Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Computation in Engineering

Image-based finite element analysis

László Kudela, M.Sc.

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Prof. Dr.-Ing. Fabian Duddeck
Prüfer der Dissertation:	1.	Prof. Dr. rer. nat. Ernst Rank
	2.	Prof. Dr.-Ing. Uwe Stilla
	3.	Prof. Zohar Yosibash, D.Sc

Die Dissertation wurde am 03.02.2020 bei der Technischen Universität München eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt am 12.03.2020 angenommen.

Abstract

This thesis demonstrates the steps which are needed to be taken to construct a finite element model for objects that do not possess a digital CAD representation. Usually, this process comprises the following main steps: shape acquisition, geometry recovery, mesh generation, application of boundary conditions and material parameters, and finite element analysis. This work revisits the steps of this *measurement-to-analysis* pipeline, exploring various techniques to deal with the difficulties associated to the individual stages or even circumvent them completely.

The complete pipeline is demonstrated through an application that aims at constructing finite element meshes for tubular objects that are immersed in a refractive medium. To this end, a modified bundle adjustment formulation is presented that allows to account for the distortion caused by the refractive effects.

Furthermore, it is shown how the potentially difficult step of mesh generation can be avoided by employing a high-order immersed boundary technique, the Finite Cell Method (FCM). One challenging aspect of the FCM is that elements of the background mesh that are cut by the physical boundary give rise to discontinuous integrands. Within this context, a numerical integration technique is presented that allows for computing such discontinuous integrals in an efficient yet accurate manner. Two- and three-dimensional examples demonstrate that these methods yield more accurate results than traditional approaches but remain computationally cheaper.

Finally, it is shown how the Finite Cell Method can be combined with point cloud-based geometric representations. This way, structural analysis becomes possible directly on the data measured in the shape acquisition step, allowing for significant simplifications in the measurement-to-analysis pipeline. For this application, a solution strategy to the issue of weak enforcement of boundary conditions on point-based surfaces is outlined. The concept is demonstrated through numerical examples computed on historical structures.

Zusammenfassung

Diese Arbeit untersucht, wie Finite-Elemente-Modelle von Objekten erzeugt werden können, die keine CAD-Darstellung besitzen. In der Regel werden hierzu folgende Schritte durchlaufen: Formerfassung, Wiederherstellung der Geometrie, Netzgenerierung, Aufbringung der Randbedingungen und Eingabe der Materialparameter sowie die Finite-Elemente-Analyse.

Dieser Prozess von der Messung bis zur Berechnung wird in der vorliegenden Arbeit neu betrachtet. Dabei werden verschiedene Methoden erforscht, um Schwierigkeiten bei der Durchführung einzelner Schritte zu bewältigen oder sogar vollständig zu umgehen. Das komplette Vorgehen wird anhand eines Anwendungsbeispiels demonstriert, in dem Finite-Elemente-Netze für rohrförmige Objekte erzeugt werden sollen, die in einem lichtbrechenden Medium eingelegt sind. Zu diesem Zweck wird eine modifizierte Formulierung der Bündelausgleichung vorgestellt, mit der die Verzerrungen infolge der Brechungseffekte berücksichtigt werden können.

Darüber hinaus wird gezeigt, wie der potenziell schwierige Schritt der Netzgenerierung durch den Einsatz einer Embedded Domain Methode hoher Ordnung, der Finite-Zellen-Methode (FCM), vermieden werden kann. Ein herausfordernder Aspekt der FCM ist, dass Elemente des Hintergrundnetzes diskontinuierliche Integranden aufweisen. In diesem Zusammenhang wird eine numerische Integrationstechnik vorgestellt, die eine effiziente und dennoch genaue Berechnung solcher diskontinuierlichen Integrale ermöglicht. Mithilfe zwei- und dreidimensionaler Beispiele wird gezeigt, dass diese Methoden genauere Ergebnisse liefern als traditionelle Ansätze und mit einem geringeren Rechenaufwand verbunden sind.

Schließlich wird gezeigt, wie die Finite-Zellen-Methode mit punktwolkenbasierten geometrischen Modellen kombiniert werden kann. Auf diese Weise ist es möglich, Strukturanalysen unmittelbar auf der Grundlage von Daten durchzuführen, die bei der Formerfassung gemessen werden. Der Prozess von der Messung zur Berechnung lässt sich dadurch erheblich vereinfachen. In diesem Kontext wird eine Lösungsstrategie für das Problem der schwachen Aufbringung von Randbedingungen auf punktbasierten Oberflächen dargestellt. Die Anwendung des Konzepts der punktwolkenbasierten Strukturanalyse auf historische Strukturen wird anhand numerischer Beispiele demonstriert.

Acknowledgments

This thesis was created during my time as a PhD student at the Chair for Computation in Engineering at the Technical University of Munich, in the years from 2013 to 2019. Many people supported me on the way towards completing this work. In the following, I would like to gratefully acknowledge their contribution.

I thank my supervisor, Prof. Ernst Rank for his trust in me, patience, and the continuous support he provided during all these years. I especially thank him for the supportive working environment he created at the chair, where everyone is given the room and opportunity to grow freely, at their own pace.

I thank Prof. Zohar Yosibash and Prof. Uwe Stilla for being the reviewers of my thesis and Prof. Fabian Duddeck for chairing my examination.

I thank Dr. Stefan Kollmannsberger for keeping his door always open and being ready to listen and discuss. His support on both scientific and non-scientific matters contributed greatly towards the completion of this thesis.

The initial phase of my PhD was partially financed by the German-Israeli Foundation for Scientific Research and Development under grant no. GIF-1189-89.2/2012. This financial support is gratefully acknowledged. In the scope of this project, I had the pleasure to work together with the members of the experimental biomechanics laboratory at the Ben Gurion University of the Negev, Israel. I thank Prof. Zohar Yosibash for the hospitality during my visits in Beersheba. I've always felt welcome and had a great time visiting his group. I also owe my thanks to Prof. Yosibash for all the discussions we had during these years and for all the valuable insights he provided me. I gratefully acknowledge the effort of the people of the lab who designed and operated the experimental device of Section 3.4.4. In particular, I would like to thank Ofry Yossef, Avihai Uzan and Itay Manor for their effort in preparing the test samples and for the numerous photos they recorded for me using the device during the years 2013-2017.

I want to thank all the people who I had the pleasure to meet and work together with from the *Chair for Computation in Engineering* and *Chair of Computational Modeling and Simulation*. They made the 3$^{\text{rd}}$ floor of building no. 1 of the TUM a place where I was always happy to arrive in the morning.

In particular, I am grateful to have shared an office for a bit more than six years with my friend and colleague, Mohamed Elhaddad. I enjoyed all the discussions and activities we had in this time, in the office and outside of the university as well. I also thank Mohamed for proofreading this thesis.

I want to thank my family: my parents and my brothers for their unconditional support and accepting my decisions during my whole life.

Finally, I want to thank my girlfriend Emese for her loving understanding during the past years.

László Kudela
Munich, October 2020

:wq

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For many decades, the Finite Element Method (FEM) has been serving engineers and scientists in their quest towards getting better insight into a wide variety of physical phenomena. Thanks to the increasing availability and efficiency of computational resources, the FEM has become able to deal with problems of scales that were probably seen as unreachable at the time of its introduction. Although it was initially developed to solve problems in structural mechanics, continuous research in the field has made it possible to apply the method in other areas of science and engineering as well: fluid mechanics, acoustics, biomechanics, just to name a few.

Regardless of the field of application or the problem size, a fundamental step in most applications of the method is the decomposition of the geometry of interest into an interconnected network of finite elements. Designing reliable and efficient meshing algorithms is a science (some would even say "art") of its own, attracting nearly as much research interest as the FEM itself. Such algorithms need to deal with increasingly complex geometric models, involve as little human interaction as possible but still produce an analysis-suitable finite element mesh.

In most applications, the geometry to be meshed is given to the mesh generator in the form of a digital CAD model. When such CAD models are designed, numerical analysis is not the only aspect that is kept in mind: there is a big variety of downstream applications that all derive their representation from a geometric model or a set of models. They form the basis for the manufacturing process, product documentations, maintenance planning, photorealistic renderings and other elements from the product life cycle management circle. Therefore, prior to conducting a finite element analysis on a geometric model, the CAD file needs to be brought in a form which is suitable for mesh generation. This step is hard to automate and is consequently often performed manually. In certain scenarios the effort invested in preparing an analysis-suitable geometric model and the subsequent meshing step may require more effort than the actual numerical analysis itself [1]. Due to these reasons, the question of bridging the gap between geometric modeling and numerical analysis has attracted a lot of research interest in computational mechanics recently.

1.1 Motivation

Apart from simulations starting from a CAD model, there is a strong trend towards applications of the FEM where the mesh is derived from alternative geometric representations. Typically, this is the case in the context of biomechanical simulations, where models are recorded

by medical imaging techniques, such as qCT scans. These methods require special algorithms to recover a geometric model and eventually a finite element mesh from the imaging data – see e.g. [2] for a conceptual overview of these multi-step pipelines.

For some physical structures, volumetric imaging is not the most cost efficient approach to record the shape of interest. It is especially large objects that do not allow for a direct application of CT scanning. Nonetheless, it can be of importance to be able to compute the structural behavior of large objects – e.g. in the field of cultural heritage preservation, as there are often no digital CAD models available for historical structures. Moreover, even if there are schematic drawings, the actual shape of the object may differ from them, especially if the structure is exposed to damaging effects such as erosion, floods, earthquakes, or wars. In these cases, other shape measurement techniques need to be employed. The two most popular methods for this purpose are terrestrial laser scanning and close range photogrammetry-based reconstructions. Especially photogrammetry has gained a lot of attention recently, due to the inexpensiveness of the required equipment and because of the rapid development of the computational resources as well as the associated algorithms that allow for efficient, almost real-time reconstructions [3].

The methods of laser scanning and photogrammetry both reproduce the shape of the geometry of interest in the form of point clouds: a set of points representing the surface of the object. Such point clouds are not directly suited for numerical analysis. In order to transform the recorded data into an analysis-suitable model, it needs to pass through several stages, similar to the necessary procedure for models stemming from volumetric imaging.

Usually, these *measurement-to-analysis* procedures are characterized by the following main steps (Figure 1.1):

1. *Shape acquisition*
   A 3D shape measurement technique is employed to capture the shape of the domain of interest, resulting in a point cloud representing the surface of the object.
2. *Surface reconstruction*
   A geometric model is derived from the point cloud information using geometric segmentation and surface fitting methods. The resulting model is stored using standardized geometric representation techniques, such as STL, STEP, or IGES files.
3. *Mesh generation*
   The CAD model from the previous step is discretized into a finite element mesh.
4. *Finite Element Analysis*
   The mesh is handed over to a finite element solver together with the corresponding material properties and structural constraints.

Numerous applications implement the steps above – see e.g. [5–9] for examples in the preservation of historical structures.

Research in different fields of computational science and engineering has resulted in well-established approaches that allow to perform these steps one-by-one. Still, their deep integration into a seamless chain is not trivial, as it requires the interplay of various algorithms. Other than the problems inherent to the data transfer between different implementations, an even bigger challenge is posed by generating a finite element mesh from the geometric model reconstructed in the second step. The fine details recovered by modern surface reconstruction algorithms (e.g.[10–12]) are not necessarily the details that need to be carried over to a finite element mesh, where the process of refinement is usually governed by the physics of the

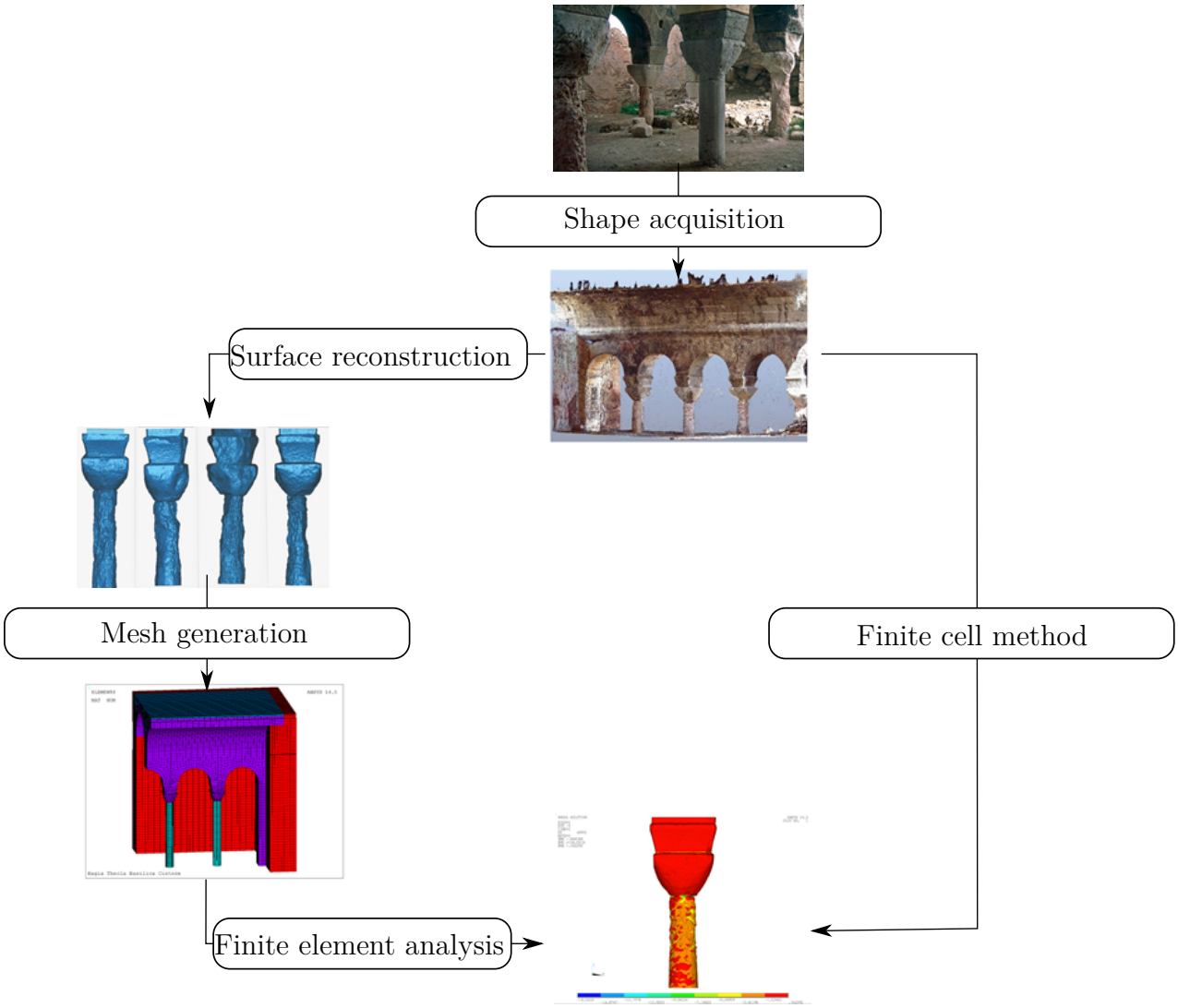


Figure 1.1: *The main steps of the measurement-to-analysis pipeline [4]. The main stages of usual approaches are on the left side, they are discussed in Chapters 2 and 3. The right side shows an alternative solution proposed within Chapter 4.*

problem rather than aesthetic aspects. While a geometric defeaturing step may be applied to remove physically uninteresting details, manipulating the geometry carries the danger of introducing flaws in the geometric model, resulting in an invalid, "dirty" geometry that cannot be meshed directly [13, 14].

1.2 Structure, contributions

The goal of this work is to explore and demonstrate the techniques that can be employed to circumvent the difficulties that are encountered when performing numerical analysis based on digital shape measurements. Accordingly, the structure of the thesis follows the main building blocks of the measurement-to-analysis pipeline of Figure 1.1.

Chapter 2 gives a short introduction to the Finite Element Method and demonstrates an

alternative approach, the Finite Cell Method (FCM) which can be employed in order to avoid the difficult task of mesh generation. One challenge which is often faced within the context of the FCM is the question of accurate numerical integration. Within the chapter, an alternative numerical integration strategy (integration by smart octrees) is introduced which aims at circumventing this problem in an accurate but efficient manner. It will be demonstrated by numerical examples how the proposed integration strategy compares to conventional techniques employed within the FCM.

Chapter 3 discusses digital shape measurement techniques, with a special focus on image-based 3D reconstructions. Within the chapter, a modified bundle adjustment approach is described that can be employed when the recording device is moved around an object which is immersed in a refractive medium. The proposed method will be demonstrated by numerical examples on synthetic and real images.

Having demonstrated the basic building blocks of the standard measurement-to-analysis pipeline, Chapter 4 discusses how the FCM can be combined with the data coming from digital shape measurements, i.e. oriented point clouds. The members of the point cloud and the vectors associated to them provide the bare minimum geometric information that is needed by the FCM, allowing for structural analyses of objects directly on their cloud representation. This way, the tedious tasks of recovering a geometric model and generating a boundary conforming mesh can be avoided, allowing for significant simplifications in the measurement-to-analysis pipeline, as represented on the right side of Figure 1.1. The capabilities of the method will be shown through various examples from the field of cultural heritage preservation.

The contributions presented within this thesis have been published in several scientific articles:

- The two-dimensional numerical integration method was introduced in [15], with a more thorough discussion on the computational effort of the algorithm presented in [16].
- *Smart octrees*, the extension of the algorithm to three dimensions was described in [17].
- The modified bundle adjustment algorithm aiming at the meshing of tubular objects immersed in refractive media was introduced in [18].
- The combination of the Finite Cell Method and point cloud-based geometry representations was presented in [4].

Throughout the thesis, the following footnotes are used to reference the original publications and literal transposition. [a] [b] [c] [d]

[a] The following chapter/section/paragraph is based on [16]. The main scientific research as well as the textual elaboration of the publication was performed by the author of this work.

[b] The following chapter/section/paragraph is based on [17]. The main scientific research as well as the textual elaboration of the publication was performed by the author of this work.

[c] The following chapter/section/paragraph is based on [18]. The main scientific research as well as the textual elaboration of the publication was performed by the author of this work.

[d] The following chapter/section/paragraph is based on [4]. The main scientific research as well as the textual elaboration of the publication was performed by the author of this work.

Chapter 2

The Finite Element and Finite Cell Methods

The aim of this chapter is to outline the fundamental ideas of the Finite Element and Finite Cell Methods. The first part of the chapter focuses on the formulation of the standard, boundary conforming finite elements and their p-extension. The second part outlines the key concepts of the Finite Cell Method, with a special focus on numerical quadratures and their effect on the efficiency and accuracy of the method. In this context, an enhanced quadrature technique—the method of smart-octrees—is introduced, showing that the accuracy of the FCM can be drastically improved for geometric models with explicit, high-order surface information. The closing remarks of the chapter point out the role of geometric representations for the FEM and FCM. This motivates the next chapter that addresses the question of how to derive an analysis-suitable numerical model from geometries that do not possess a geometric representation in the form of a CAD file.

2.1 Fundamentals of the Finite Element Method

In the following, the formulation of the finite element method is summarized for problems of linear elasticity.

2.1.1 Galerkin method

Let $n_d (= 1, 2 \text{ or } 3)$ denote the number of space dimensions of the problem under consideration, a physical body represented by the domain Ω (e.g. as shown for $n_d = 2$ in Figure 2.1), and the boundary $\partial\Omega$ is divided into two parts Γ_{N} and Γ_{D} such that $\Gamma_{\mathrm{N}} \cap \Gamma_{\mathrm{D}} = \emptyset$. Assuming that the material of the body is linear elastic, the physical behavior of the structure is formally

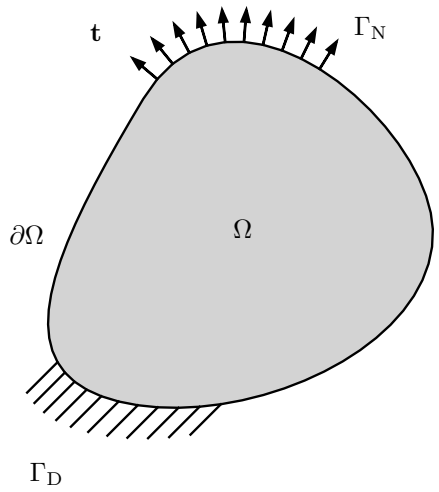


Figure 2.1: *Conceptual sketch of a linear elastic problem on the domain Ω, with $n_d = 2$.*

described by the following boundary value problem in its strong form:

$$\begin{aligned}
&\text{Given } \boldsymbol{b} : \Omega \to \mathbb{R}^{n_d}, \hat{\boldsymbol{t}} : \Gamma_\mathrm{N} \to \mathbb{R}^{n_d}, \hat{\boldsymbol{u}} : \Gamma_\mathrm{D} \to \mathbb{R}^{n_d}, \text{ find } \boldsymbol{u} : \Omega \to \mathbb{R}^{n_d} \text{ such that} \\
&\qquad \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \qquad \forall \boldsymbol{x} \in \Omega \\
&\qquad \boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\epsilon} \qquad \forall \boldsymbol{x} \in \Omega \\
&\qquad \boldsymbol{\epsilon} = \frac{1}{2}\left[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\mathrm{T}\right] \qquad \forall \boldsymbol{x} \in \Omega \\
&\qquad \boldsymbol{\sigma} \cdot \boldsymbol{n} = \hat{\boldsymbol{t}} \qquad \forall \boldsymbol{x} \in \Gamma_\mathrm{N} \\
&\qquad \boldsymbol{u} = \hat{\boldsymbol{u}} \qquad \forall \boldsymbol{x} \in \Gamma_\mathrm{D},
\end{aligned} \tag{2.1}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ denote the stress and strain tensor, respectively. Further, $\boldsymbol{C}$ is the linear elastic constitutive tensor, $\hat{\boldsymbol{t}}$, $\hat{\boldsymbol{u}}$ and $\boldsymbol{b}$ are the prescribed displacement, traction and body load vectors, and $\boldsymbol{n}$ is the outward-pointing unit normal vector on the boundary $\partial\Omega$. The formulation of the finite element method starts with transforming the strong form into a weak formulation:

$$\begin{aligned}
&\text{Given } \boldsymbol{b} : \Omega \to \mathbb{R}^{n_d}, \hat{\boldsymbol{t}} : \Gamma_\mathrm{N} \to \mathbb{R}^{n_d}, \hat{\boldsymbol{u}} : \Gamma_\mathrm{D} \to \mathbb{R}^{n_d}, \text{ find } \boldsymbol{u} \in \mathcal{S} \\
&\text{such that for all } \boldsymbol{v} \in \mathcal{V} \\
&\qquad a(\boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{v}), \\
&\text{with} \quad a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{\epsilon}(\boldsymbol{v}) : \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \mathrm{d}\Omega \\
&\text{and} \quad f(\boldsymbol{v}) = \int_\Omega \boldsymbol{v} \cdot \boldsymbol{b} \mathrm{d}\Omega + \int_{\Gamma_\mathrm{N}} \boldsymbol{v} \cdot \hat{\boldsymbol{t}} \mathrm{d}\Gamma.
\end{aligned} \tag{2.2}$$

In the above statement, the solution function $\boldsymbol{u}$ and the weighting function $\boldsymbol{v}$ are prescribed to be found in the Sobolev space of first order $H^1(\Omega)$. Further, $\boldsymbol{u}$ and $\boldsymbol{v}$ comply to the Dirichlet boundary conditions and their homogeneous counterparts:

$$\begin{aligned}
\mathcal{S} &= \left\{ u_i \in H^1(\Omega) : u_i = \hat{u}_i \quad \forall \boldsymbol{x} \in \Gamma_\mathrm{D} \right\} \\
\mathcal{V} &= \left\{ v_i \in H^1(\Omega) : v_i = 0 \quad \forall \boldsymbol{x} \in \Gamma_\mathrm{D} \right\}.
\end{aligned} \tag{2.3}$$

The transformation of the strong form into its weak counterpart can be accomplished by various techniques, e.g. the principle of virtual work, the method of weighted residuals or the minimization of the total potential energy.

While the equations of the weak form possess a rather simple structure, their analytical solution is only possible in simple cases. Therefore, the finite element method seeks to cast Equation 2.2 into a discretized representation that allows for a numerical approximation of the solution. For standard engineering applications, the most popular discretization technique is based on Galerkin's method, which is summarized in the following.

