aided Geometric Design*, 27(1):96–105, 2010.

[61] C. Peyrega, D. Jeulin, C. Delisée, and J. Malvestio. 3D morphological characterization of phonic insulation fibrous media. *Advanced Engineering Materials*, 13(3):156–164, 2011.

[62] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[63] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[64] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Springer Science & Business Media, 2013.

[65] F. Prieto, R. Lepage, P. Boulanger, and T. Redarce. A CAD-based 3D data acquisition strategy for inspection. *Machine Vision and Applications*, 15:76–91, 2003.

[66] D. F. Rogers and N. R. Fog. Constrained B-spline curve and surface fitting. *Computer-Aided Design*, 21(10):641–648, 1989.

[67] S. Sakane, R. Niepold, T. Sato, and Y. Shirai. Illumination setup planning for a hand-eye system based on an environmental model. *Advanced Robotics*, 6(4):461–482, 1991.

[68] S. Sakane and T. Sato. Automatic planning of light source and camera placement for an active photometric stereo system. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1080–1087 vol.2, 1991.

[69] W. R. Scott. Model-based view planning. *Machine Vision and Applications*, 20:47–69, 2009.

[70] W. Sheng, N. Xi, M. Song, Y. Chen, and J. S. Rankin. Automated CAD-guided automobile part dimensional inspection. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1157–1162. IEEE, 2000.

[71] W. Sheng, N. Xi, J. Tan, M. Song, and Y. Chen. Viewpoint reduction in vision sensor planning for dimensional inspection. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, volume 1, pages 249–254. IEEE, 2003.

[72] X. Shi, T. Wang, and P. Yu. A practical construction of G^1 smooth biquintic B-spline surfaces over arbitrary topology. *Computer-Aided Design*, 36(5):413–424, 2004.

[73] P. Slavık. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237 – 254, 1997.

[74] P. Soille. On morphological operators based on rank filters. *Pattern recognition*, 35(2):527–535, 2002.

[75] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Trans. Robotics and Automation*, 11(1):86–104, 1995.

[76] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. The MVP sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1):72–85, Feb 1995.

[77] G. H. Tarbox and S. N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 65(1):84–111, 1995.

[78] The CGAL Project. CGAL user and reference manual. `https://doc.cgal.org/4.14/Manual/packages.html`, 2019. [Online; accessed 2019-12-30].

[79] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017. `https://topology-tool-kit.github.io/` [Online; accessed 2019-12-30].

[80] B. Weiss. Fast median and bilateral filtering. *ACM Transactions on Graphics (TOG)*, 25(3):519–526, 2006.

[81] T. Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, page 4. ACM, 2005.

[82] F. Wolf, P. Ebermann, F. Lackner, D. Mosbach, C. Scheuerlein, K. Schladitz, and D. Schoerling. Characterization of the stress distribution on Nb3Sn Rutherford cables under transverse compression. *IEEE Transactions on Applied Superconductivity*, 28(3):1–6, 2017.

[83] D.-J. Yoo. Three-dimensional surface reconstruction of human bone using a B-spline based interpolation approach. *Computer-Aided Design*, 43(8):934–947, 2011.

[84] H. Yoshihara, T. Yoshii, T. Shibutani, and T. Maekawa. Topologically robust B-spline surface reconstruction from point clouds using level set methods and iterative geometric fitting algorithms. *Computer Aided Geometric Design*, 29(7):422–434, 2012.

[85] X. Zhao, C. Zhang, L. Xu, B. Yang, and Z. Feng. IGA-based point cloud fitting using B-spline surfaces for reverse engineering. *Information Sciences*, 245:276–289, 2013.

[86] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, and J. Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.

Appendix A

Curriculum Vitae

Education

March 2009	Abitur, Nikolaus-von-Weis Gymnasium, Speyer, Germany
August 2013	Bachelor of Science (B. Sc.), Applied Computer Science, University of Kaiserslautern, Germany
December 2015	Master of Science (M. Sc.), Applied Computer Science, University of Kaiserslautern, Germany

Academic and Professional Experience

12/2011 - 09/2012	Student assistant, Institute for Manufacturing Technology and Production Systems, University of Kaiserslautern, Germany
12/2013 - 06/2014	Student assistant, Fraunhofer ITWM, Kaiserslautern, Germany
07/2014 - 10/2014	Visiting student within the scope of the Data Science at Scale Summer School, Los Alamos National Laboratories, New Mexico, USA

Appendix B

List of Publications

Core publications

- D. Mosbach, P. Gospodnetić, M. Rauhut, B. Hamann, and H. Hagen: Feature driven viewpoint placement for model based surface inspection. *to be published in Machine Vision and Applications*.
- D. Mosbach, H. Hagen, M. Godehardt, and O. Wirjadi: Fast and memory-efficient quantile filter for data in three and higher dimensions. *IEEE International Conference on Image Processing (ICIP)*, pp. 2928-2932, (2014).
- D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen: Topologically robust B-spline reconstruction of fibers from 3D images. *to be published in Topological Methods in Visualization: Theory, Software and Applications*.
- D. Mosbach, K. Schladitz, B. Hamann, and H. Hagen: A local approach for computing smooth B-spline surfaces on arbitrary quadrilateral base meshes. *to be published in ASME Journal of Computing and Information Science in Engineering*.

Further publications

- M. Godehardt, D. Mosbach, D. Roldan, and K. Schladitz: Efficient 3D erosion dilation analysis by sub-pixel EDT. *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pp. 243-255 (2019).
- F. Wolf, P. Ebermann, F. Lackner, D. Mosbach, C. Scheuerlein, K. Schladitz, and D. Schoerling: Characterization of the stress distribution on Nb3Sn Rutherford cables under transverse compression. *IEEE Transactions on Applied Superconductivity*, 28(3), pp. 1-6 (2017).