The first step in the formulation of the method is to construct finite dimensional approximations of the function spaces $\mathcal{S}$ and $\mathcal{V}$, denoted by $\mathcal{S}^h$ and $\mathcal{V}^h$, such that:

$$\begin{aligned} &\mathcal{V}^h \subset \mathcal{V} \\ &\mathcal{S}^h \subset \mathcal{S}. \end{aligned} \tag{2.4}$$

Then, the solution function $\tilde{\mathbf{u}}^h$ is sought:

$$\begin{aligned} &\text{Find } \tilde{\boldsymbol{u}}^h \in \mathcal{S}^h \text{ such that for all } \boldsymbol{v}^h \in \mathcal{V}^h \\ &a(\tilde{\boldsymbol{u}}^h, \boldsymbol{v}^h) = f(\boldsymbol{v}^h). \end{aligned} \tag{2.5}$$

Following Equation 2.3, if $\hat{\boldsymbol{u}} \neq 0$ anywhere along Γ_{D}, the spaces $\mathcal{S}^h$ and $\mathcal{V}^h$ cannot be equal, as the members of $\mathcal{S}$ and $\mathcal{V}$ need to satisfy the essential boundary conditions and their homogeneous counterparts, respectively. To remove this restriction, the solution function $\boldsymbol{u}^h$ can be shifted by a *given* function $\boldsymbol{g}^h \in \mathcal{S}^h$ to satisfy the essential boundary conditions, such that $\tilde{\boldsymbol{u}}^h = \boldsymbol{u}^h + \boldsymbol{g}^h$, with $\boldsymbol{u}^h \in \mathcal{V}^h$. This allows for casting Equation 2.5 into the following form:

$$\begin{aligned} &\text{Find } \boldsymbol{u}^h \in \mathcal{V}^h \text{ such that for all } \boldsymbol{v}^h \in \mathcal{V}^h \\ &a(\boldsymbol{u}^h, \boldsymbol{v}^h) = f(\boldsymbol{v}^h) - a(\boldsymbol{g}^h, \boldsymbol{v}^h). \end{aligned} \tag{2.6}$$

As $\mathcal{V}^h$ is finite dimensional, any member $v^h \in \mathcal{V}^h$ can be written as a linear combination of given functions $\boldsymbol{N} = \{N_0, N_1, ..., N_n\}$:

$$v^h = \sum_{i=0}^{n} N_i c_i, \tag{2.7}$$

where the coefficients denoted by c_i are often called *degrees of freedom*. Following Einstein's summation convention, the above equation can be equivalently written as:

$$\begin{aligned} &v^h = N_i c_i, \quad \text{and similarly,} \\ &u^h = N_j d_j. \end{aligned} \tag{2.8}$$

Collecting the c_i and d_j coefficients into the vectors $\boldsymbol{c}$ and $\boldsymbol{d}$ and substituting Equation 2.8 into Equation 2.6, the latter can be cast in the following form:

$$\begin{aligned} &\text{Find } \boldsymbol{d} \in \mathbb{R}^n \text{ such that for all } \boldsymbol{c} \in \mathbb{R}^n \\ &a(N_j d_j, N_i c_i) = f(N_i c_i). \end{aligned} \tag{2.9}$$

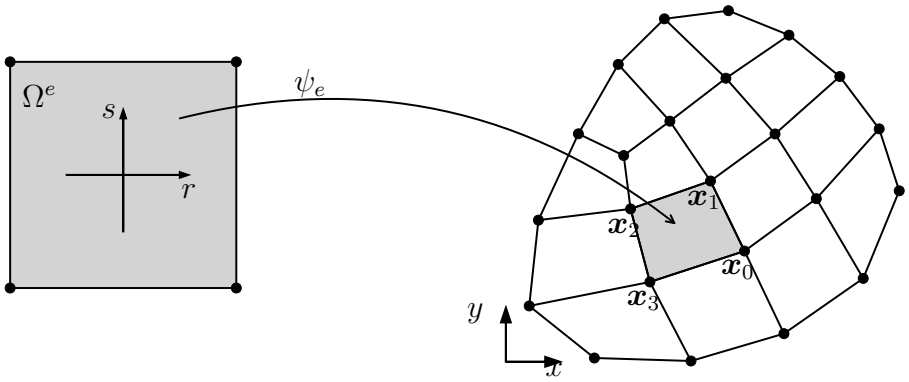


Figure 2.2: *A reference element and a two-dimensional quadrilateral mesh.*

Due to the linearity of $a(\cdot,\cdot)$ and $f(\cdot)$, the coefficients can be factored out:

$$\begin{aligned} &c_i d_j a(N_j, N_i) - c_i f(N_i) = 0 \\ &\qquad\qquad\qquad\quad \downarrow \\ &c_i \left(a(N_j, N_i) d_j - f(N_i)\right) = 0. \end{aligned} \tag{2.10}$$

As the above has to hold for any arbitrary choice of $\mathbf{c}$, the term inside the brackets has to be zero:

$$a(N_i, N_j) d_j - f(N_i) = 0. \tag{2.11}$$

Thus the unknown coefficients $\mathbf{d}$ can be found by solving the following system of linear equations:

$$\mathbf{K}\mathbf{d} = \mathbf{f}, \tag{2.12}$$

with

$$\begin{aligned} K_{ij} &= a(N_i, N_j), \quad \text{and} \\ f_i &= f(N_i). \end{aligned} \tag{2.13}$$

The matrix $\boldsymbol{K}$ and vector $\mathbf{f}$ are often called stiffness matrix and force vector in the context of computational mechanics.

2.1.2 Linear finite elements

In order to construct the space $\mathcal{V}^h$, the finite element method decomposes the domain Ω into a set of non-overlapping subdomains, called elements. The collection of elements is denoted as *mesh* and the process of subdividing the domain Ω into elements is called *mesh generation*. Typically, the elements of a finite element mesh possess simple geometric shapes, such as triangles, quadrilaterals, tetrahedra, prisms and hexahedra. While this element-wise definition is easy to implement, it allows for resolving highly complex geometric domains at the same time. Figure 2.2 depicts a two dimensional finite element mesh consisting of quadrilateral elements.

The corners of the elements are called *nodes*, the connecting lines between the nodes *edges*, and a closed loop of edges forms a *wire*. A wire encloses the face of the element, which in the 2D case forms the element domain Ω^e. On its own domain, each element uses *standard shape functions* to interpolate the unknown degrees of freedom. The simplest standard shape functions linearly interpolate between the nodes of a single element.

As an example, consider a quadrilateral element in two dimensions. The domain of the element is given in its local coordinates (r, s), with $r \in [-1, 1]$ and $s \in [-1, 1]$. Let $\boldsymbol{u}_i$ with $i = \{1, 2, 3, 4\}$ denote the unknown displacements at the four nodes of the element. Then, the displacement field internal to the domain of the element is given by:

$$\boldsymbol{u}(r, s) = \sum_{i=1}^{4} N_i^e \boldsymbol{u_i}, \tag{2.14}$$

with:

$$N_i^e(r, s) = \frac{1}{4}(1 + r_i)(1 + s_i), \tag{2.15}$$

where r_i and s_i denote the local coordinates of node i. The bijective map $\psi_e : (r, s) \rightarrow (x, y)$ depicted in Figure 2.2 establishes the coordinate transformation between the element's local domain and the physical domain Ω. Using ψ_e, the locally defined shape functions from Equation 2.14 are mapped to the global space as:

$$N_i(x, y) = N^e \circ \psi_e^{-1}. \tag{2.16}$$

Linear shape functions for three dimensional hexahedra are constructed analogously. Here, the local space of the element consists of three directions (r, s, t) with the displacement field and the shape functions defined as:

$$\begin{aligned} \boldsymbol{u}(r, s, t) &= \sum_{i=1}^{8} N_i^e \boldsymbol{u_i}, \\ N_i^e(r, s, t) &= \frac{1}{8}(1 + r_i)(1 + s_i)(1 + t_i). \end{aligned} \tag{2.17}$$

The assembly of these local functions allows for the construction of the global shape functions N_i of Equation 2.7. Let $\tilde{\boldsymbol{u}}$ denote the vector of unknown displacements within one element, e.g. in three dimensions defined as:

$$\tilde{\boldsymbol{u}} = [u_1, ...u_n, v_1, ..., v_n, w_1, ...w_n], \tag{2.18}$$

where n is the number of shape functions of the element. (For linear hexahedra, $n = 8$). Then, the displacement field within one element can be written as:

$$\boldsymbol{u} = \boldsymbol{N}\tilde{\boldsymbol{u}}, \tag{2.19}$$

where the shape functions are collected in the shape function matrix $\boldsymbol{N}$:

$$\boldsymbol{N} = \begin{bmatrix} N_1 & ... & N_n & 0 & ... & 0 & 0 & ... & 0 \\ 0 & ... & 0 & N_1 & ... & N_n & 0 & ... & 0 \\ 0 & ... & 0 & 0 & ... & 0 & N_1 & ... & N_n \end{bmatrix}. \tag{2.20}$$

Having formulated the displacement field in terms of the shape function matrix $\boldsymbol{N}$ and the degrees of freedom $\tilde{\boldsymbol{u}}$, the strain within the element can be written as:

$$\boldsymbol{\varepsilon} = \boldsymbol{B}\tilde{\boldsymbol{u}}, \tag{2.21}$$

where $\boldsymbol{B}$ is the strain-displacement operator:

$$\boldsymbol{B} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \dots & \frac{\partial N_n}{\partial x} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & \frac{\partial N_1}{\partial y} & \dots & \frac{\partial N_n}{\partial y} & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 & \frac{\partial N_1}{\partial y} & \dots & \frac{\partial N_n}{\partial y} \\ \frac{\partial N_1}{\partial y} & \dots & \frac{\partial N_n}{\partial y} & \frac{\partial N_n}{\partial x} & \dots & \frac{\partial N_n}{\partial x} & 0 & \dots & 0 \\ \frac{\partial N_1}{\partial z} & \dots & \frac{\partial N_n}{\partial z} & 0 & \dots & 0 & \frac{\partial N_n}{\partial x} & \dots & \frac{\partial N_n}{\partial x} \\ 0 & \dots & 0 & \frac{\partial N_n}{\partial z} & \dots & \frac{\partial N_n}{\partial z} & \frac{\partial N_1}{\partial y} & \dots & \frac{\partial N_n}{\partial y} \end{bmatrix}. \tag{2.22}$$

It follows that the stiffness matrix $\mathbf{K}$ and the force vector $\mathbf{f}$ is formed by assembling the individual contributions of the respective quantities of each element in the mesh:

$$\begin{aligned} \mathbf{K} &= \mathop{\mathsf{A}}_{e=0}^{n_{elem}} \mathbf{k}^e \\ \mathbf{f} &= \mathop{\mathsf{A}}_{e=0}^{n_{elem}} \mathbf{f}^e, \end{aligned} \tag{2.23}$$

where A denotes the assembly operator, $\mathbf{k}^e$ and $\mathbf{f}^e$ denote the elemental stiffness matrix and force vector associates to the individual elements. The definitions of $\mathbf{k}^e$ and $\mathbf{f}^e$ for three-dimensional linear elastostatic problems read:

$$\begin{aligned} \mathbf{k}^e &= \int_{\Omega^e} \boldsymbol{B}^\top \boldsymbol{C}\boldsymbol{B} \, \mathrm{d}\Omega^e \\ \mathbf{f}^e &= \int_{\Omega^e} \boldsymbol{N}^\top \boldsymbol{b} \, \mathrm{d}\Omega^e + \int_{\Gamma_{\mathrm{N}}^e} \boldsymbol{N}^\top \hat{\boldsymbol{t}} \, \mathrm{d}\Gamma_{\mathrm{N}}^e. \end{aligned} \tag{2.24}$$

In order to compute the derivatives of the shape functions with respect to the global coordinates (x, y, z), the chain rule is applied:

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i}{\partial r} & \frac{\partial N_i}{\partial s} & \frac{\partial N_i}{\partial t} \end{bmatrix} \boldsymbol{J}^{-1} \tag{2.25}$$

where $\boldsymbol{J}^{-1}$ is the inverse of the Jacobian of the mapping ψ_e:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix}. \tag{2.26}$$

Usually, integration is performed over the local coordinates (r, s, t) of the respective elements, using change of variables. For the element stiffness matrix $\boldsymbol{k}^e$, integration over the reference domain takes on the form:

$$\boldsymbol{k}^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^\top \boldsymbol{C}\boldsymbol{B} \, |\boldsymbol{J}| \, \mathrm{d}r\mathrm{d}s\mathrm{d}t, \tag{2.27}$$

where $|\boldsymbol{J}|$ denotes the determinant of the Jacobian matrix.

2.1.3 The p-Version of the FEM

The 2D and 3D linear shape functions of Equations 2.15 and 2.17 are one of the most frequent bases that are applied in practice. However, the approximation power of the Finite Element Method may be drastically increased if higher order polynomials are used. Especially in regions where the solution is expected to be smooth, elevating the order of the elements provides an economical alternative to mesh refinement for improved solution accuracy. A popular choice is to use Lagrange polynomials as basis functions, see e.g. in [19, 20].

Another possibility is to use higher order hierarchical basis functions. For the one-dimensional case, the bases for an order p element are defined as [21, 22]:

$$\begin{aligned} N_1(r) &= \frac{1}{2}(1-r) \\ N_2(r) &= \frac{1}{2}(1+r) \\ N_i(r) &= \Phi_{i-1}(r), \qquad i = 3...p-1. \end{aligned} \tag{2.28}$$

Here the first two functions are the standard linear bases introduced in Equations 2.15 and 2.17, while the function $\Phi_i(r)$ is defined as:

$$\Phi_i(\xi) = \sqrt{\frac{2i-1}{2}} \int_{-1}^{\xi} L_{j-1}(x)\, \mathrm{d}x = \frac{1}{\sqrt{4j-2}} \left(L_j(\xi) - L_{j-2}(\xi) \right), \tag{2.29}$$

where the $L_j(\xi)$'s are the Legendre polynomials, generated by the recursion formula:

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x), \tag{2.30}$$

with

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x. \end{aligned} \tag{2.31}$$

Figure 2.3 depicts hierarchical bases for $p = 2$ and $p = 3$.

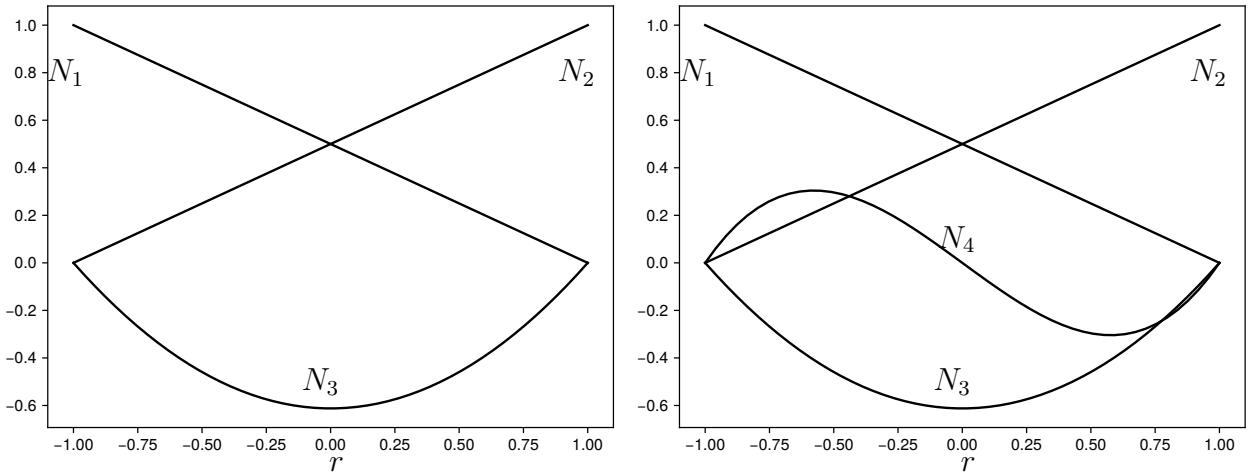
The basis functions for quadrilateral and hexahedra are constructed by taking the tensor product of the 1D basis functions. For quadrilaterals, this gives rise to the following groups of shape functions (Figure 2.5):

- *Nodal modes* are the standard bilinear basis functions introduced in Equation 2.15. Following the numbering convention for quadrilaterals on Figure 2.4, the nodal mode associated to node 1 is:

$$N^{n_1}(r,s) = \frac{1}{2}(1-r)(1-s) \tag{2.32}$$

- *Edge modes* are non-zero on exactly one edge and vanish on all other edges. The i-th edge mode associated to edge 1 is:

$$N_i^{e_1}(r,s) = \frac{1}{2}(1-s)\Phi_i(r) \tag{2.33}$$



(a) $p = 2$ (b) $p = 3$

Figure 2.3: *Examples of hierarchical basis functions with* $p = 2$ *and* $p = 3$.

- *Face modes* are zero at the nodes and edges of the element, and are non-zero only on the internal parts.

$$N_{i,j}^{\text{int}} = \Phi_i(r)\Phi_j(s) \tag{2.34}$$

For hexahedral elements, the same concept of modes apply. In this case, one additional group of modes appear: *internal modes*, i.e. shape functions that are non-zero only inside the volume enclosed by the element.

2.1.4 Coordinate transformations, the blending function method

The function Ψ_e associated to each element in the mesh defines the mapping between the local coordinates (r, s) of the element and physical coordinates (x, y). For standard, linear quadrilateral finite elements, this mapping is constructed by bi-linearly interpolating the positions of the nodes of the element:

$$\Psi_e^{LIN} = \sum_{i=1}^{4} N_i \boldsymbol{x}_i, \tag{2.35}$$

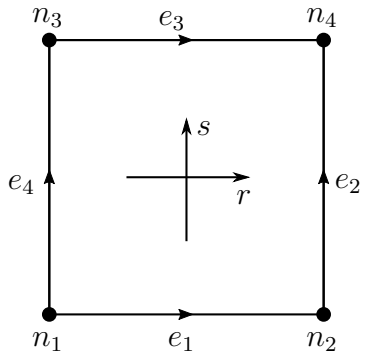
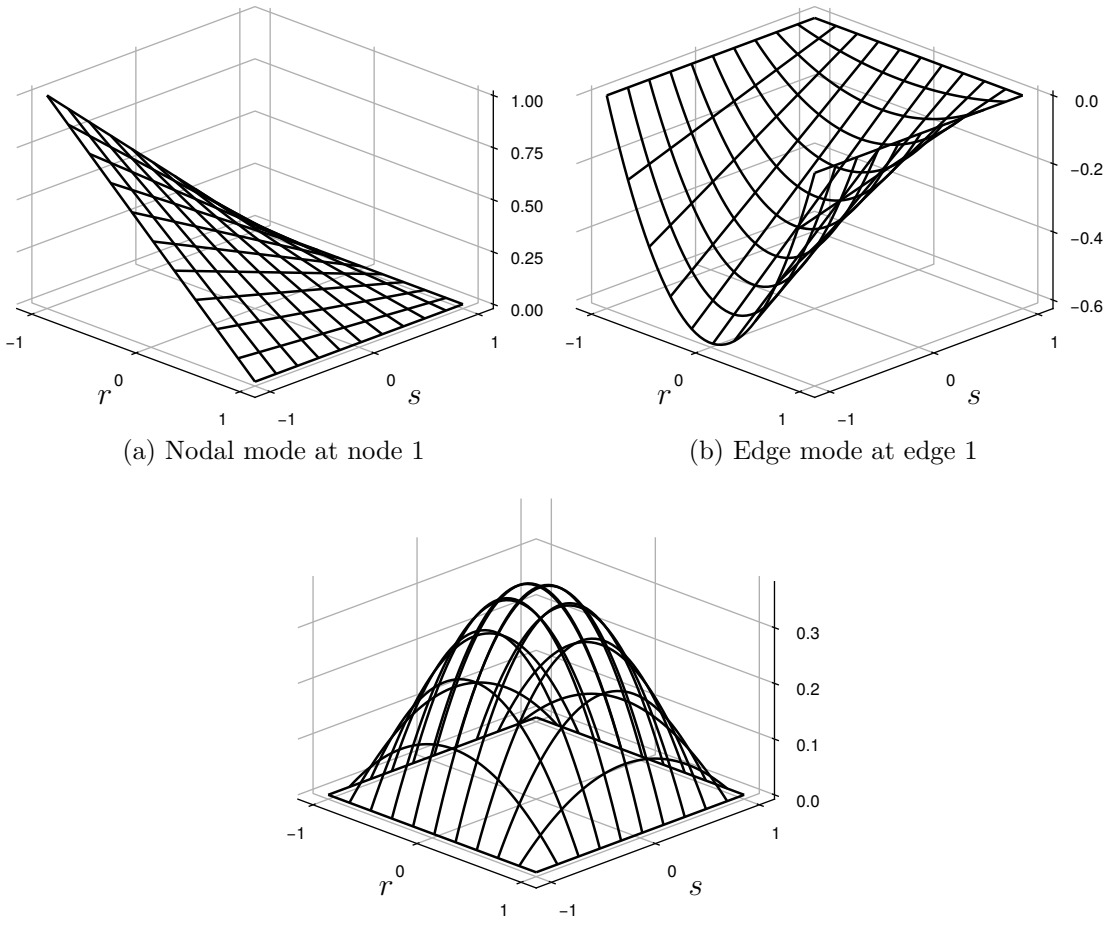


Figure 2.4: *Node and edge numberings of a quadrilateral element.*



(a) Nodal mode at node 1

(b) Edge mode at edge 1

(c) Face mode $N^{\text{int}}_{2,2} = \Phi_2(r)\Phi_2(s)$

Figure 2.5: *Examples of shape functions for a quadrilateral element*

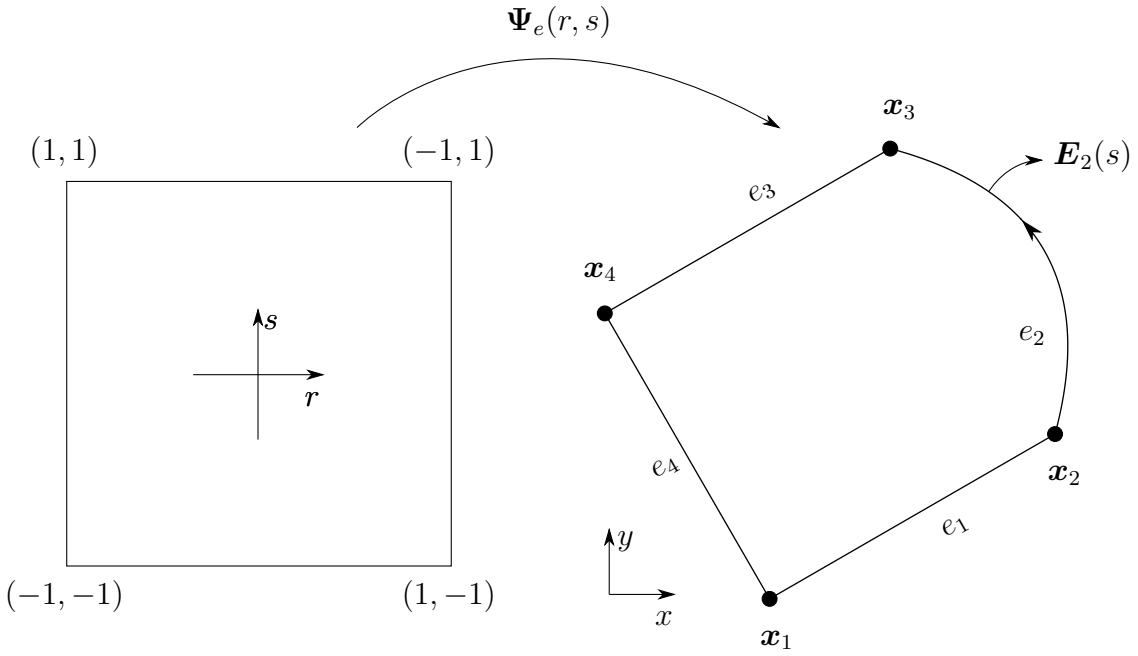


Figure 2.6: *Blended quadrilateral with one curved side*

where the $\boldsymbol{x}_i$'s represent the corner nodes (Figure 2.2).

Clearly, if the boundaries of the physical domain (denoted by $\partial\Omega$) are curved, elements that use this form of mapping can only represent $\partial\Omega$ in an *approximate* sense. While the error of this geometric approximation may be reduced by decreasing the element size (i.e. by *mesh refinement*), the increase in the number of elements will also increase the size of the equation system to be solved. For p-FEM, an accurate and potentially high-order geometric representation becomes even more important in order to fully leverage the approximation power lying in the high-order discretization of the solution. One method that enables Ψ_e to represent curved boundaries in an exact sense is the *blending function method*, introduced in [23].

To demonstrate the method, consider the quadrilateral element with one curved edge e_2, depicted in Figure 2.6. Let $\boldsymbol{E}_2(s) : [-1,1] \to \mathcal{R}^2$ denote the parametric mapping of the curve associated to the edge, with $\boldsymbol{E}_2(-1) = \boldsymbol{x}_2$ and $\boldsymbol{E}_2(1) = \boldsymbol{x}_3$, i.e. the endpoints of the curve coincide with the respective nodes of the element. Then, the mapping $\Psi_e(r,s) : [-1,1]^2 \to \mathcal{R}^2$ between the local space of the element and the physical space is:

$$\begin{aligned}\Psi_e(r,s) = \frac{1}{4}\Big[&(1-r)(1-s)\boldsymbol{x}_1 + (1+r)(1-s)\boldsymbol{x}_2 \\ &+(1+r)(1+s)\boldsymbol{x}_3 + (1-r)(1+s)\boldsymbol{x}_4\Big] + \boldsymbol{f}_2(s),\end{aligned} \tag{2.36}$$

with

$$\boldsymbol{f}_2(s) = \frac{1}{2}(1+r)\left[\boldsymbol{E}_2(s) - \frac{1}{2}\left((1-s)\boldsymbol{x}_2 + (1+s)\boldsymbol{x}_3\right)\right]. \tag{2.37}$$

The term $\boldsymbol{f}_2(s)$ is the difference between the curve $\boldsymbol{E}_2$ and the straight line $\boldsymbol{x}_2 \to \boldsymbol{x}_3$, multiplied with a linear term that is one at $r = 1$ and vanishes at $r = -1$. This concept can be extended to

quadrilaterals where all the sides are curved. After adding the terms for the three remaining edges to Equation 2.36 and rearranging, the complete definition of a blended quadrilateral reads:

$$\begin{aligned}\Psi_e(r,s) = \frac{1}{2}\Big[&(1-s)\boldsymbol{E}_1(r) + (1+r)\boldsymbol{E}_2(s) \\ &+(1+s)\boldsymbol{E}_3(r) + (1-r)\boldsymbol{E}_4(s)\Big] - \sum_{i=1}^{4} N_i \boldsymbol{x}_i.\end{aligned} \tag{2.38}$$

2.1.5 NURBS representation of parametric curves and surfaces

The definition of the blending function interpolation from Equation 2.38 requires the bounding curves to be represented in parametric form. An especially popular parametric representation for curves is given by *non-uniform rational B-splines*, shortly NURBS. An in-depth discussion of NURBS curves and surfaces is found in [24]. A NURBS curve is defined by the following:

- A polynomial order p.
- A vector of m ascending real numbers $U = [u_1, u_2, ..., u_m]$, with $u_i \in \mathcal{R}$, called the *knot vector*.
- A vector of n control points $\boldsymbol{\pi} = [\boldsymbol{P}_1, \boldsymbol{P}_1, ..., \boldsymbol{P}_n]$, with $\boldsymbol{P}_i \in \mathcal{R}^{n_d}$. Here, n_d is the dimensionality of the embedding space.
- A vector of n weights for the control points $\boldsymbol{w} = [w_1, w_2, ..., w_n]$, with $w_i \in \mathcal{R}$. One weight is associated to each control point.

The knot vector U together with a polynomial order p defines $n = m - p - 1$ basis functions of order p, with the following recursive definition:

$$\begin{aligned}B_{i,0}(u) &= \begin{cases} 1 & \text{if} \quad u \in]u_i, u_{i+1}] \\ 0 & \text{otherwise,} \end{cases} \\ B_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(u).\end{aligned} \tag{2.39}$$

An example with $p = 2$ and $U = \left[0, 0, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, 1, 1\right]$ is depicted in Figure 2.7. The bases possess the following important properties:

- A basis function $N_{i,p}$ is non-zero only in the interval $[u_i, u_{i+p+1}]$.
- At any point in the interval $[u_1, u_m]$, the sum of the non zero basis functions is one, also known as the *partition of unity* property.
- Inside a knot span $[u_i, u_{i+1}]$, the basis functions are C^∞ continuous.
- If a knot u_i in the vector U is repeated k times (i.e. it has a *multiplicity* of k), the basis functions at u_i are C^{p-k} continuous.

Now, a B-Spline curve can be constructed by a linear combination of the control points and the basis functions:

$$\boldsymbol{C}(u) = \sum_{i=1}^{n} B_{i,p}(u) \boldsymbol{P}_i. \tag{2.40}$$

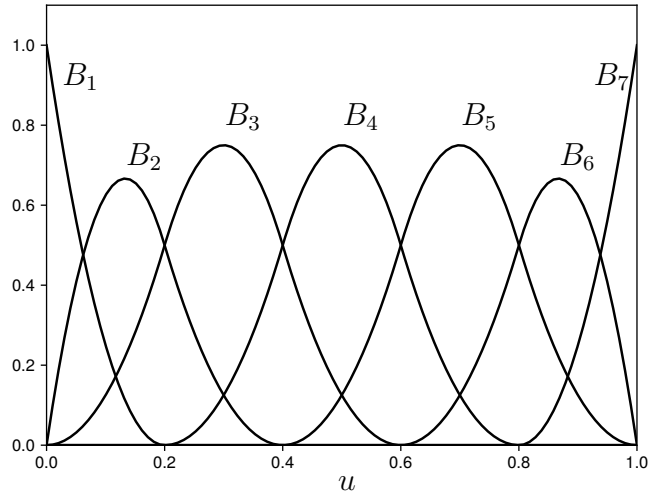


Figure 2.7: *B-Spline basis functions of order* $p = 2$ *with knot vector* $U = [0, 0, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, 1, 1]$.

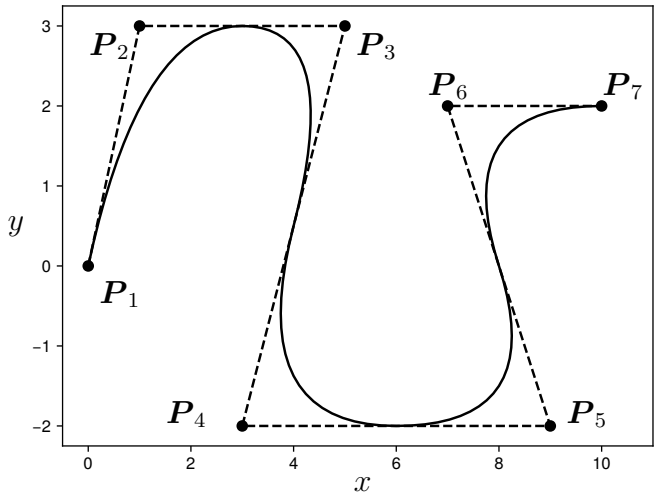


Figure 2.8: *A B-Spline curve and its control points with* $p = 2$.

Figure 2.8 depicts an example of a B-Spline curve of order $p = 2$, knot vector $U = [0, 0, 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, 1, 1]$ and the following control points:

$$\begin{aligned}
\boldsymbol{P}_1 &= \begin{bmatrix} 0 & 0 \end{bmatrix}^\top \\
\boldsymbol{P}_2 &= \begin{bmatrix} 1 & 3 \end{bmatrix}^\top \\
\boldsymbol{P}_3 &= \begin{bmatrix} 5 & 3 \end{bmatrix}^\top \\
\boldsymbol{P}_4 &= \begin{bmatrix} 3 & -2 \end{bmatrix}^\top \\
\boldsymbol{P}_5 &= \begin{bmatrix} 9 & 2 \end{bmatrix}^\top \\
\boldsymbol{P}_6 &= \begin{bmatrix} 7 & 2 \end{bmatrix}^\top \\
\boldsymbol{P}_7 &= \begin{bmatrix} 10 & 2 \end{bmatrix}^\top .
\end{aligned} \tag{2.41}$$

Even though B-Spline curves provide enough flexibility to model a broad set of curves that

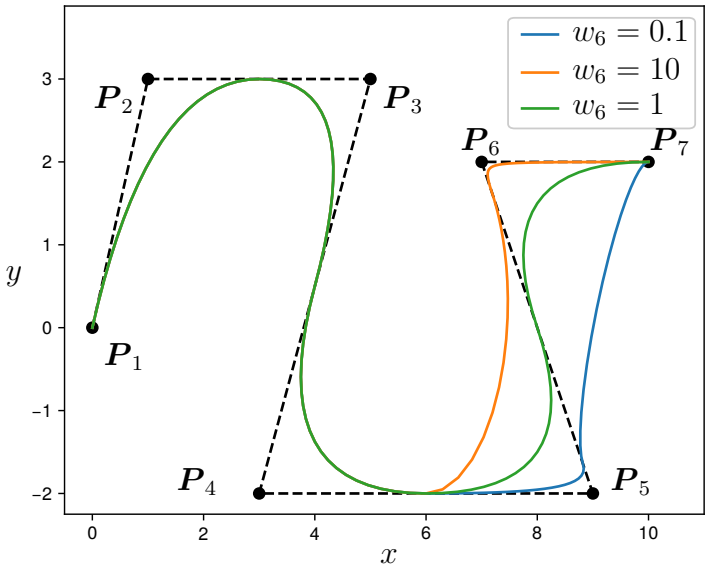


Figure 2.9: *NURBS curves with $p = 2$ and a varying weight for $\boldsymbol{P}_6$.*

may appear in engineering practice, they are not capable of providing an exact representation of *all* geometric shapes. For example, a class of curves that cannot be represented by B-Splines are conic sections: circles, ellipses or hyperbolas. In order to be able to represent these shapes *exactly*, the definition in Equation 2.40 is extended by assigning a weighting factor w_i to each control point on the curve. Then, Equation 2.40 is modified to:

$$\boldsymbol{C}(u) = \frac{\sum_{i=1}^{n} B_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=1}^{n} B_{i,p}(u) w_i}, \tag{2.42}$$

which is a rational curve, hence the name non-uniform *rational* B-spline. Increasing the weighting factor w_i associated to a single control point will cause the curve to be pulled toward $\boldsymbol{P}_i$ on the interval $[u_i, u_{i+p+1}]$, or conversely, decreasing w_i will push the curve away from $\boldsymbol{P}_i$. Figure 2.9 depicts a NURBS with control points and knot vector as defined in 2.41, and a weight vector where all the weights are unity except for w_6, which is varied in the range 0.1...10.

Apart from curves, NURBS are able to represent higher dimensional manifolds as well. Surfaces, for example, can be constructed by taking the tensor product of one dimensional bases. For this case, a surface is defined by:

- Two polynomial orders p_u and p_v for the basis functions along the two parametric directions u and v.
- Two knot vectors U and V for the basis functions along the two parametric directions u and v.
- A grid of control points $\boldsymbol{P}_{ij}$, where $i = 1...n$, $j = 1...m$, with n and m being the number of basis functions along the directions u and v, respectively.
- A grid of weights w_{ij}.

Then, the NURBS surface is defined as:

$$\boldsymbol{S}(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p_u}(u)N_{j,p_v}(v)w_{ij}\boldsymbol{P}_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p_u}(u)N_{j,p_v}(v)w_{ij}}. \tag{2.43}$$

2.1.6 Numerical quadrature

In general, integrating the expression of the element stiffness matrix in Equation 2.27 is not possible analytically. Instead, the finite element method employs numerical quadrature rules for integration. The most widely used rule for this purpose is the Gauss-Legendre (GL) quadrature. Given a function $f(r) : I \to \mathcal{R}$ to be integrated on the domain $I = [-1, 1] \subset \mathcal{R}$, the GL rule transforms the integral into a weighted sum of function evaluations at specified locations:

$$\int_{-1}^{1} f(r)\mathrm{d}r \approx \sum_{i=1}^{n} f(r_i)w_i, \tag{2.44}$$

where the r_i-s are the function evaluation locations (abscissas) dictated by the quadrature rule, and the w_i-s are the associated weights. A quadrature rule of Gauss-Legendre type with n abscissas is capable of exactly integrating a polynomial of degree $2n - 1$.

Two- or three-dimensional integrals can be evaluated by applying the quadrature rule for the respective dimensions in a nested manner:

$$\begin{aligned}
\int_{-1}^{1}\int_{-1}^{1} f(r,s)\ \mathrm{d}r\mathrm{d}s &= \sum_{i=1}^{n}\sum_{j=1}^{m} f(r_i,s_j)w_iw_j \\
\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(r,s,t)\ \mathrm{d}r\mathrm{d}s\mathrm{d}t &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} f(r_i,s_j,t_k)w_iw_jw_k.
\end{aligned} \tag{2.45}$$

Here, the function $f(...)$ is evaluated $n \times m$, or $n \times m \times l$ times, for the 2D and 3D cases, respectively. It is noted here that the number of quadrature points in each coordinate direction need not to be equal. This is especially true for the p-version of the Finite Element Method, where the order of shape functions may be chosen differently along the local coordinate directions of the respective elements. These anisotropic discretizations benefit from a quadrature rule with a varying number of points, as the total number of function evaluations can be decreased, while the error of the integration can be kept minimal. In this work, only the case with equal number of points per direction is considered.

2.2 The Finite Cell Method

One of the main steps of conducting a finite element simulation is the partitioning of the domain of interest Ω_{phy} into elements, called mesh generation. Mesh generation is a challenging

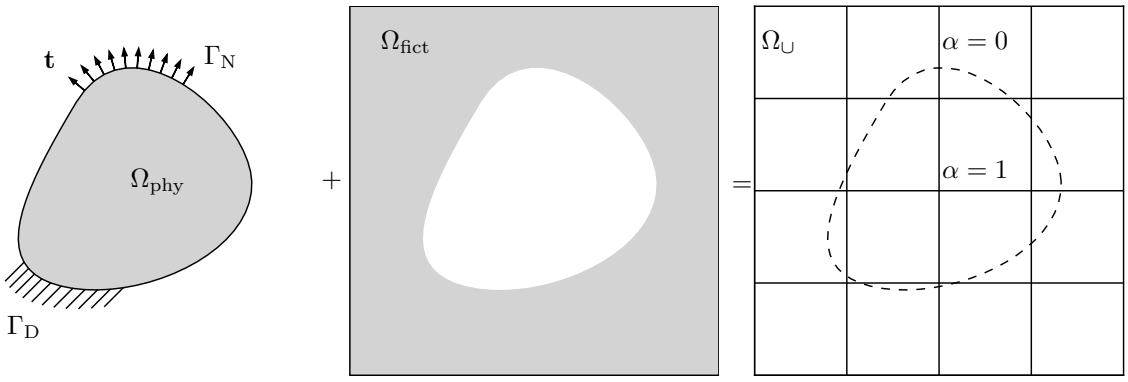


Figure 2.10: *The core concept of the FCM. The physical domain Ω_{phy} is extended by the fictitious domain Ω_{fict}. Their union, the embedding domain $\Omega_\cup$ can be meshed easily. The influence of the fictitious domain is penalized by the scaling factor α [4].*

task, with at least as much research invested in understanding the problems and solutions regarding the process as there is research conducted on the finite element method itself. Some good resources about mesh generation are e.g. [25–27]. A mesh generator for high-order FEM, where the resulting elements utilize the blending function interpolation can be found in [28].

Despite the vast choice of methods available, meshing remains a challenging task even today. The preparation of the geometric model and the subsequent step of mesh generation may account for as much as 80% of the complete analysis time [1]. Thus, recent years' research interest in the computational mechanics community has turned towards alternative approaches which aim at avoiding this difficult (and thus expensive) step. One of these methods is the Finite Cell Method (FCM) a high-order immersed boundary technique, first introduced in [29].

The FCM extends the boundaries of the domain of interest Ω_{phy} by a so-called fictitious domain Ω_{fict} such that the union of the two forms an embedding domain $\Omega_\cup$. Usually, the embedding domain is formed such that it coincides with the bounding box of the original physical domain. Such box-like geometries can be meshed very easily into a regular grid of rectangular (or right-angled hexahedral) finite elements. To remain consistent with the original problem, the material properties in the fictitious domain are multiplied with a scaling factor α, chosen in the range of $\alpha = 10^{-12}...10^{-6}$. The weak stiffness introduced by α ensures that the strain energy in the fictitious domain remains very close to zero, therefore it does not have a significant contribution to the overall solution. The solution field found in Ω_{phy} thus remains a very good approximation to the solution of the original problem.

In the following subsections, the formulation of the method is outlined, as well as a brief overview of the challenges that might be encountered when implementing the FCM. A detailed discussion of the matter may be found in [22, 30].

2.2.1 Formulation

Figure 2.10 illustrates the core idea of the FCM. The boundaries of the physical domain of interest Ω_{phy} are extended by a fictitious part Ω_{fict}. Their union $\Omega_{\text{phy}} \cup \Omega_{\text{fict}}$ forms the embedding domain $\Omega_\cup$. As $\Omega_\cup$ possesses a simple, box-like geometry, it can easily be meshed into a structured grid of rectangular finite elements in 2D and cuboids in 3D. To extinguish the influence of Ω_{fict}, the material parameters are multiplied with an indicator function $\alpha(\boldsymbol{x})$:

$\mathcal{R}^{n_d} \to \mathcal{R}$, defined as:

$$\alpha(\boldsymbol{x}) = \begin{cases} 1 & \forall \boldsymbol{x} \in \Omega_{phy} \\ 10^{-q} & \forall \boldsymbol{x} \in \Omega_{fict}. \end{cases} \tag{2.46}$$

Then, the weak form of Equation 2.2 is written as:

$$\begin{aligned} &\text{Given } \boldsymbol{b} : \Omega_{\cup} \to \mathbb{R}^{n_d}, \hat{\boldsymbol{t}} : \Gamma_{\mathrm{N}} \to \mathbb{R}^{n_d}, \hat{\boldsymbol{u}} : \Gamma_{\mathrm{D}} \to \mathbb{R}^{n_d}, \text{ find } \boldsymbol{u} \in \mathcal{S} \\ &\text{such that for all } \boldsymbol{v} \in \mathcal{V} \\ &\qquad\qquad a(\boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{v}), \\ &\text{with} \quad a(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega_{\cup}} \boldsymbol{\epsilon}(\boldsymbol{v}) : \alpha(\boldsymbol{x}) \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \, \mathrm{d}\Omega \\ &\text{and} \quad f(\boldsymbol{v}) = \int_{\Omega_{\cup}} \boldsymbol{v} \cdot \alpha(\boldsymbol{x}) \boldsymbol{b} \, \mathrm{d}\Omega + \int_{\Gamma_{\mathrm{N}}} \boldsymbol{v} \cdot \hat{\boldsymbol{t}} \, \mathrm{d}\Gamma. \end{aligned} \tag{2.47}$$

This weak formulation is the auxiliary problem to the classical weak form of Equation 2.2. To bring Equation 2.47 into a discrete form, the same methodology is applied as for standard finite elements outlined in Section 2.1.2. In this process, shape functions are assigned to the quadrilateral (or cuboidal) elements of the regular mesh generated on $\Omega_{\cup}$. As the boundaries of these elements do not necessarily coincide with the geometric boundaries of the physical domain Ω_{phy}, the elements of the background mesh are referred to as *cells*, in order to distinguish them from standard finite elements.

So far, various choices of shape functions in combination with the FCM have been investigated. In the original version of the method, integrated Legendre polynomials were employed, showing exponential rates of convergence [29, 31]. Instead of conventional C^0 bases, an alternative is to use B-Spline basis functions [32, 33], where the "cells" in the grid are defined by the knot spans of the respective bases.

Because the boundaries of the finite cell mesh do not coincide with the boundaries of the original physical domain, implementing the finite cell method poses a set of challenges that need to be addressed:

1. *Boundary conditions*
   Homogeneous Neumann boundary conditions are automatically satisfied by setting α to zero or to a very small value in Ω_{fict}. Non-homogeneous Neumann boundary conditions can be applied by evaluating the contour integral of Equation 2.47 over Γ_{N}. However, in contrast to standard finite elements, it is generally not possible to realize Dirichlet boundary conditions by directly constraining the degrees of freedom associated to the boundaries of the element, since the elements do not coincide with the boundaries of the physical domain. Therefore, these essential boundary conditions are applied in the weak sense, i.e. by using the penalty method [34], Lagrange multipliers [35], or Nitsche's method [36].
2. *Local refinement*
   Localized phenomena in the governing physics of the problem requires the refinement of the discretization. This is the case for singular problems due to re-entrant corners, material interfaces, or highly localized physical processes. The inherent problem of mesh

refinement is the need to ensure compatibility between element boundaries: without further treatment, no hanging nodes and edges are allowed to appear in the mesh. Traditional methods employ refinement templates that aim at refining the mesh towards singular points in 2D, however, their implementation is quite delicate in three dimensions, especially when the aim is to resolve edge singularities. To avoid the challenges associated to refinements and hanging nodes, a popular approach is to use a hierarchy of recursively refined overlay meshes, see in [37, 38] and [39].

3. *Conditioning of the linear system of equations*
   Finite cells that are barely cut by the domain boundary and contain only a small portion of Ω_{phy} will lead to a high condition number of the resulting system of equations. This causes slower convergence when using an iterative solver, which would be essential for solving larger systems. Increasing the value of α may help in stabilizing the system, but it comes at the cost of introducing a higher modeling error in the finite cell approximation. Various techniques exist to overcome this problem, the most promising one being the *Symmetric Incomplete Permuted Inverse Cholesky* (SIPIC) preconditioner introduced in [40].
4. *Integration of cut cells*
   In those cells that are cut by the boundary, the scaling factor $\alpha(\boldsymbol{x})$ introduces a discontinuity in the integrands of the element stiffness matrices. Standard integration schemes employed in finite elements are not able to deal with such discontinuities. Therefore, the FCM needs to employ specially constructed quadrature rules to achieve optimal convergence rates. In the remaining parts of this chapter, possible solution strategies for dealing with discontinuous integrands are outlined.

2.2.2 Quadrature schemes

Due to the discontinuity that is introduced by penalizing the constitutive matrix $\boldsymbol{C}$, the Gaussian quadrature loses its accuracy in the cut cells, leading to sub-optimal convergence rates [29, 31, 41]. To overcome this problem, various alternative methods have been suggested in the literature. The following sections summarize these approaches.

2.2.2.1 Spacetrees

The most widely used method to improve the accuracy of the numerical integration uses a composed Gaussian quadrature rule combined with a spacetree subdivision of the cells that are cut by the domain boundaries. In two dimensions, this means that every cut cell is recursively subdivided into 4 equal integration subcells until a predefined depth k is reached (Figure 2.11). This way, the resulting set of integration subcells – the integration mesh – is adaptively refined towards the interface. Consequently, this subdivision method localizes the discontinuity to the smallest leaves of the spacetree.

In the three-dimensional case, the octree algorithm bisects each cut cell into eight octants. In this procedure, 19 new internal nodes are created, which can be categorized as follows (refer to Figure 2.12):

- By bisecting every edge of the cell, twelve *edge nodes* are created.
- Similarly to the edge nodes, six *face nodes* are created in the center of each face.

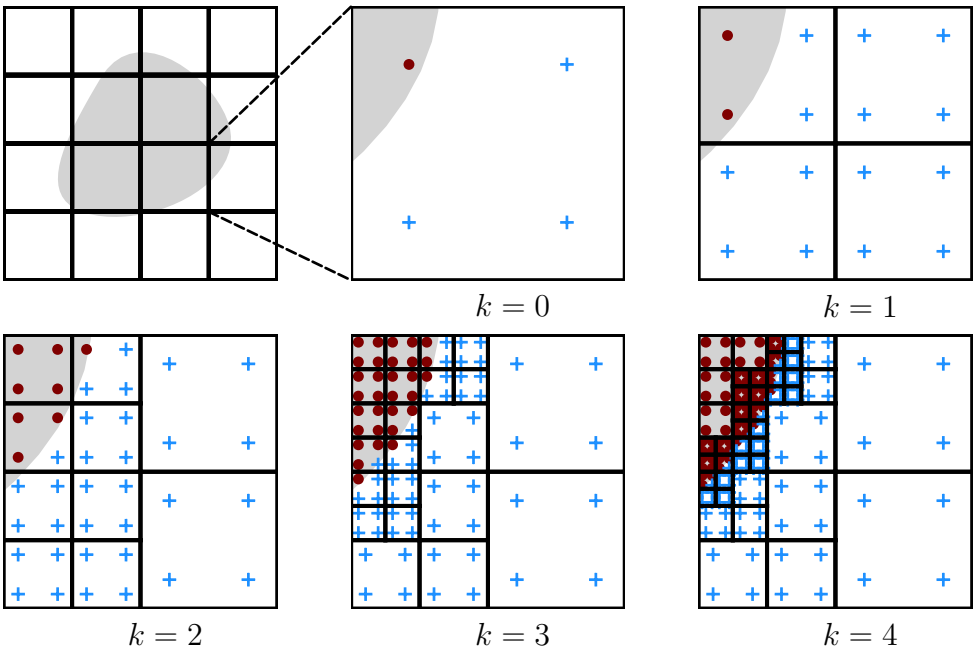


Figure 2.11: *Spacetree-based integration domains for different values of maximum subdivision depth k. Red dots represent integration points that lie in Ω_{phy}, blue crosses are in Ω_{fict} [4].*

- A *mid-node* is created in the center of the cell.

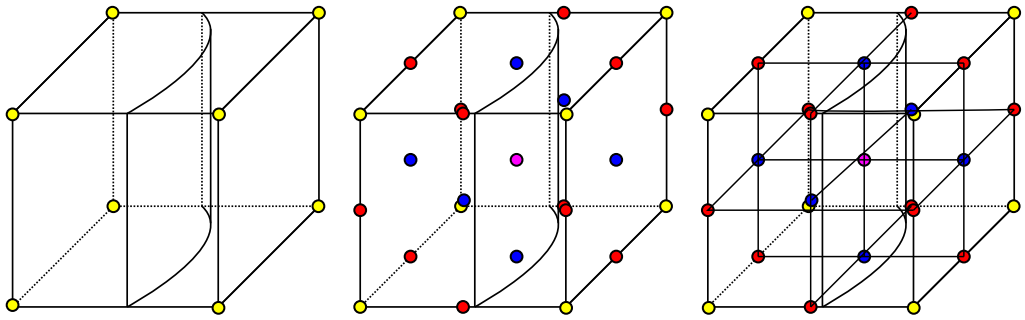


Figure 2.12: *Octree generation: The edge nodes (red), face nodes (blue) and the mid-node (purple) are connected to each other, forming eight octants [17].*

Once the spacetree is constructed, the quadrature points are distributed in the parameter space of each resulting integration cell and then mapped to the parameter space of the finite cell. Then, the Jacobian term in Equation 2.27 is the product of the mappings $\boldsymbol{\psi}(r, s)$ and $\tilde{\boldsymbol{\psi}}(\tilde{r}, \tilde{s})$, where the terms with $\sim$ denote the local coordinates of the integration cell and the mapping from the parameter space of the integration cell to the parameter space of the finite cell. The concept of the mappings is depicted in Figure 2.13.

The only information that spacetree methods need to extract from the geometric model is whether a cell is cut by the interface. This can be achieved by performing inside-outside tests on dedicated seed points, distributed on the domain of the cell. Because this information can be extracted virtually from any geometric model, the octree approach can be applied to a wide set of applications, such as voxel-based domains [42], implicit geometries [43] and BREP

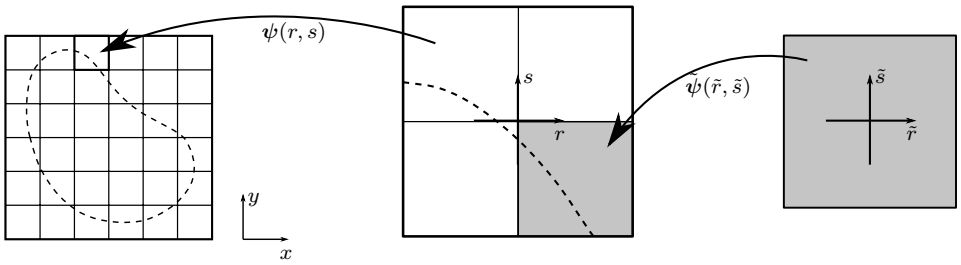


Figure 2.13: *The concept of mappings for quadtree depth $k = 1$ [16].*

models [44]. The clear advantage of spacetree methods lies in their robustness, because–regardless of how the interface intersects the cell–the subcells are always created as explained before.

However, the disadvantage of the spacetree-based approaches is that they are not necessarily accurate and efficient at the same time. Because they leave the more detailed information provided by the geometric model unexploited, the accurate integration may demand many levels of subdivision. This can lead to an excessive number of integration cells, making the method less efficient.

2.2.2.2 Moment fitting-based schemes

Quadtrees and octrees manipulate the integration domains by decomposition and leave the integration rule unchanged on the leaves of the integration tree. It is also possible to keep the integration domain unchanged and modify the quadrature rule such that it is "tailored" to each cut cell. An example for this idea is the moment fitting method, introduced for implicit geometries in [45] and applied to the finite cell method in [46, 47].

The basic idea of moment fitting is to find a set of integration points $\{\boldsymbol{r}_i\}$ and associated weights $\{w_i\}$ such that they exactly integrate the high-order polynomials that are the entries in the element stiffness matrices. The points and weights can be found by solving:

$$\sum_{i=1}^{n} f_j(\boldsymbol{r}_i) w_i = \int_{\Omega^e} f_j(\boldsymbol{r}) \mathrm{d}\Omega, \tag{2.48}$$

where the f_j's are m independent basis functions. This can be written in matrix form:

$$\begin{bmatrix} f_1(\boldsymbol{r_1}) & \cdots & f_1(\boldsymbol{r_n}) \\ \vdots & \ddots & \vdots \\ f_m(\boldsymbol{r_1}) & \cdots & f_m(\boldsymbol{r_n}) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_{\Omega^e} f_1(\boldsymbol{r}) \mathrm{d}\Omega \\ \vdots \\ \int_{\Omega^e} f_n(\boldsymbol{r}) \mathrm{d}\Omega \end{bmatrix}. \tag{2.49}$$

To set up this equation system, the volume integrals on the right hand side need to be evaluated. While one possibility is to apply spacetree decomposition, the other option is to use the divergence theorem and turn integration over volumes into integration over contours, see in [45]. The advantage of this lies therein that the contours required for integration are usually readily available for BREP or STL models.

The difficulty that arises with moment fitting is that Equation 2.49 is non-linear with respect to the integration point positions $\boldsymbol{r}_i$. This property makes the method less appealing for

computing quadrature rules for the FCM, especially if the finite cell mesh contains many cut cells. The non-linearity of the problem can be removed by fixing the quadrature point locations, and only computing the unknown weights. Even with this modification, every cut cell practically requires the solution of an equation system, rendering the method relatively expensive when applied on a large number of cut cells.

A modification presented in [48] resolves this issue by using Lagrange polynomials for the basis functions:

$$f_j(r) = l_j(r) = \prod_{\substack{k=1 \\ k \neq j}}^{p_d} \frac{r - r_k^{\mathrm{GL}}}{r_j^{\mathrm{GL}} - r_k^{\mathrm{GL}}}, \tag{2.50}$$

where p_d is the order of quadrature and r_k^{GL} is the position of the k-th quadrature point from the standard Gauss-Legendre quadrature rule. It follows from the Kronecker delta property of Lagrange polynomials that if the positions of the quadrature points are fixed to lie exactly at the r_k's, the matrix in Equation 2.49 becomes diagonal, and the weights can be computed directly:

$$w_i = \int_{\Omega^{\mathrm{e}}} l_j(\boldsymbol{r}) \mathrm{d}\Omega. \tag{2.51}$$

Due to these modifications, the need to solve the linear system of equations in Equation 2.49 can be completely avoided.

2.2.2.3 Conforming integration meshes in 2D[a]

A different way to avoid the problems associated to spacetree subdivisions is to decompose the cut cells into subcells with boundaries that coincide with the interface $\partial\Omega$, resulting in a boundary-conforming integration mesh. This approach has attracted an increasing interest in recent years, not only in the context of FCM, but also in other areas of computational mechanics, such as eXtended Finite Elements, isogeometric analysis of trimmed surfaces, or meshless methods. In contrast to the spacetree-based integration, subcells provided by these methods account for the more detailed information available from the geometric model.

The main challenge of boundary-conforming subdivisions is the high number of possible intersection patterns which can appear in cut elements. For two-dimensional applications various algorithmic approaches have been developed to deal with this problem robustly and efficiently. For XFEM, a suitable method is described in [49]. Here, the cut cells are subdivided into non-overlapping quadrilaterals and triangles—and subcell mappings are defined using Lagrangian shape functions, leading to a better approximation of the boundary. Also in the context of XFEM, [50] presents a method that delivers triangles and mappings known from Nurbs-Enhanced Finite Elements (NEFEM [51]). For the FCM, a particularly effective decomposition algorithm was presented in [16], which is described in the following.

The approach is based on the observation that if a cut cell is divided into two halves along one of its diagonals, the two resulting triangles can always be decomposed topologically into a triangle and a quadrilateral. This idea is depicted in Figure 2.14. To identify how the diagonal line has to be created, the algorithm performs an inside-outside test on the four corner vertices

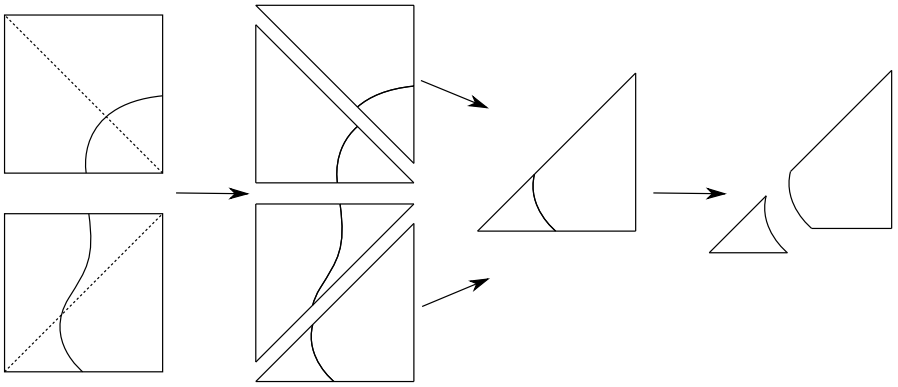


Figure 2.14: *The general flow of the proposed algorithm. Every cut cell is subdivided into two triangles along a diagonal line. The triangles are further cut by the boundary into a triangle and a quadrilateral [16].*

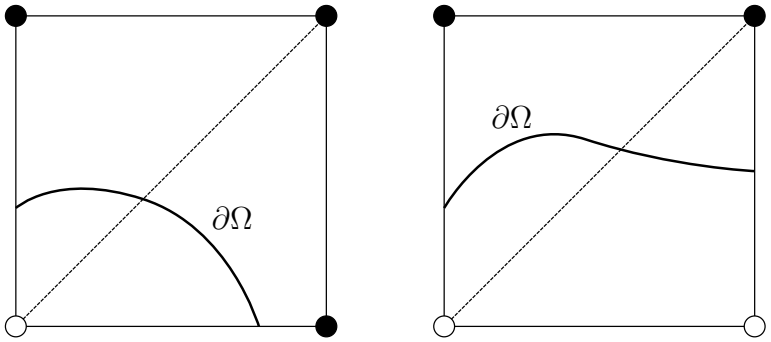


Figure 2.15: *The process of determining the orientation of the diagonal cut line (drawn in dashed style). Black and white dots represent corners lying on opposite sides of the domain boundary $\partial\Omega$. One pair of corners with opposite inside-outside is joined [16].*

of the cell. If two opposite vertices have opposite states, the diagonal line is formed by joining them, as depicted in Figure 2.15.

After the line is created, the intersection points between the boundary and the diagonal line as well as the cell edges are computed. The boundary is then trimmed at these points of intersection. Finally, the resulting trimmed curves are used to form curved triangles and quadrilaterals defined using the blending function method of Section 2.1.4.

Degenerate cuts

If a non-convex boundary cuts the cell, there can be several intersection points between this boundary and the diagonal line (Figure 2.16). Therefore, no triangular decomposition can be made. Likewise, if the boundary has no intersection with the diagonal line, there is no straightforward decomposition that matches the procedure described before (Figure 2.17). This second kind of special configuration is detected by evaluating the inside-outside state of dedicated seed points on the domain of the cell. Thus, the special case in Figure 2.17 may be missed if the resolution of these seed points is not fine enough. If a special configuration is detected, the algorithm performs a quadtree subdivision and runs the triangulated decomposition on the leaf cells of the tree. This is done recursively, until the generic case of Figure 2.14 can be constructed. This quadtree based fallback option is depicted in Figures 2.16 and 2.17.

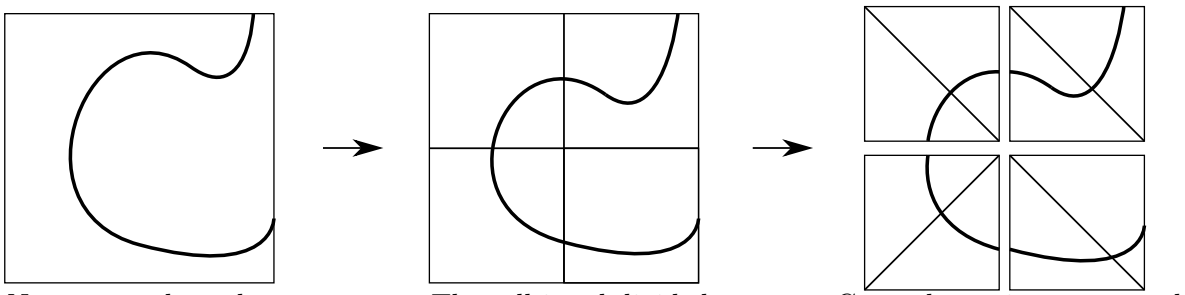


Figure 2.16: *Non-convex boundary cuts the cell*

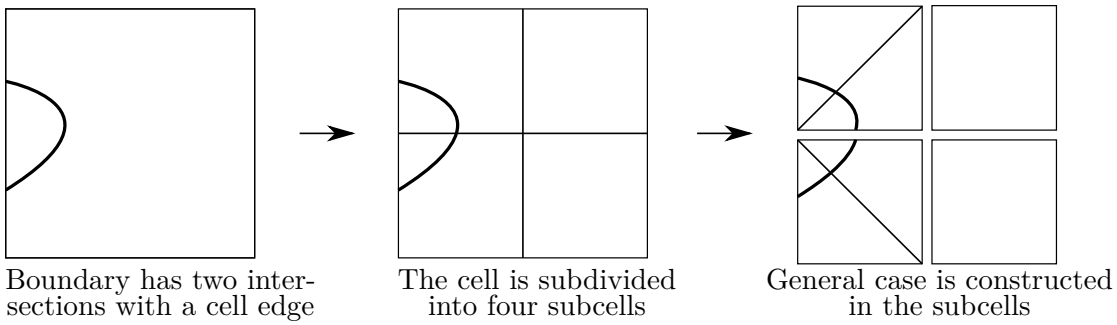


Figure 2.17: *A boundary intersects a cell edge twice [16].*

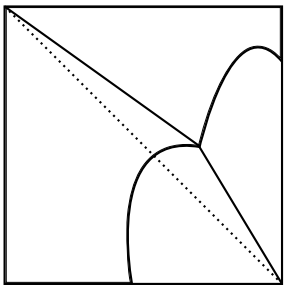


Figure 2.18: *Resolution strategy for kinks: the diagonal line is replaced by two connected segments [16].*

Kinks and corners

In many cases, the boundary of the domain is not composed of one continuous curve, but is a set of connected curve segments. These points usually represent discontinuous jumps in the curve derivatives and have to be taken into account by the decomposition algorithm in order to maintain the precision of the integration.

Therefore, if more than one curve is detected in a cell, the diagonal line is replaced by two linear segments. The point in which these segments are connected is the location of the kink in the cell. The extraction of the kink from a BREP model is a straightforward operation. The case when two curves meet at a sharp edge is depicted in Figure 2.18.

Piecewise definition of the boundary

As Equation 2.38 suggests, the nature of the parametric description of the bounding curves has a strong influence on the mapping $\tilde{\boldsymbol{\psi}}(\tilde{r}, \tilde{s})$ and thus on the Jacobian determinant of the integration cell. If any of the bounding curves of the integration cell is defined piecewise, the Jacobian determinant in the integrand of the element stiffness matrix (Equation 2.27) becomes non-smooth as well. Because the Gaussian quadrature is able to cope with polynomials but not with functions that are defined piecewise, this has to be taken into account by the decomposition algorithm.

Having piecewise defined curves as boundary description is a very frequent case, as many geometries in engineering computations use B-Splines or NURBS. The parameter space of these curves is divided into a set of subintervals and the parametric equation of the curve changes at the breakpoint between neighboring subintervals. In the context of B-Splines or NURBS, these breakpoints are often referred to as knots.

In 1D, integrating piecewise polynomials numerically is performed by employing composed integration, based on the sum of the integrals on the separate intervals the curve is defined on. In order to perform exact numerical integration, this concept has to be applied to 2D: instead of integrating subintervals, the integration has to be carried out on subregions, where the boundaries of the regions are defined by the locations of the breakpoints.

As an example, consider a blended integration cell with $e_b(s)$ being a piecewise-defined, curved boundary (Figure 2.19) that has one breakpoint at $s = -0.5$.

Because the definition of $e_b(s)$ changes at $s = -0.5$, the Jacobian determinant of the blended mapping is discontinuous, too. Thus, the integration cell has to be further subdivided along the $s = -0.5$ isoparametric line. Integration then takes place on these subcells separately, and

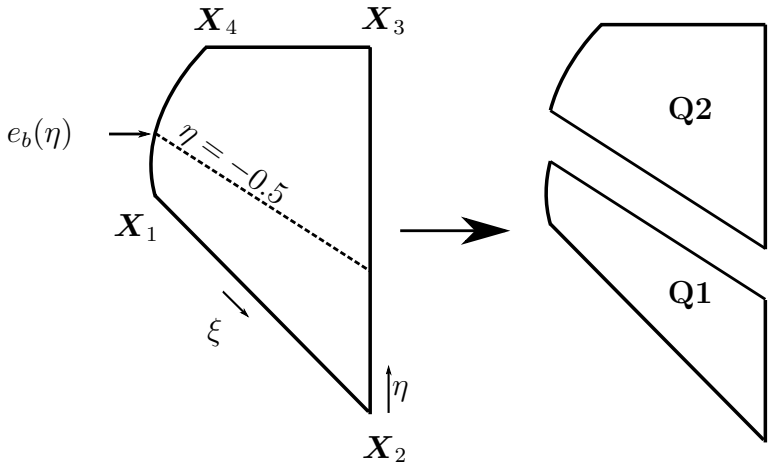


Figure 2.19: *Example of knot subdivision [16].*

the complete integral is computed by the sum of the integrals on the subcells.

In general, the triangulated subdivision algorithm is followed by an additional decomposition: all cells that are bounded by piecewise defined curves are further subdivided along the breakpoints of the curves.

As it will be pointed out later, this breakpoint-wise subdivision is a necessary step in order to be able to compute highly accurate integrals of subcells bounded by piecewise curves.

2.2.2.4 Smart octrees - Conforming integration meshes in 3D[b]

While the previously outlined approach is able to deliver an accurate and robust integration for two-dimensional discontinuous functions, its extension to three dimensions is not straightforward. The challenge is not only the drastically increased number of intersection cases, but also the necessity to account for the kinks and sharp edges of the geometric model that lie inside a cut element. For implicitly defined geometries, [52] presents an algorithm to decompose the three-dimensional cut elements into non-overlapping tetrahedra and prisms. Similarly to the 2D case, the integration subcells use Lagrangian functions as mapping functions. Other three-dimensional approaches rely on predefined intersection templates: In the framework of the Cartesian grid Finite Element Method (cgFEM [53]), [54] introduces a method using the basic intersection patterns of the marching cubes algorithm [55] as templates, employing the NEFEM concept for the mapping functions. When sharp features are present in the cell, the method generates specific sets of tetrahedra using a Delaunay procedure.

In the following, an algorithm is presented which:

- generates boundary-conforming integration cells,
- does not rely on predefined template solutions,
- resolves sharp features of the BREP model.

To break down the complexity of the problem, the algorithm follows a two-step approach. The first step resolves the intersection topology by subdividing each cut cell into eight octants and moving the 19 internal nodes of the octree so that they lie exactly on the boundary $\partial\Omega$. If the geometric model contains parametric entities that are defined by nonlinear mapping functions, the second step reparametrizes the trilinear subcells in such a way that they conform

to the curved interface. The following two sections serve to explain the two steps of the algorithm in detail.

First step - decomposition into trilinear subdomains

As discussed before, similar to the octree procedure, the first step delivers 8 subcells. Now, the octants are arranged such that none of the octants have vertices on different sides of the domain interface. The essential key to achieve this purpose is to realize that the position of the 19 internal nodes is not fixed. Instead, they can be shifted along their corresponding edge, face or volume without changing the topology of the geometric objects. This allows to separate the octants into the two domains Ω_{phy} and Ω_{fict} by moving the internal nodes onto the interface as follows (refer to Figure 2.20):

- If any of the 12 edges is intersected by the domain $\partial\Omega$, it is identified as an *active edge*. On active edges, the corresponding edge node is moved to the point of intersection. If an edge is inactive, its edge node stays in the center.
- If any of the 6 faces is bounded by an active edge, the face is identified as an *active face*. On active faces, the face node is moved onto the intersection curve between $\partial\Omega$ and the face. If the face is inactive, its face node remains in its center.
- The mid-node of the cut cell is moved onto $\partial\Omega$.

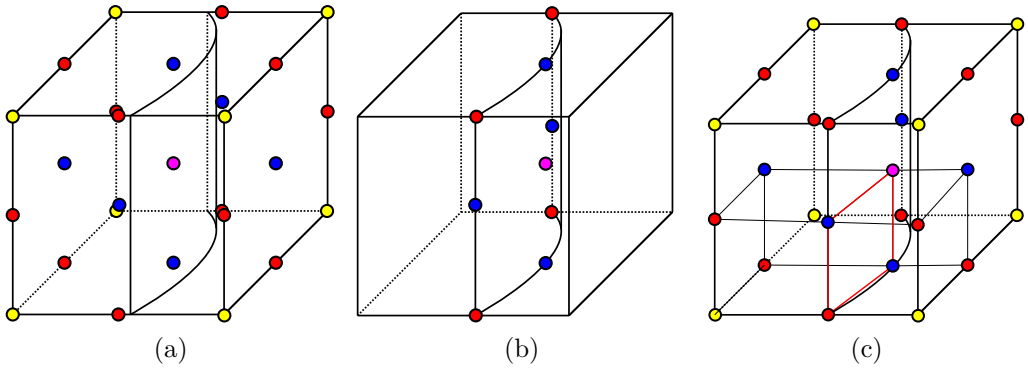


Figure 2.20: *Smart octree generation. All edge nodes (yellow), face nodes (red) and the mid-node (blue) is shown in (a). The nodes on active edges, faces and the mid-node are moved onto the interface (b). For simplicity, (c) shows only two resulting subcells. The corners of the red bilinear quadrilateral lie on the interface [17].*

Computing the intersection point between the active edges and the interface is a relatively easy task, as it only requires an evaluation of the curve-surface intersections.

Concerning active faces, the location of the corresponding face point is determined as follows. First, the four corners of the active face are checked whether they lie in Ω_{phy} or in Ω_{fict}. If two opposite corners lie on the opposite sides of the interface, the diagonal line connecting them is an active diagonal. When the active diagonal is identified, the face node is moved to the point of intersection between the diagonal and the interface. This curve-surface intersection problem is significantly easier to solve than computing an explicit intersection curve between the cell face and the interface.

Similarly, the mid-node is moved onto the interface by identifying one of the active diagonals in the cell and computing its intersection with the interface. Figure 2.21 depicts an example of moving a face node and a mid-node based on the active diagonal.

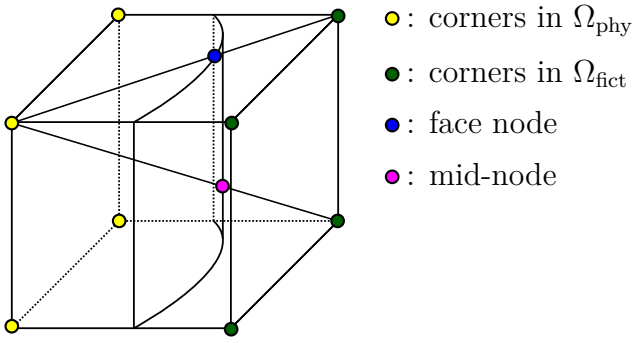


Figure 2.21: *Active diagonals are identified by checking whether opposite corners lie on opposite sides of the interface. Face nodes and mid-nodes are created by intersecting the corresponding active diagonal with the interface [17].*

So far, the outlined approach requires the interface inside the cell to be smooth and free of edges and sharp corners. In the following, the algorithm is extended to also account for these situations.

First, consider the case of a sharp edge intersecting the cell. Let $\boldsymbol{c}(t)$ denote the parametric mapping that represents this edge. In order to resolve this feature, the face nodes are computed as the intersection between $\boldsymbol{c}(t)$ and the corresponding face of the cell. These intersection points are characterized by the parameter values t_1 and t_2. Knowing these two values, the mid-node is computed as $\boldsymbol{c}(^1/_2 \cdot (t_1 + t_2))$. Figure 2.22 shows how sharp edges are resolved in cases where the edge enters and exits the cell on two opposite faces.

If two or more sharp edges meet at a vertex v inside the cell, the algorithm moves the mid-node to the location of v. The edge and face nodes are computed as outlined before. This is depicted in Figure 2.23.

Topologically exceptional cuts

The outlined algorithm requires that every edge, face and cell has a single associated internal point v. If this criterion is not fulfilled, the cut case is called *topologically exceptional*. Cases like this can be detected by checking whether the curve-surface intersection results in more internal points than expected. The topologically exceptional cases are reduced into simple cases by subdividing the cut cells into equal octants. This *refinement step* is repeated recursively until the criterion of single internal nodes per edge/face/cell is fulfilled in the octants. The resolution of a typical topologically special case is depicted in Figure 2.24. The refinement step is also applied for cells where the decomposition would result in negative Jacobian determinants for example in cases of concave integration subcells. If a topologically exceptional case persists after a maximum number k_{max} of octree refinement steps, the *fallback strategy* is to compute integrals by classical octree. In the following, typical examples are given for situations where refinement steps or the fallback strategy might be needed.

Because only one sharp corner can be resolved by moving the mid-node of the cell to its location and only one sharp edge can exit the cell per cell-face, there are restrictions on the

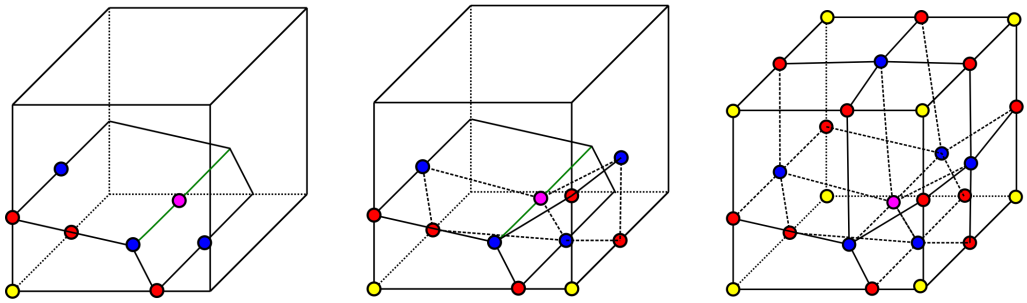


Figure 2.22: *Edge resolution. Active nodes that take part in the formation of one subcell with a sharp edge (in green color) are shown in (a). The face nodes are the intersection points between the edge (green color) and the corresponding faces. Two subcells are shown in (b), while the complete subdivision is shown in (b) [17].*

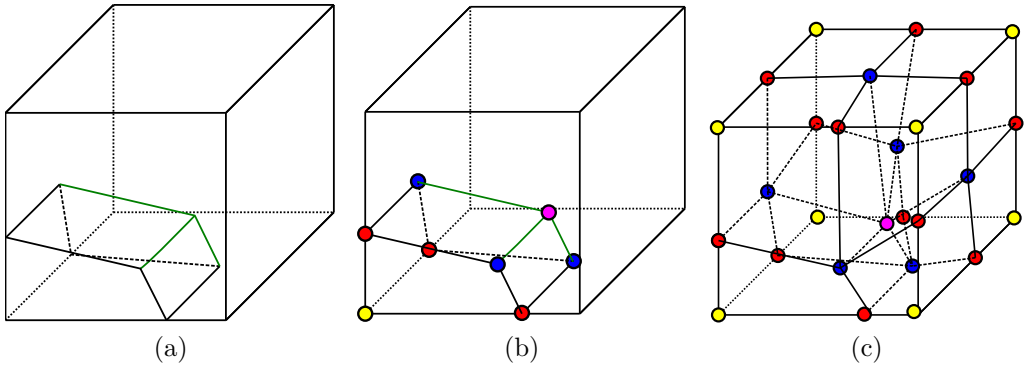


Figure 2.23: *Sharp corner resolution. Three sharp edges (dark green color) meet in a sharp corner, as shown in (a). The mid-node of the cell is moved onto the sharp corner, while the sharp edges (green color) are dealt with as shown in Figure 2.22. One subcell is shown in (b) and the complete subdivision is shown in (c) [17].*

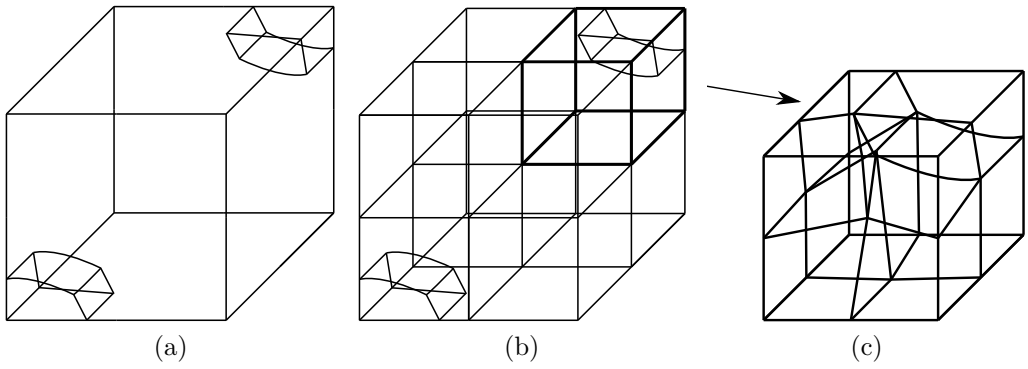


Figure 2.24: *The algorithm performs an octree subdivision if the cut case cannot be resolved in one step: (a) topologically complex cut in the cell, (b) the cell is subdivided in to eight octants, (c) base smart octree case in one of the octants [17].*

sharp geometric entities interior to the cell. While the "one sharp corner" restriction can be resolved by a recursive application of the refinement step, the restriction on the number of edges exiting the cell requires further considerations. If more than two coplanar or more than three non-coplanar sharp edges meet at the mid-node, additional angle conditions have to be fulfilled in order to guarantee that they exit the cell, or its octree-like refined cells through different faces. The examples in Figures 2.25 and 2.26 demonstrate these situations. Furthermore, because there are only six face nodes—each of them connected to the mid-node—there can be a maximum of six sharp edges meeting at the center node of the cell. This cannot be resolved even by a recursive octree refinement, as the example in Figure 2.27 demonstrates. In these situations, the fallback strategy is applied, integrating on the classical octree cells. The application of the refinement step and the fallback strategy are major factors contributing to the robustness of the algorithm, because both of them are based on the—per definition robust—octree method. As demonstrated by the practical example in Section 2.2.3.2, in the majority of the cells no refinement is necessary and only a negligible amount of cells reach the refinement level where the fallback strategy is needed. Even if it is applied, the introduced integration error is small, as it is generally restricted to very small parts of the domain of computation. In our practical examples, we choose k_{max} between 3 and 5, as a compromise between computational effort and accuracy of the numerical integration.

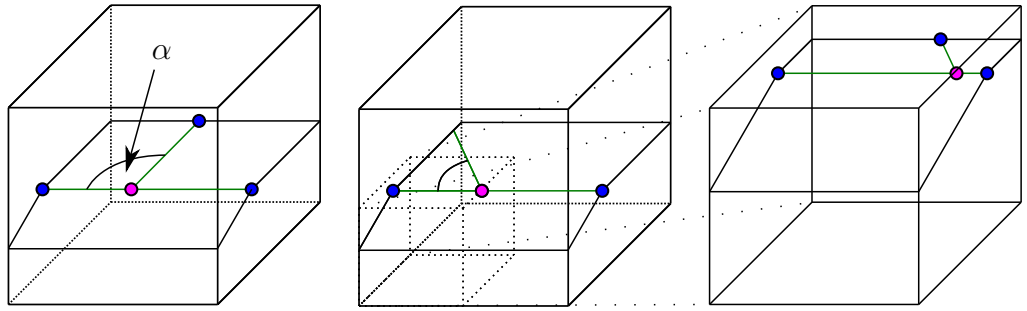


Figure 2.25: *More than two coplanar edges (green color) meeting at a BREP vertex (purple). If the angle α between them is sufficiently high (a), they exit the cell through different faces (blue nodes). If α decreases, the edges intersect the same face (b). In this case, the refinement step has to be applied until the basic smart octree case can be recovered (c) [17].*

Relation to finite element mesh generation techniques

It is important to note here that the method differs significantly from conventional finite element mesh generators, because its aim is to generate subdomains for numerical integration purposes. Therefore it is not constrained by those criteria that other meshing algorithms have to fulfill. For the task of numerical integration, the aspect ratio or "quality" of the generated cells plays a less important role. Furthermore, the integration mesh allows for hanging nodes and edges, which can become tedious to handle in FEM, and therefore are not allowed in classical finite element meshes.

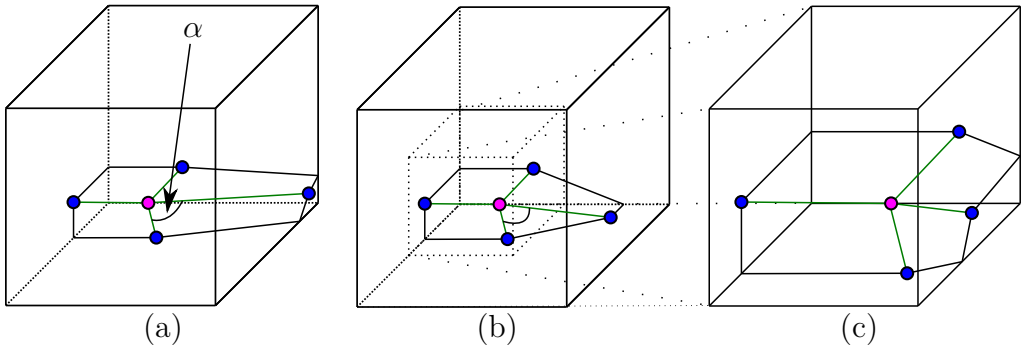


Figure 2.26: *More than three non-coplanar edges (green) meeting at a BREP vertex (purple). If the angle α between them is sufficiently high (a), they exit the cell through different faces (blue nodes). If the angle is too small (b), the refinement step is applied to reduce the cut case into a basic smart octree case (c) [17].*

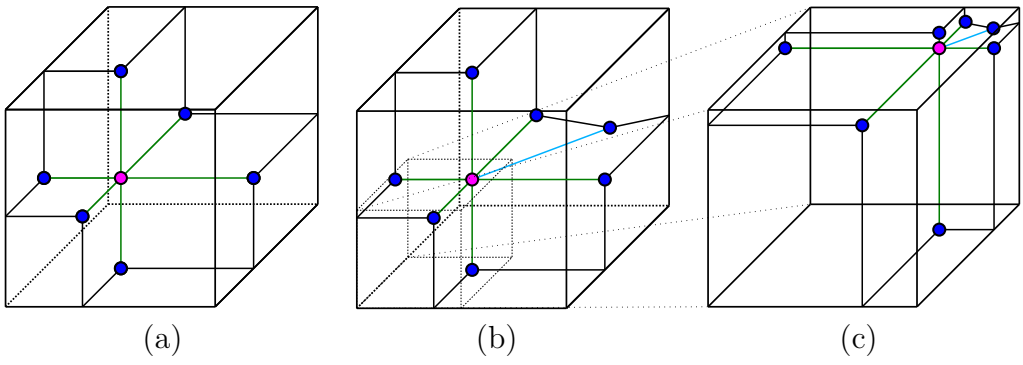


Figure 2.27: *Example on the restriction of number of edges meeting at a BREP corner. In (a), six sharp edges (green) meeting at the irregular node (purple) exit the cell through different faces (blue nodes). In (b), a seventh edge (light blue) ends at the irregular node. This case cannot be reduced into a basic smart octree case even by an octree refinement (c) [17].*

Second step: elevating the order of linear subdomains

As described in the previous paragraphs, the first step of the smart octree algorithm partitions the cut cell into eight octants, whereby $\partial\Omega$ is approximated linearly. In this way, the complexity of the intersection pattern is resolved. Thus, the internal domain interface is guaranteed to be smooth in each octant. This alone leads to an improved domain integration when combined with composed quadrature. Further improvements are possible when the parametric entities of the geometric model are of high order. In this case, the trilinear subcells can be reparametrized to conform with the curved boundaries of Ω_{phy}. This is the aim of the second step of the smart octree approach. To this end, the linear version of the transfinite interpolation approach [23] is employed. This method—also known as blending function interpolation [56]— defines a mapping between a reference cube and a hexahedron bounded by curved parametric surfaces and edges.
The blending function method requires the high order information of Ω_{phy}. To this end, a local reparametrization of the surface is employed, based on interpolating a set of sampling points, similar to the method explained in [52].
The locations of the sampling points are chosen according to the parametric collocation points computed by Babuška and Chen [57]. The sampling points are projected on Ω_{phy} and interpolated using Lagrangian functions, as depicted in Figure 2.28. The faces of the resulting curved hexahedra approximate the high-order geometric boundaries with a higher accuracy.

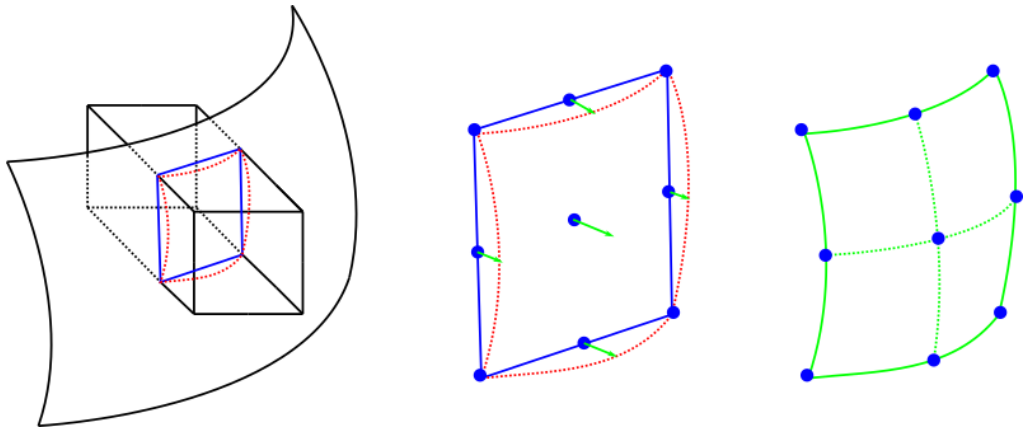


Figure 2.28: *If a cell is cut by a high-order surface (a), the order of a linear quadrilateral (blue) is elevated to conform with the section of the curved surface (red). Interpolation points are sampled on the quadrilateral (blue dots) and mapped on the real surface (b). These points are interpolated, resulting in an approximating quadrilateral (c) [17].*

2.2.3 Numerical examples

This section demonstrates the decomposition methods of Sections 2.2.2.3 and 2.2.2.4.

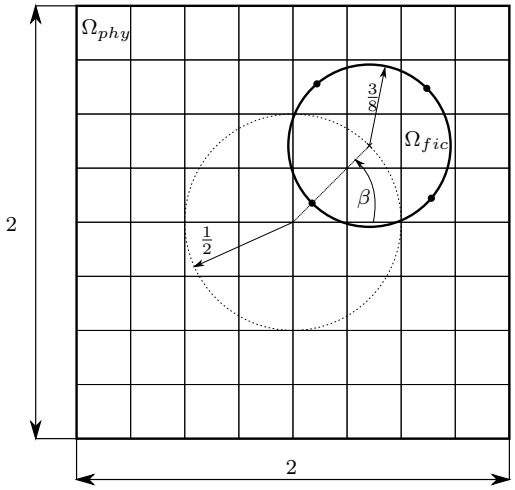


Figure 2.29: *Setting of the moving circle example [16].*

2.2.3.1 Integration in 2D[a]

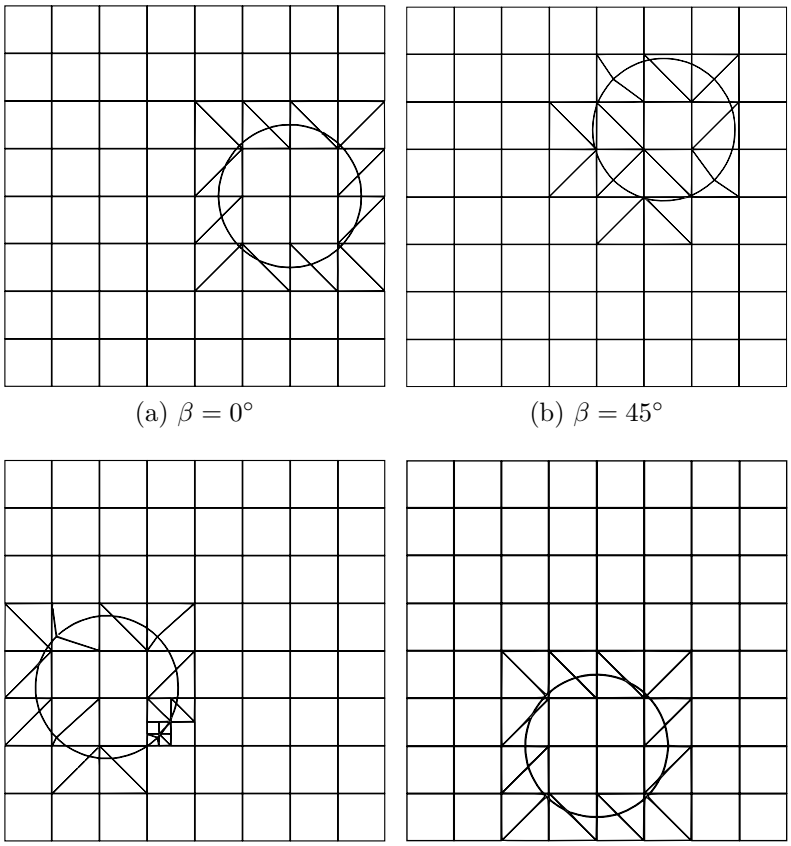
Consider a square-shaped domain with a circular hole inside, the center of which moves on a circular path (Figure 2.29). The boundary of the circular hole is composed of 4 arcs. In every time step, a different geometrical setting has to be partitioned. This way, it can be assessed, how the 2D algorithm resolves with non-regular configurations. As Figure 2.30 depicts, every cut cell is decomposed into two pairs of quadrilaterals and triangles. In cells where neighboring arcs join, the diagonal cutting line is drawn according to the meeting point of the curves. It is worth observing how the degenerate case is resolved by the quadtree decomposition in Figure 2.30c.

Integration test

The precision of the entire subdivision algorithm is assessed on a geometric setup that contains all the special cases of Figures 2.16 and 2.17. To this end, an "integration patch test" is introduced, with the following idea: the value of the scaling factor is chosen to be $\alpha = 1$ both on Ω_{phy} (with a possibly complex geometry) and Ω_{phy}. Because the scaling factor is the same in both domains, the discontinuity in the cells vanishes and the numerical problem simplifies to a 2D finite element computation on a rectangular domain with a quadrilateral mesh. If this domain is subjected to constant strains, the linear shape functions spanned on the quadrilateral mesh have to be able to represent the solution exactly, because the completeness condition is satisfied [19, 20]. Note that the integrands of the element stiffness matrices (Equation 2.27) are still computed on the subcells resulting from the decomposition algorithm. Therefore, any possible difference between the numerical and the analytical solution is a sign that there is an error in the integration. To quantify these differences in a global sense, the numerical and analytical strain energy are compared using the following error measure:

$$e = \sqrt{\frac{|u_{ex} - u_{num}|}{u_{ex}}}, \tag{2.52}$$

where u_{ex} and u_{num} are the exact and numerical values of the strain energy, respectively.



(a) $\beta = 0°$ (b) $\beta = 45°$

(c) $\beta = 202.5°$ (d) $\beta = 270°$

Figure 2.30: *Decomposition example: moving circle [16].*

As an example, consider the geometric setting depicted in Figure 2.31. The boundaries of Ω_{phy} represent the cover plate of a violin, including the f-holes. The boundary of the violin - including the f-holes - are represented by cubic B-Spline curves. On this domain, the Laplace equation

$$\Delta\Phi = 0 \tag{2.53}$$

is solved with Dirichlet boundary conditions applied on the top and bottom of the mesh grid, such that the field value $\Phi = 0$ on Γ_{D_1} and $\Phi = h$ on Γ_{D_2}, where h is the height of Ω_{fict}. As a result, the gradient of the field value $\frac{\partial\Phi}{\partial x}$ is equal to one throughout the whole domain. The boundaries denoted by Γ_N are defined as free Neumann boundaries. Because the solution is linear, the polynomial order of the shape functions is chosen to be $p = 1$.

Figure 2.32a illustrates the results of the decomposition without applying breakpoint-wise subdivision, while the resulting integration mesh with breakpoint subdivision is depicted in Figure 2.32b.

The analytical value of the strain energy is

$$u_{ex} = \frac{1}{2}\int_{\Omega}\left(\frac{\partial\Phi}{\partial x}\right)^2 d\Omega = \frac{1}{2}wh, \tag{2.54}$$

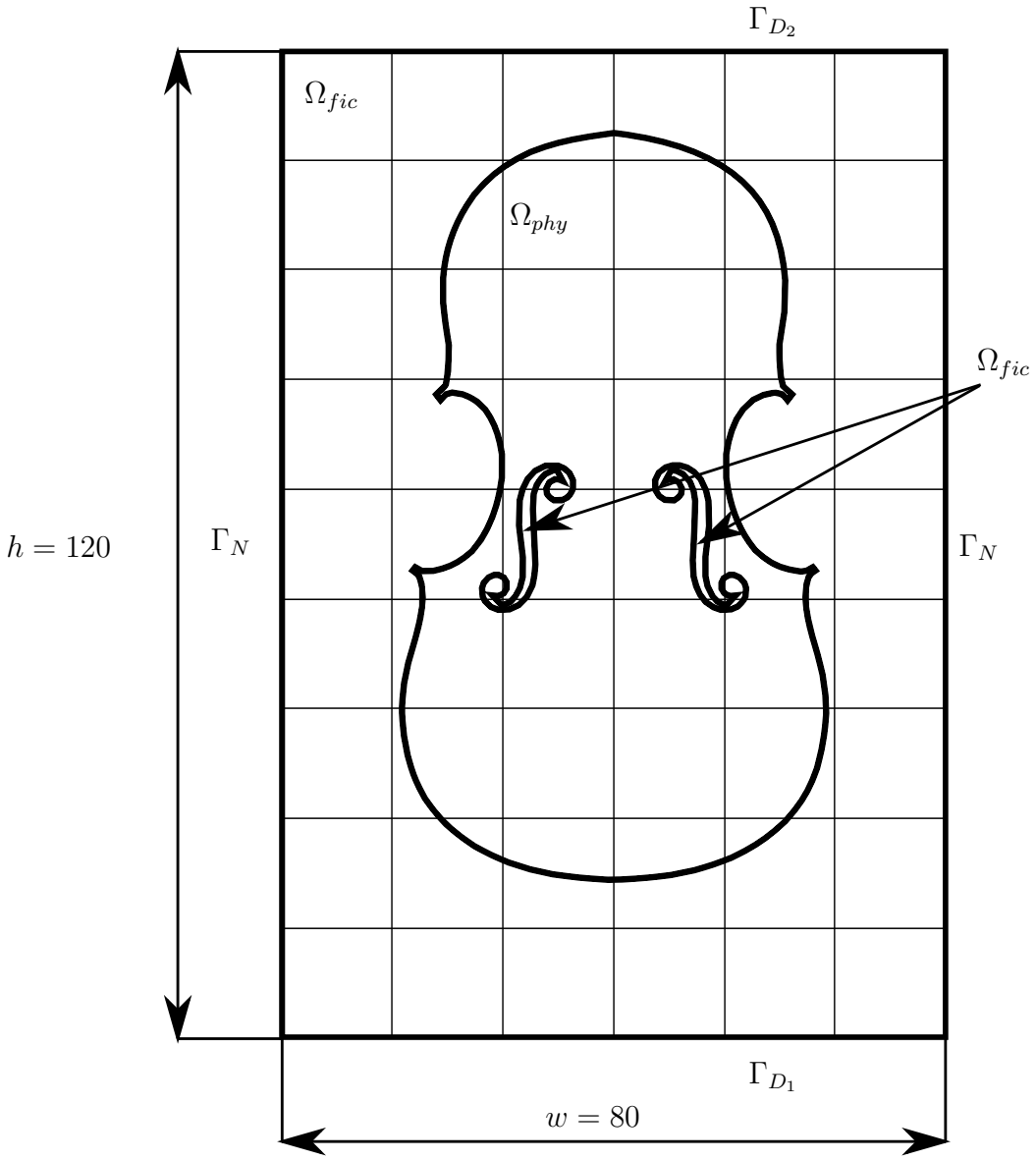
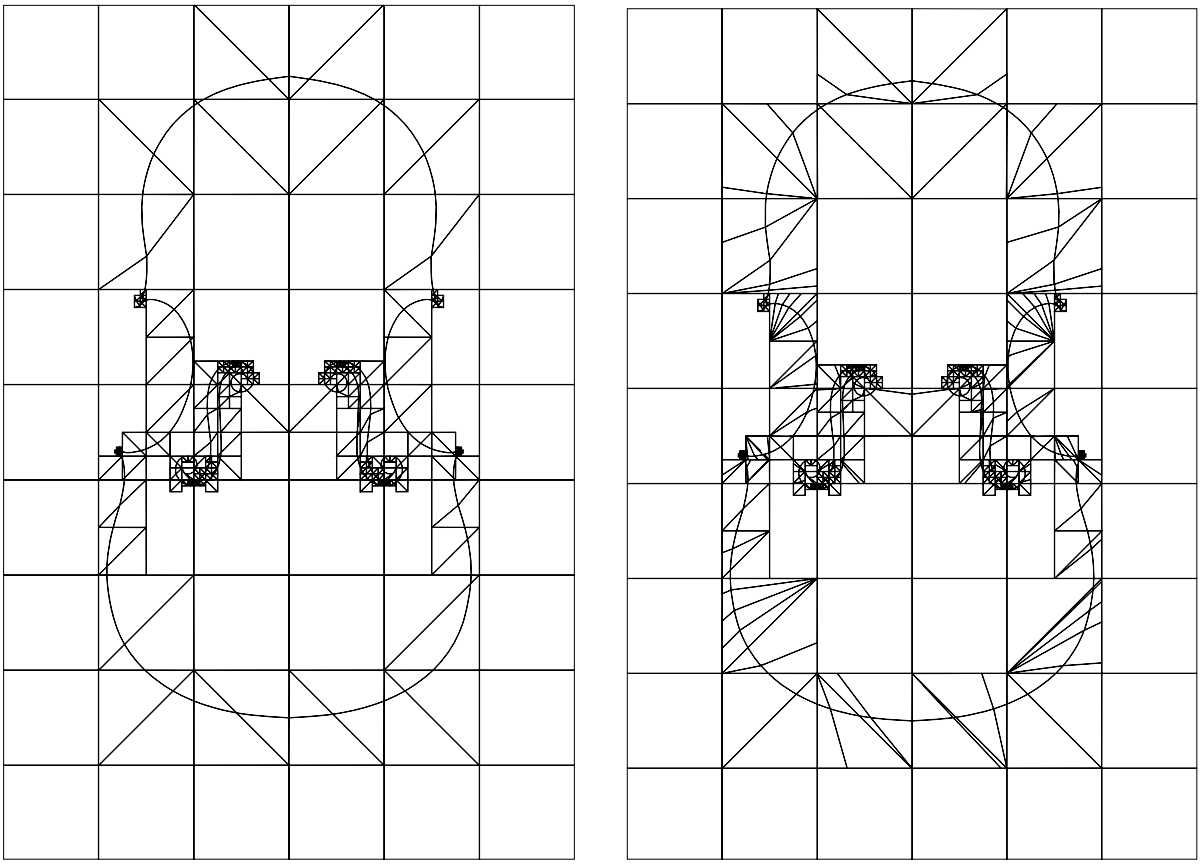


Figure 2.31: *Geometry setup for the violin partitioning example [16].*



(a) Without knotspan subdivision (b) With knotspan subdivision

Figure 2.32: *Result of partitioning on the violin example [16].*

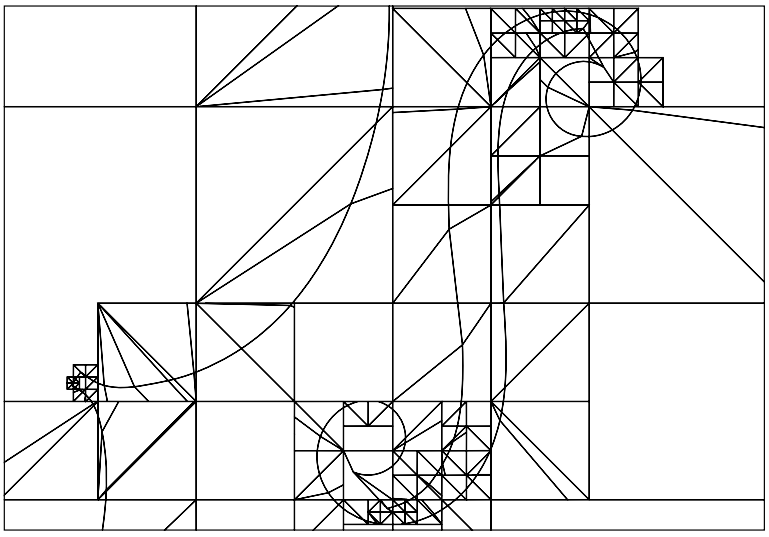


Figure 2.33: *A detailed view of the f-hole [16].*

where w denotes the width of the domain. Figure 2.34 shows the error in the strain energy depending on the number of quadrature points distributed per parametric direction in each cell.

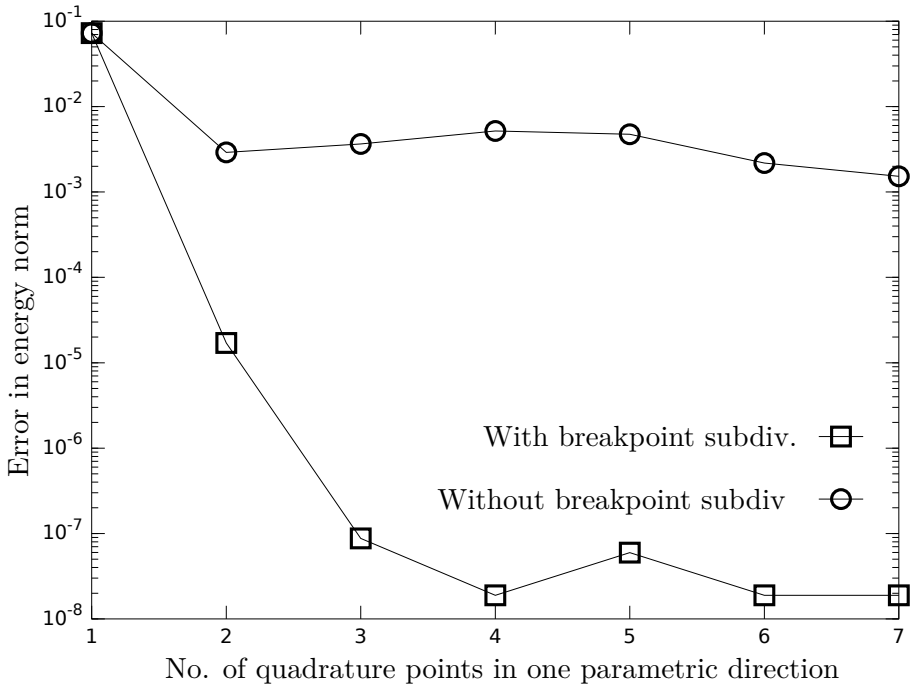


Figure 2.34: *Error in energy norm with and without knotspan subdivision [16].*

If the breakpoints are not taken into account as explained on page 27, the error remains higher regardless of how many integration points are distributed in the cells. However, if the integration cells are subdivided along the breakpoints of the bounding curves, the error in the energy norm converges to machine precision.

Here, it is worth noting that the total number of integration points is influenced by the parametric definition of the curves to a great extent. This means that if there are many breakpoints present in the boundary curves, there will also be a lot of breakpoint-wise subdivision performed. Therefore a high number of quadrature points is generated which results in high overall runtime.

In order to determine whether a cell is cut, both the blended and the quadtree based approaches require the evaluation of the inside-outside state of dedicated seed points. However, due to its recursive nature, the quadtree based technique has to evaluate the point membership in every cell of every level of subdivision. Thus, for higher subdivision levels the quadtree based integration mesh generation becomes significantly more expensive than the proposed algorithm. This is also confirmed by the comparison in Figure 2.35.

2D integration for the FCM

In the following, the combination of the 2D integration algorithm and the FCM is demonstrated. To this end, consider the plane stress problem that was already analyzed in the context of FCM [29], with the geometric setting depicted in Figure 2.36. The material of the perforated plate is steel, with the properties $E = 2.069 \cdot 10^5[MPa]$, $\nu = 0.29[-]$. The

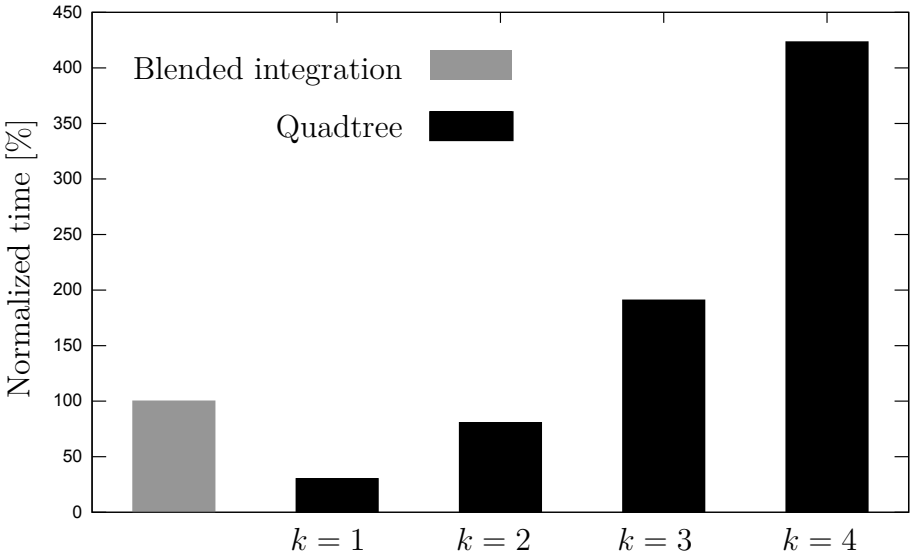


Figure 2.35: *Comparison of the time required for generating different integration meshes for the violin example [16].*

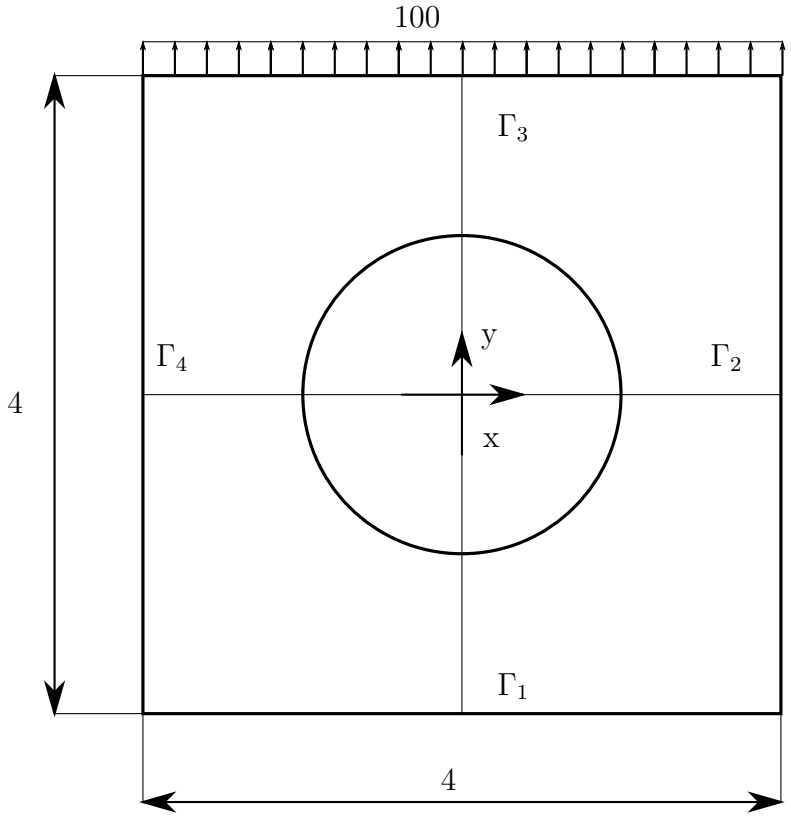
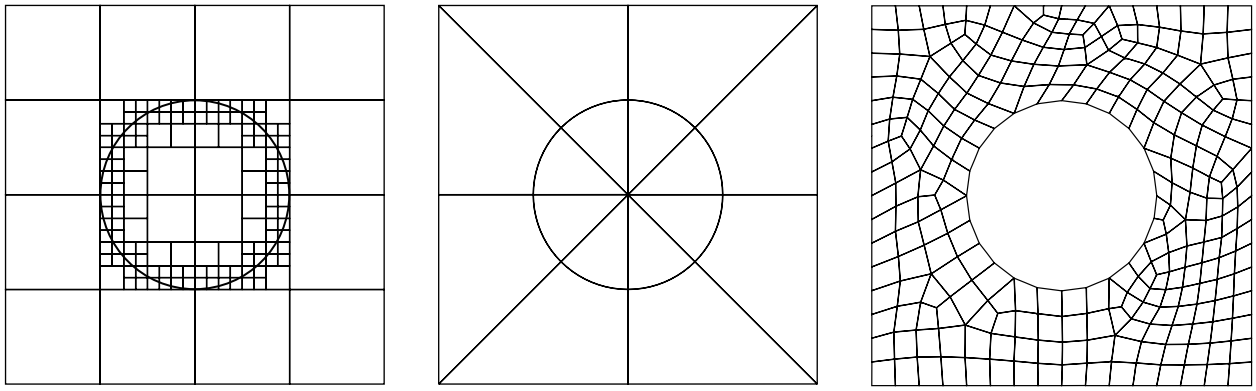


Figure 2.36: *Perforated plate problem [16].*

plate is vertically loaded by $100[MPa]$. Symmetry conditions are applied on Γ_1 and Γ_4. The boundaries of the hole and Γ_2 are treated as free boundaries. The domain is discretized into 2×2 finite cells. The polynomial degree of the shape functions is increased from $p = 1$ to $p = 15$. The reference strain energy of the problem is $U = 0.7021812127$, obtained by an "overkill" p-FEM solution from [29].
The integration is performed on blended integration cells and on quadtree cells with depths $k = 4$ and $k = 5$. For comparison, the same problem is solved by means of linear finite elements (h-FEM) with different element sizes. For the quadtree integration cells and the linear finite elements the number of quadrature points is chosen to be $(p + 1)^2$. To account for the high order boundaries of the blended integration mesh, $(p + 4)^2$ integration points are distributed in the curved integration cells. Figure 2.37 depicts the integration cell meshes and a mesh of linear finite elements. The error in the strain energy (Equation 2.54) is plotted in Figure 2.38a.



(a) Quadtree integration mesh $k = 4$

(b) Blended integration mesh

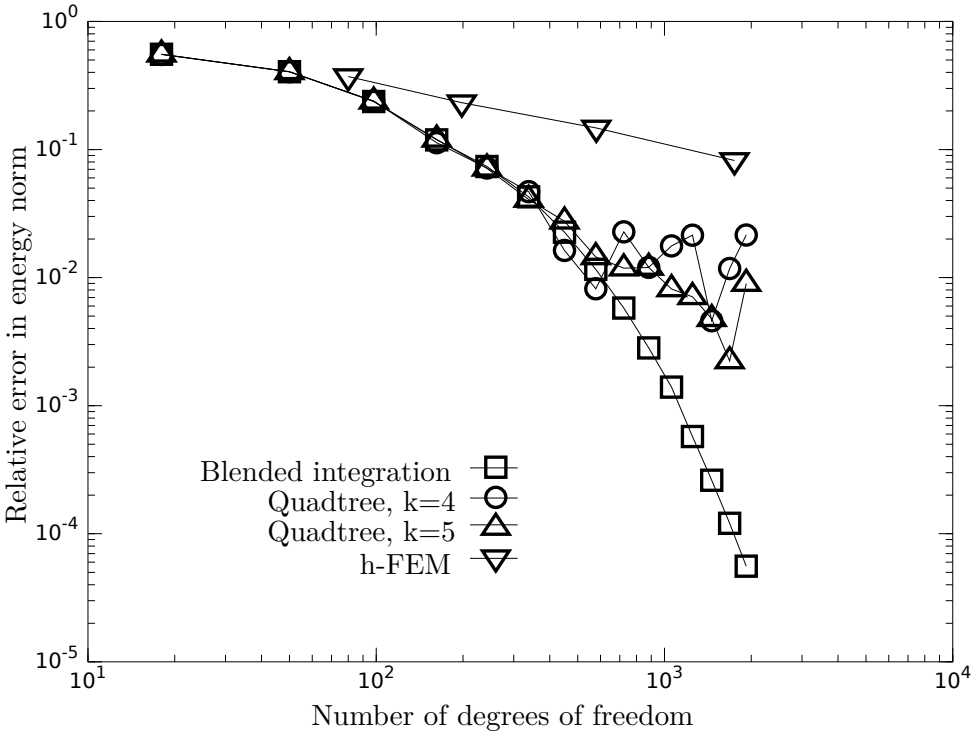
(c) Mesh of 246 linear quadrilaterals

Figure 2.37: *Integration meshes and a finite element mesh for the perforated plate example [16].*

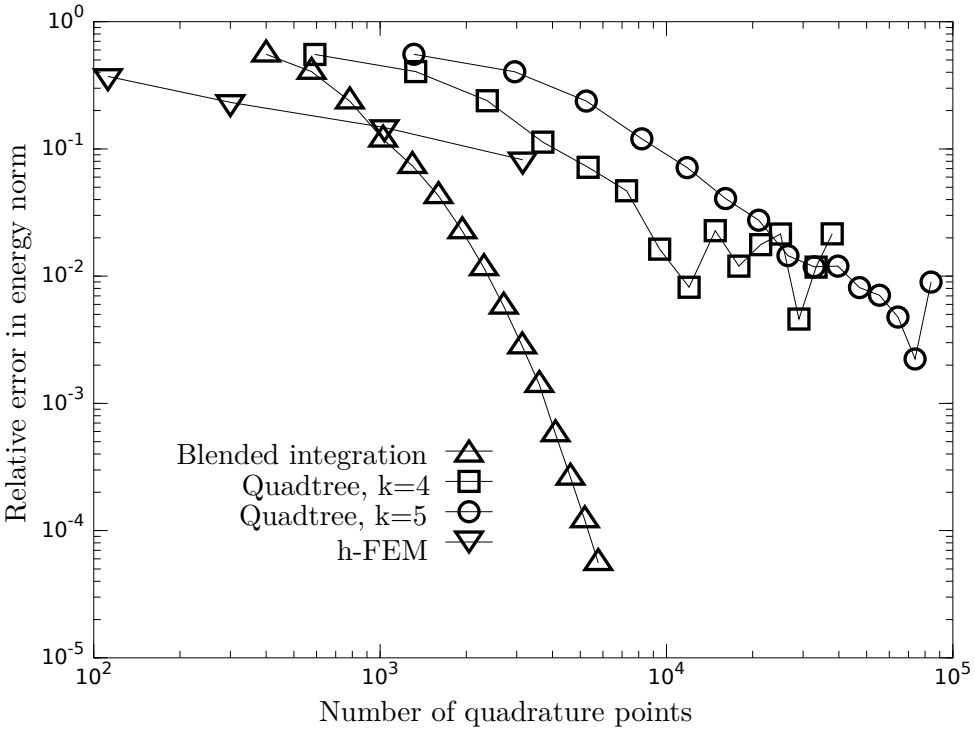
Both the quadtree and the blended integration show exponential convergence, similar to p-FEM. However, the curve representing the quadtree integration levels off at an error of approximately $10^{-2}[-]$.

At this point the integration error dominates over the discretization error which renders a further increase of the polynomial degree pointless. The integration error can be reduced by adding more levels of refinement to the spacetree subdivision - however, the low approximation of the integration does not allow for exponential convergence in the asymptotic range. In comparison, the blended integration that uses the parametric description of the boundaries shows exponential convergence also in the asymptotic sense.

It is worth noting that the curves corresponding to the spacetree-based integration does not represent a monotonic convergence behavior. Instead, they tend to oscillate around the level-off value. The reason for this lies therein that for a fixed spacetree depth, increasing the integration order changes the ratio of quadrature points that are inside or outside. Depending on this ratio, the spacetree-based integration may over- or underestimate the integral, leading to the oscillation.



(a) Convergence w.r.t. no. of degrees of freedom



(b) Convergence w.r.t. no. of quadrature points for $p = 1..15$

Figure 2.38: *Comparison of the convergence characteristics of different methods [16].*

Discussion

The number of quadrature points has a major influence on the overall computational cost of a numerical simulation. Therefore, the relationship between the number of integration points and the relative error in strain energy is an important aspect when it comes to assessing the performance of the different approaches. This relationship is depicted in Figure 2.38b.

Apart from the better convergence characteristics, the other advantage of the blended subcell integration lies in the total number of Gauss points distributed on the domain. For the same error in the strain energy, the blended integration cells require approximately one order of magnitude less Gauss points than a quadtree integration mesh with a depth of $k = 4$. If the depth of the quadtree is increased to $k = 5$, the point at which the convergence curve levels off is slightly shifted to a lower value. However, this gain in precision comes with the cost of high computational overhead. As an example, consider a relative error of 10^{-4} in the energy norm, where the $k = 5$ quadtree integration mesh needs approximately 40000 quadrature points. In contrast, the blended mesh needs approximately 2500 integration points to reach the same error.

A comparison in terms of the time required to compute the stiffness matrix -including the generation of the integration mesh- is plotted in Figure 2.39. The quadtree based computation becomes more expensive than the blended one for two reasons. The first reason is due to the extra inside-outside tests required by the quadtree with high levels of k (refer to Section 2.2.3.1). Secondly, the high number of quadrature points leads to an excessive number of matrix-matrix product evaluations in Equation 2.24.

Both the blended and quadtree methods show better convergence characteristics in comparison to the standard h-FEM on the basis of the number of degrees of freedom (Figure 2.38a). Comparing the number of integration points of the different approaches reveals that up to approximately 1000 integration points the error in the strain energy of the h-FEM solution is smaller than the error of the blended integration. This point is located where the curve of the h-FEM error intersects the curve of the blended integration error on 2.38b. For the quadtree methods, this intersection with the h-FEM error curve lies in regions of higher number of Gauss points.

Here, it should be noted that although the h-FEM is at least as precise as the blended subcell integration up to this intersection point, it requires considerably more degrees of freedom than the FCM approach.

2.2.3.2 Integration in 3D[b]

The following section demonstrates how the 3D integration algorithm performs in combination with composed numerical quadrature. The first part of the section will focus on accurately computing volume integrals on BREP domains.

Integration test on models with sharp boundaries

This first example aims at demonstrating that the trilinear subcells obtained after the first step of the algorithm can accurately and robustly resolve edges and vertices in the structure. For this purpose, Ω_{phy} is considered as the two concentrically stacked cubes depicted in Figure 2.40.

The domain is extended by Ω_{fict}, forming a bounding box $\Omega_{\cup}$. A structured background mesh is generated on the geometry of $\Omega_{\cup}$, with $4 \times 4 \times 5$ cells. Note that the bounding box was

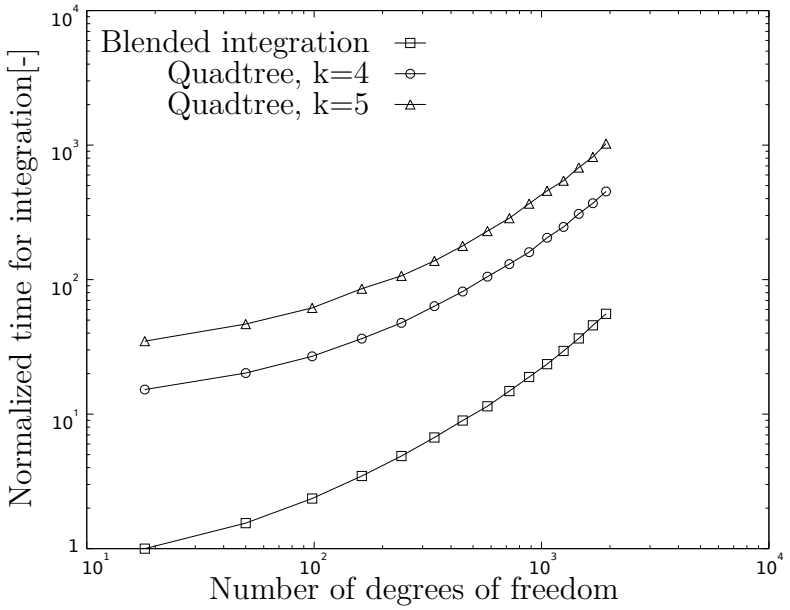
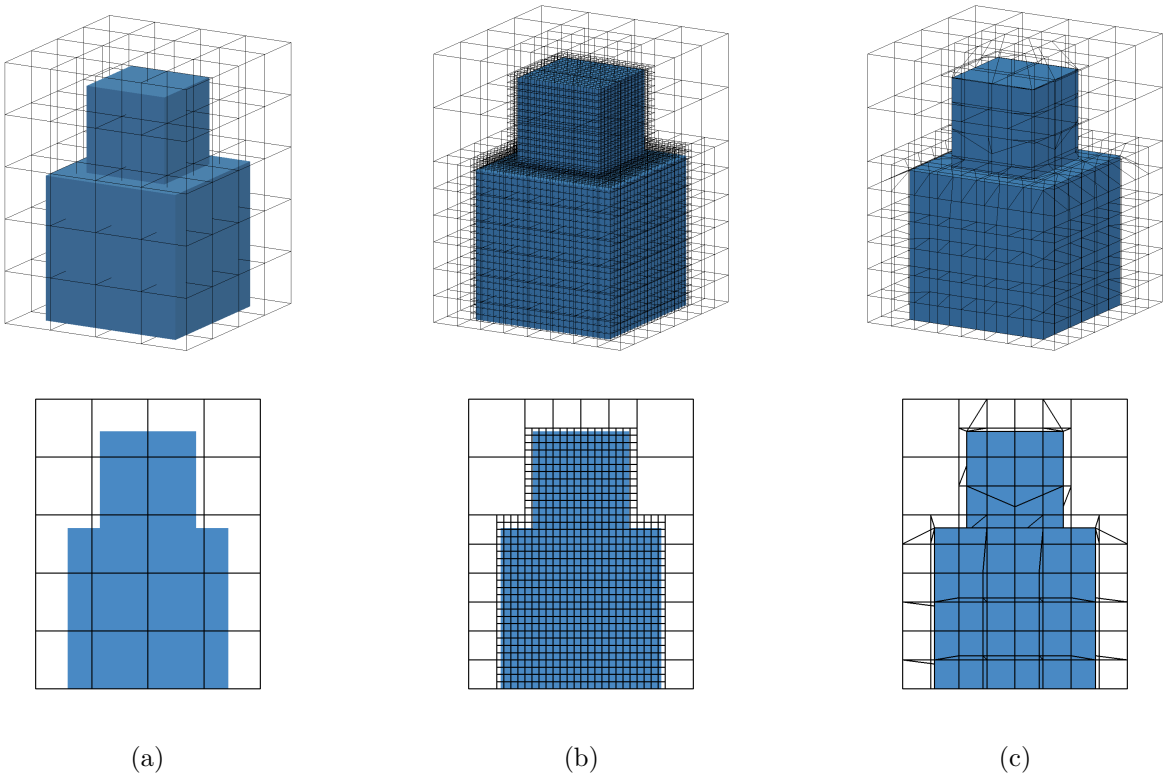


Figure 2.39: *Comparison of the time required for integrating the stiffness matrix for* $p = 1..15$ *[16].*



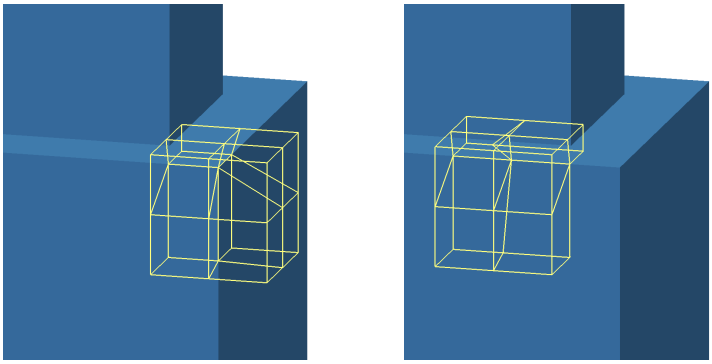
(a) (b) (c)

Figure 2.40: *Example with planar faces: (a)* Ω_{phy} *with background mesh, (b) octree with* $k = 3$*, (c) smart octree [17].*

intentionally chosen such that the classical octree refinement never catches the boundaries of the domain exactly. Let $f(\boldsymbol{x}) = 1, \forall \boldsymbol{x} \in \Omega_{\text{phy}}$ and $g(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in \Omega_{\text{fict}}$. Then, the following integral value is sought:

$$\int_{\Omega_\cup} F(\boldsymbol{x}) = \int_{\Omega_{\text{phy}}} 1 d\Omega + \int_{\Omega_{\text{fict}}} 0 d\Omega = V_{\Omega_{\text{phy}}}, \tag{2.55}$$

where $V_{\Omega_{\text{phy}}}$ is the volume of Ω_{phy}. As the geometry of Ω_{phy} is simple, its exact volume can be computed analytically. Octree integration meshes with levels $k = 2, 3, 4$ are generated on the background mesh, as well as a smart octree mesh. Figure 2.40 shows these different integration meshes. Figure 2.41 shows two cut cells containing an edge and a vertex, as well as their resolution. As the geometry is bounded by planar faces, it is sufficient to compute the integral on the subcells provided by the first step of the algorithm.



(a) (b)

Figure 2.41: *Non-smooth boundary inside the cells: (a) single vertex, (b) sharp edge [17].*

To assess the accuracy of the different spatial subdivision approaches, Gauss-Legendre quadrature is applied in the integration cells with stepwise increasing integration order by distributing $n \times n \times n$ integration points in each cell, with $n = 1..5$. The methods are compared using the following error measure:

$$e = \frac{|V_{ex} - V_{num}|}{V_{ex}}, \tag{2.56}$$

where V_{ex} is the exact domain integral (for this case $V_{\Omega_{\text{phy}}}$) and V_{num} is the numerically computed one, using different integration meshes.
Because the smart octree algorithm separates the domain of the piecewise constant integrand into subdomains where the integrand is constant, V_{num} is expected to be exact when the order of the quadrature is sufficient for accurately integrating the Jacobian determinant $||J||$ of the subcells. In this example, the subcells are defined by trilinear mappings, which is why $||J||$ is at most a quadratic function in every parametric direction. Consequently, V_{num} is expected to be exact if at least $2 \times 2 \times 2$ quadrature points are applied in each cell. This expectation is confirmed by the error curve of the smart octree approach depicted in Figure 2.42, where the curve reaches machine precision at the point that corresponds to $2 \times 2 \times 2$ integration points per cell. In contrast, the error for the octree approaches is orders of magnitude larger, even

for higher integration orders. This is due to the approximating nature of the spacetree. Thus, in case Ω_{phy} is bounded by planar surfaces, the smart octree method can clearly outperform the conventional spacetree approaches.

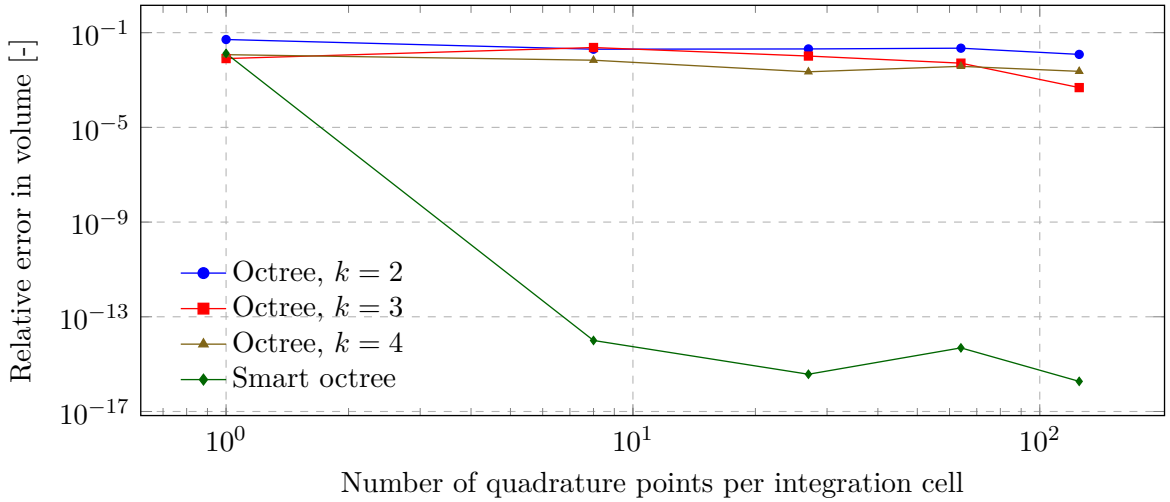


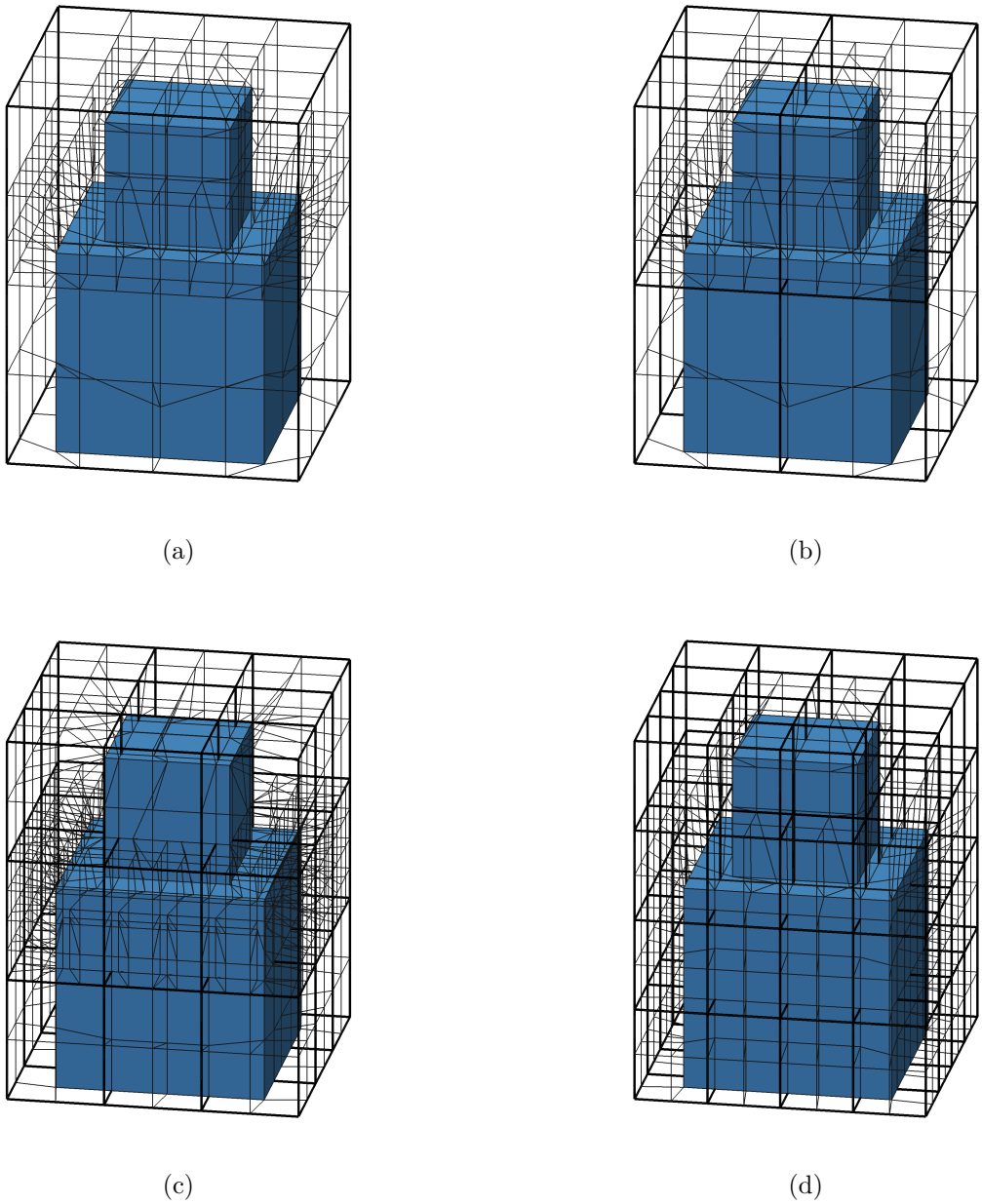
Figure 2.42: *Convergence of the error in volume w.r.t. the number of quadrature points per integration cell. The order of the integration is increased stepwise by distributing,* $n \times n \times n$ *quadrature points in every integration cell with* $n = 1..5$ *[17].*

These results show that the first step of the algorithm can robustly resolve sharp boundary features in the domain.
The octree-like refinement step ensures that the topologically special cases are reduced into simple ones. Therefore, regardless of how coarse the initial background mesh is chosen, the method keeps on subdividing the cells into equal octants until the geometry can be resolved by the smart octree approach. This means that the initial size of the elements in the background mesh is of minor importance when computing the volume integrals. To demonstrate this, the volume of the stacked cubes is computed using the smart octree approach on different background meshes. There are three initial configurations considered: the bounding box of the geometry is partitioned into $n \times n \times n$ elements, with $n = 1, 2, 3, 4$, as shown in Figure 2.43. For all the investigated cases, the error in volume reached numerical precision with $2 \times 2 \times 2$ quadrature points per integration cell, as it was shown in Figure 2.42.

The previous example demonstrated that the first step of the algorithm reliably resolves non-smooth geometric features of the domain boundary. The following example will show that, in the second step of the algorithm, also general domains bounded by curved geometries can be integrated with high accuracy. For this purpose, the geometry is changed to two concentrically stacked cylinders, as depicted in Figure 2.44. Like before, the domain is embedded into $\Omega_{\cup}$, and a background mesh of $3 \times 3 \times 4$ cells is generated. In this example, the integrand functions are chosen as:

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{1000}\left[(x-10)\cdot(x+30)\cdot(y-5)\cdot(y+20)\cdot\cos\left(\frac{z}{5}\right)^2\right] \\ g(\boldsymbol{x}) &= 0. \end{aligned} \tag{2.57}$$

The function $f(\boldsymbol{x})$ is constructed such that it does not behave symmetrically with respect to the features of the geometric model. Further, because it contains a trigonometric term, it



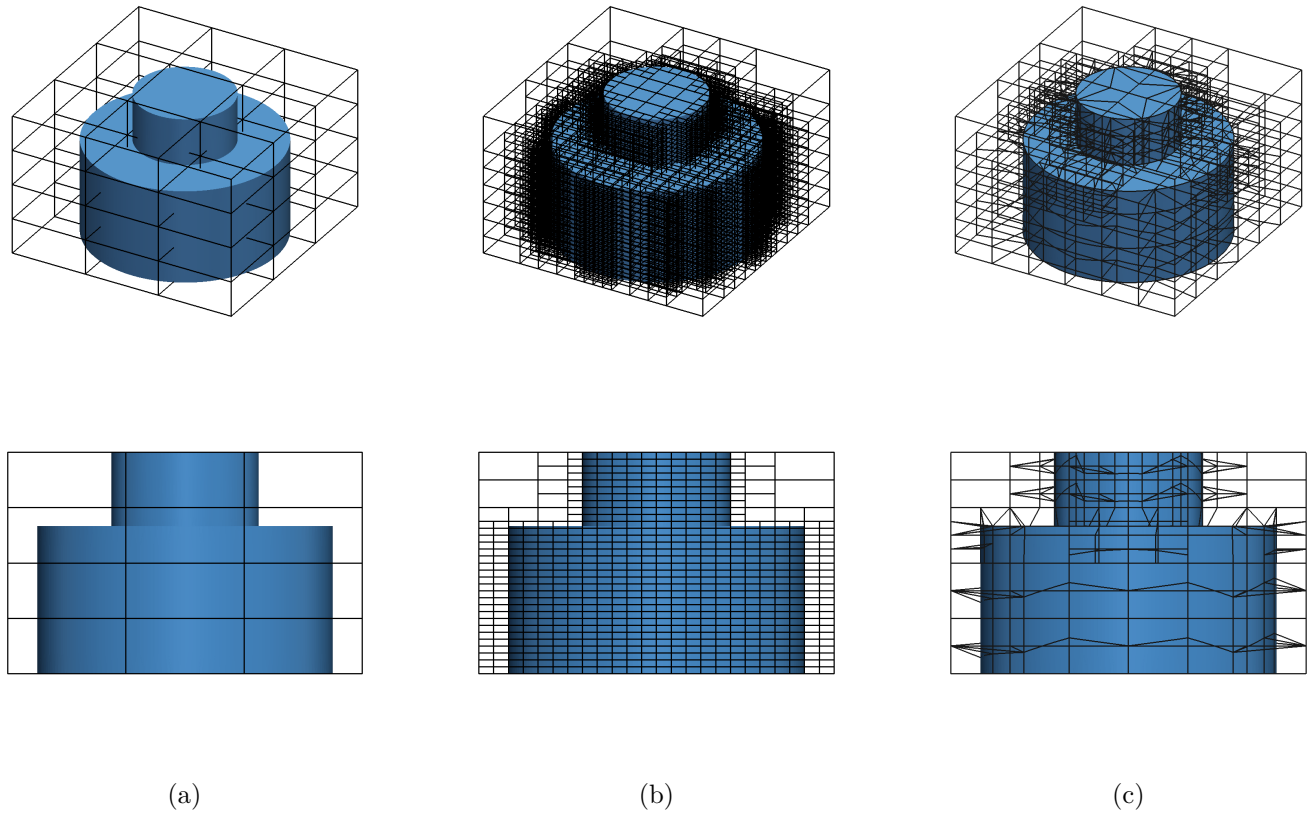
(a) (b)

(c) (d)

Figure 2.43: *Smart octrees generated on different background meshes: (a) single element, (d)* $4 \times 4 \times 4$ *elements, (c)* $3 \times 3 \times 3$ *elements, (d)* $4 \times 4 \times 4$ *elements. The edges of the initial mesh are marked with thick lines [17].*

cannot be integrated exactly by Gaussian quadrature. Therefore, instead of exact integration, the error in the volumetric integral is expected to converge towards zero when increasing the number of quadrature points. The lower and upper cylinder has a height of 50 and 25, while their radius is equal to their respective height. The cylinders are aligned parallel to the z axis of the coordinate system, and the center point of the bottom circular face is located at $(0, 0, 0)$. Integrating the expression in equation 2.57 on this geometric configuration yields the reference solution:

$$V_{ex} = 7975541.475533795 \tag{2.58}$$



(a) (b) (c)

Figure 2.44: *Example with nonlinear faces: (a) Ω_{phy} and background mesh, (b) octree with $k = 3$, (c) smart octree [17].*

Because some boundaries of Ω_{phy} are curved surfaces, the order elevation as explained on page 33 is employed. The polynomial order of the curved faces is chosen as $p_B = 2, 4, 8$. Figure 2.45 depicts a cell with a smooth cutting boundary and a cell with a topologically exceptional cut featuring sharp edges.

A comparison of the time required to generate the different integration meshes is depicted in Figure 2.46. As it is demonstrated in the figure, the standard octree algorithm becomes more expensive than the smart octree method from a subdivision depth $k = 3$. Apart from the standard point membership classification, the smart octree approach also needs to perform ray-surface intersections for the identification of the active nodes. As the octree approach

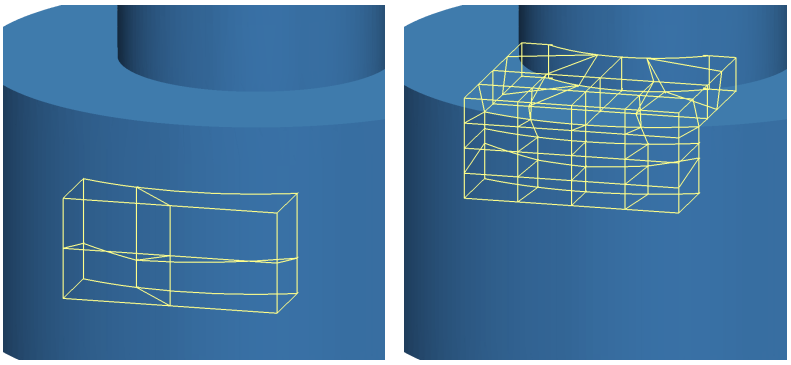


Figure 2.45: *Detailed view on two cells: (a) smooth boundary inside the cell, (b) topologically special case with sharp edges [17].*

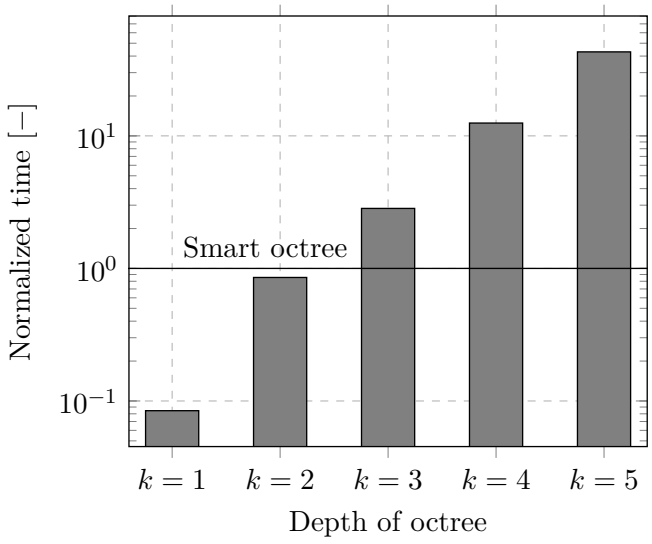
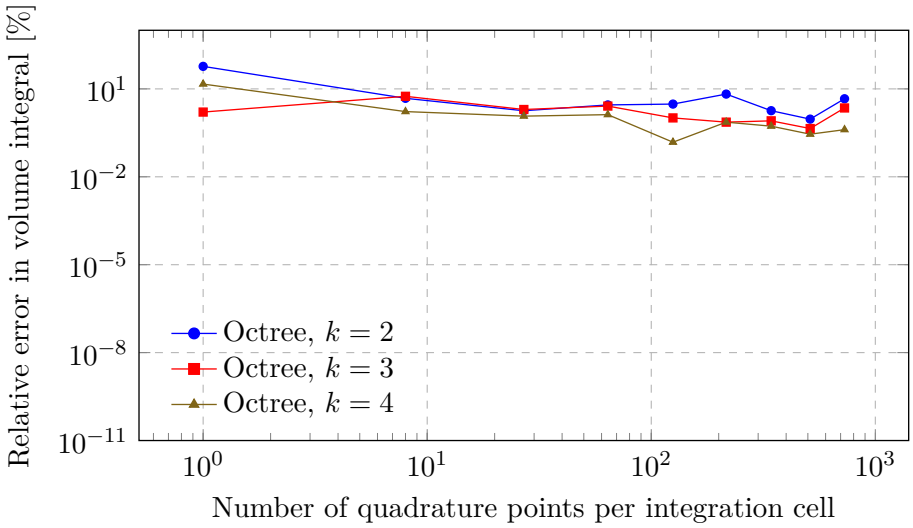
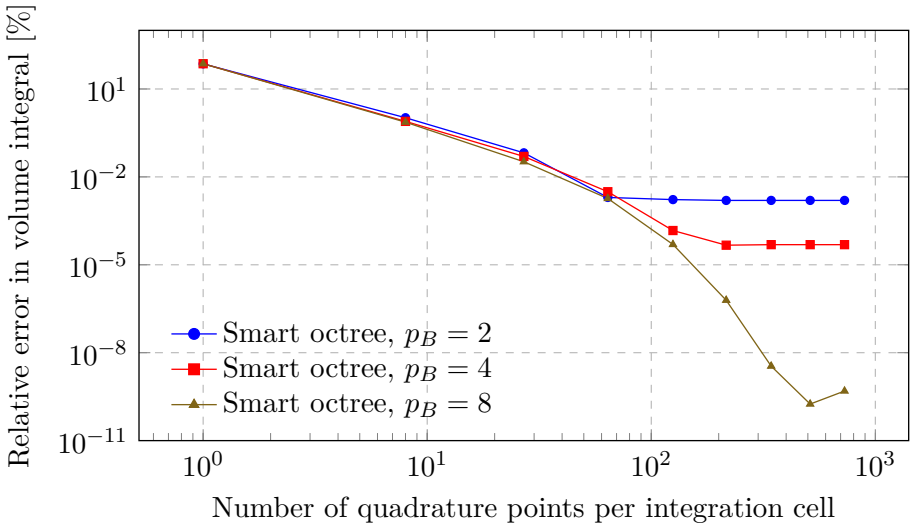


Figure 2.46: *Comparison of the time required for generating different integration meshes for the stacked cylinders [17].*

does not need this information from the geometric model, it remains cheaper for low levels of octree subdivision. However, in order to determine whether a subdivision is needed, the octree method has to evaluate the inside-outside state of seed points on every new level of subdivision. As a consequence, the generation of classical octree-based integration meshes becomes significantly more expensive than the proposed algorithm for higher levels of subdivision. Similarly to the previous study, the order of integration is increased stepwise by distributing $n \times n \times n$ quadrature points per subcell, with $n = 1 \dots 9$. The error in the volume integral defined by equation 2.55 is plotted in Figure 2.47.



(a) Octree with different maximum levels of subdivision k



(b) Smart octree with different orders of surface approximation p_B

Figure 2.47: *Convergence of the error in volume w.r.t. the number of quadrature points per integration cell, for the cylindrical example. The order of the integration is stepwise increased by distributing $n \times n \times n$ quadrature points in every integration cell, with $n = 1..9$ [17].*

Concerning the octree approach, increasing the maximum depth k does not lead to a clear

improvement in the accuracy. Because the octree-based integration cells on the lowest level are still cut by the domain boundary, the quadrature points distributed on these cells are located on both sides of the interface. When increasing the number of integration points in the subcells, the ratio of quadrature points of the cell lying in Ω_{fict} and Ω_{phy} changes, which leads to the oscillations of the convergence curves of the octree approach. It is noted here that this phenomenon could be avoided if a local triangulation is applied on the lowest subdivision level as explained in [58].

The integration error of the smart octree approach, however, can be controlled by the polynomial order of the surface approximation, p_B. As expected, depending on the order of the mapping, the convergence curves level off at a certain value. Here, the order of the numerical integration becomes comparable to the order of the surface approximation. Clearly, as p_B increases, more and more quadrature points are needed for reaching the minimum error level. However, the smart octree approach does not show the oscillatory behavior observed in the case of the octree approach and allows to reach smaller errors at the same time. Its advantage over the octree method becomes even more apparent when comparing them on the basis of the total number of quadrature points employed for the integration. This relationship is plotted in Figure 2.48. Here, it can be observed that the smart octree-based integration allows for much lower error values, with orders of magnitude less integration points compared to the octree method.

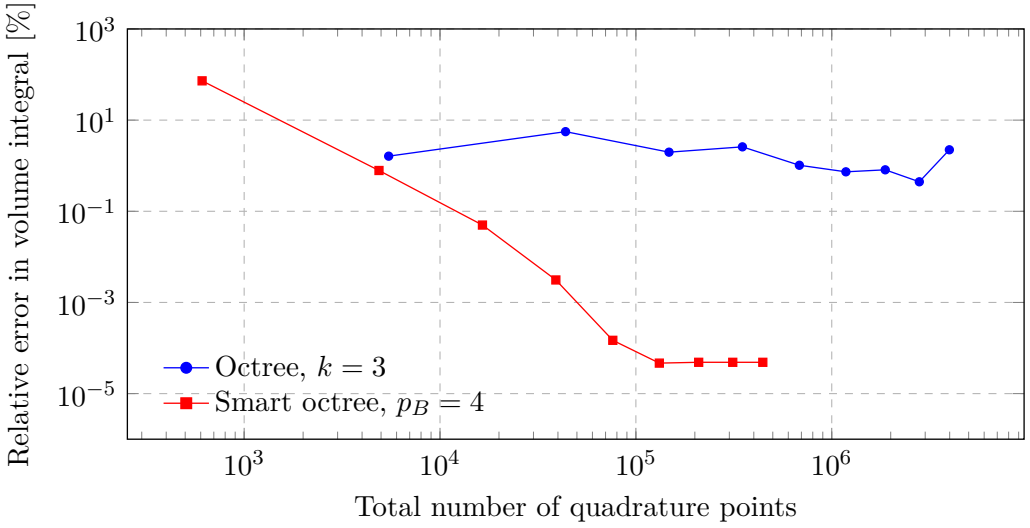


Figure 2.48: *Comparison of the classical octree method and the smart octree approach on the basis of the total number of quadrature points, for the cylindrical example. The order of the integration is increased stepwise by distributing $n \times n \times n$ quadrature points in every integration cell, with $n = 1..9$ [17].*

The overall effort of computing the domain integral is strongly linked to the total number of quadrature points, because the integrand has to be evaluated at each of them. Therefore, if their number is reduced, the number of function evaluations decreases, which brings a reduction in the time required to evaluate the domain integral as well. This is demonstrated in Figure 2.49, where the total integration time (including the setup of the integration subcells) of the octree and smart octree approaches are compared. Due to the significantly smaller

number of integration points, the proposed method computes the domain integral with orders of magnitude less time than the standard octree technique.

Note that for the case with $p_B = 8$, the error curves level off at 10^{-10}. This is not a limitation of the smart octree algorithm, but is connected to the precision of the underlying geometric kernel that performs the intersection and point projection operations.

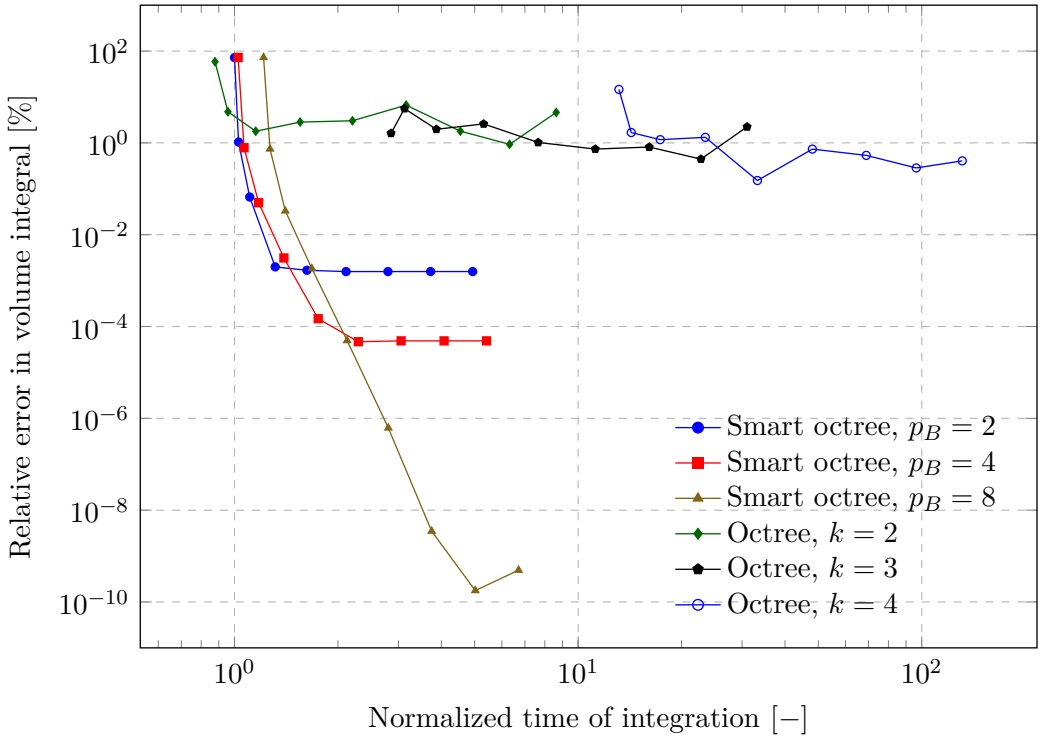


Figure 2.49: *Comparison of the total integration time of different integration approaches for the stacked cylinder example. The order of the integration is increased stepwise by distributing $n \times n \times n$ quadrature points in every integration cell, with $n = 1..9$ [17].*

Smart octree in combination with the Finite Cell Method

The previous examples showed that the smart octree approach is well suited to approximate the volume of complex-shaped domains. The next examples will demonstrate that the same idea can also be applied to solve partial differential equations numerically on a non-conforming discretization. For this purpose, the introduced smart octree technique is combined with the FCM.

The first example aims at assessing the proposed combination of methods by means of an analytical benchmark. For this purpose, an octant of a hollow sphere is considered (see Figure 2.50). On this domain Ω_{phy} we solve the Poisson equation:

$$\Delta u - f = 0, \tag{2.59}$$

where u is an unknown field value. Homogeneous Neumann boundary conditions are applied on every surface of $\partial\Omega_{\text{phy}}$ and the source term f is chosen as:

$$f = \frac{192 \cdot r}{125} + \frac{48}{5 \cdot r} - \frac{216}{25}, \tag{2.60}$$

with

$$r = \sqrt{x^2 + y^2 + z^2}. \tag{2.61}$$

The analytical solution reads:

$$u = -\frac{16 \cdot r^3}{125} + \frac{36 \cdot r^2}{25} - \frac{24 \cdot r}{5} + 5. \tag{2.62}$$

Further, the exact value of the internal energy is:

$$U_{ex} = \frac{12\pi}{7}. \tag{2.63}$$

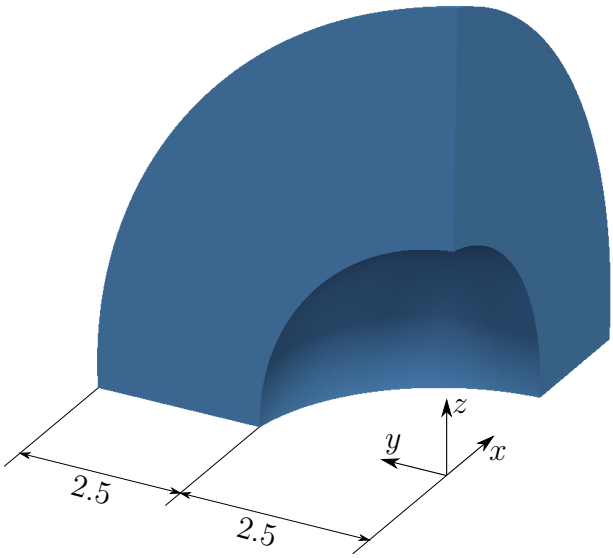


Figure 2.50: *Geometric domain of the hollow sphere example [17].*

The embedding domain $\Omega_\cup$ is discretized into $3 \times 3 \times 3$ finite cells. Both steps of the smart octree algorithm are applied on the finite cell mesh, with varying levels of geometry approximation ($p_B = 2, 4, 8$). The resulting mesh of 230 integration cells is depicted in Figure 2.51.

The polynomial order of the shape functions in the finite cells is increased from 1 to 7. The number of integration points per subcell is chosen as $(p+1)^3$ and $(p+3)^3$ for the octree and the smart octree integration meshes, respectively. The reason for more points distributed in the case of the smart octree is that the quadrature has to account for the high-order mapping functions of the curved subcells, as demonstrated by the example on page 46. The accuracy of the methods is assessed by computing the relative error in the energy norm:

$$e = \sqrt{\frac{|U_{ex} - U_{num}|}{U_{ex}}}, \tag{2.64}$$

where U_{num} is the numerically computed value of the internal energy.

Figure 2.52 depicts the error in energy norm with respect to the number of degrees of freedom. Both partitioning methods show exponential convergence, similar to p-FEM. However,

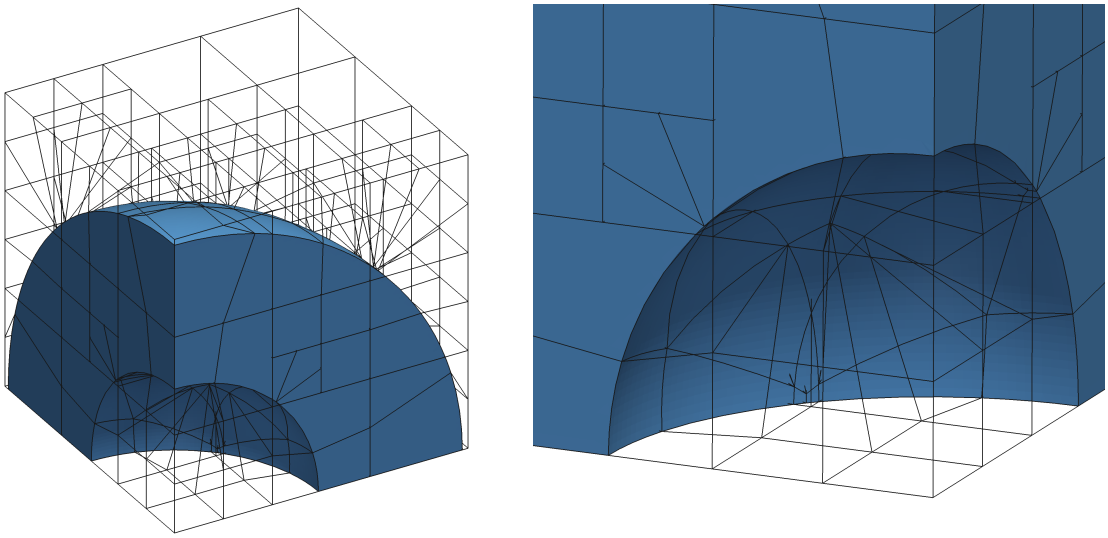


Figure 2.51: *Integration cells on an octant of a hollow sphere [17].*

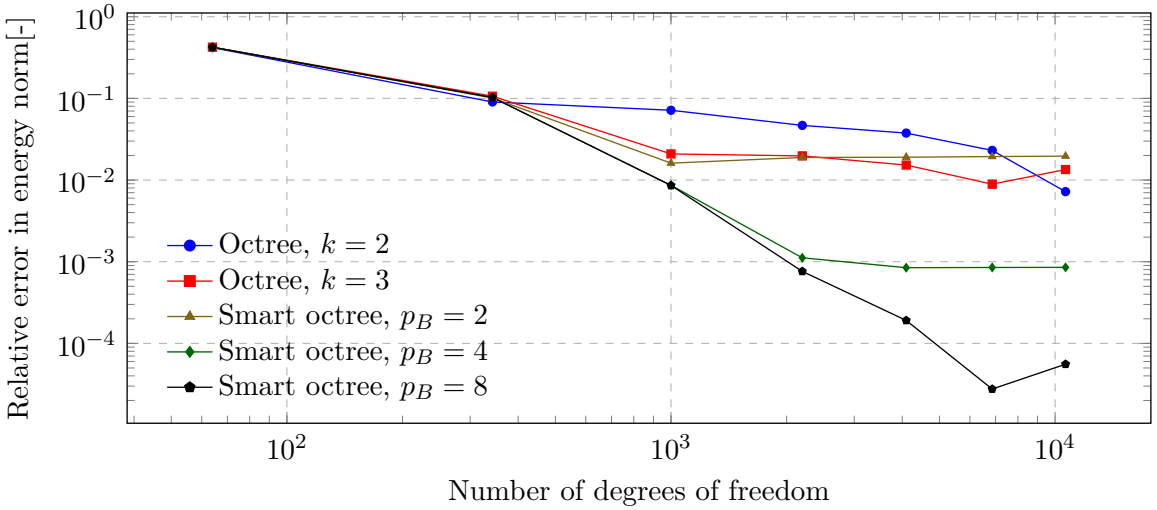


Figure 2.52: *Error in energy norm w.r.t. the number of degrees of freedom for the hollow sphere example [17].*

all the convergence curves level off at a certain error value. At the point of leveling-off, the integration error starts to dominate over the discretization error, which renders a further increase of the ansatz orders pointless.

Concerning the octree approach, the integration error can be reduced by adding more levels of refinement, as shown by the curves representing the octree with $k = 2$ and $k = 3$ refinement levels. For the smart octree approach, the accuracy can be improved by increasing the polynomial order of the geometry approximation. In comparison to the octree approaches, this yields a significant reduction in the error.

The advantage of the smart octree-based integration becomes even more pronounced when considering the relationship of the error and the total number of quadrature points. As depicted in Figure 2.53, starting from an error level of $\sim 10^{-2}$ the smart octree-based integration needs about one order of magnitude less integration points than the octree approach. This leads to a significantly reduced computational effort for the simulation.

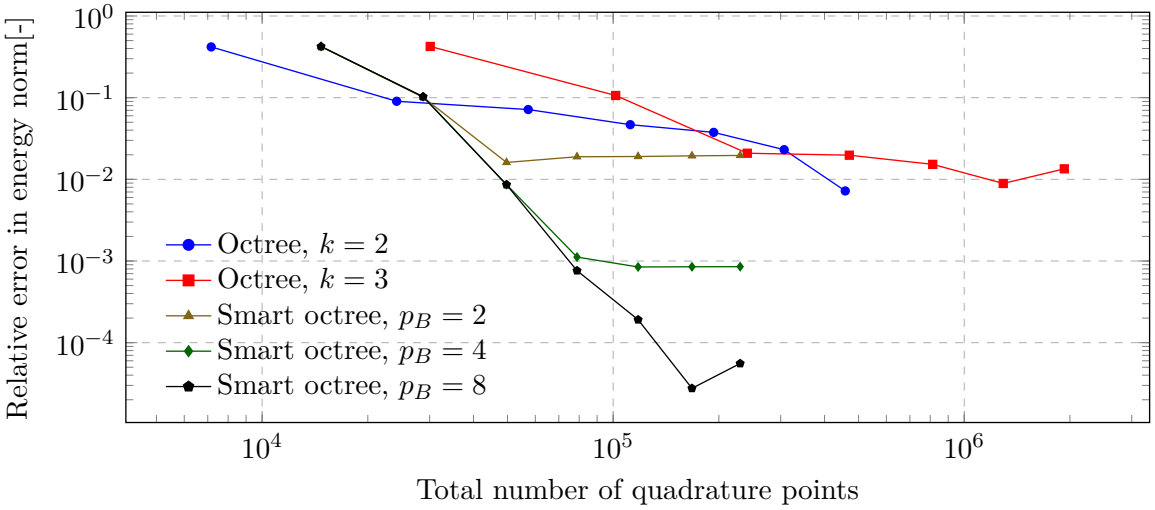
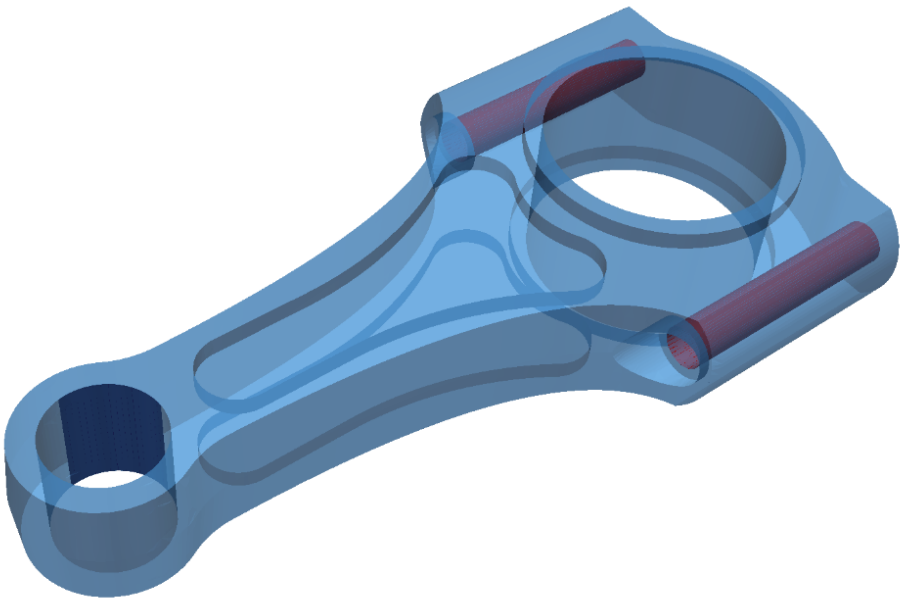


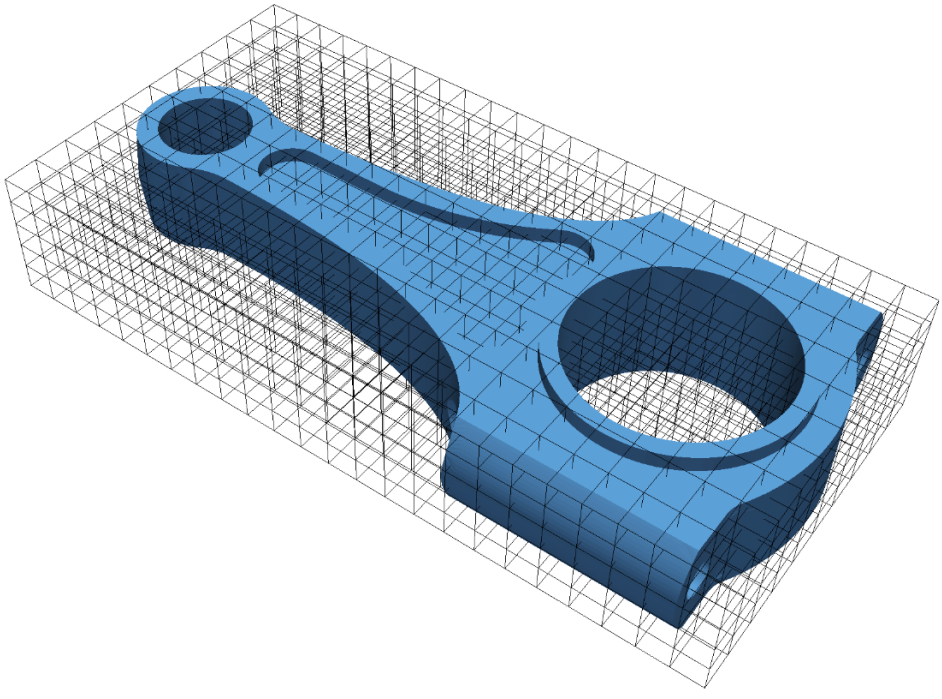
Figure 2.53: *Error in energy norm w.r.t the number of quadrature points for the hollow sphere example [17].*

The final example demonstrates that the method is also capable of resolving non-trivial engineering geometries. To this end, a connecting rod from a piston engine is considered [59], as depicted in Figure 2.54. The part is embedded by a fictitious domain which is discretized into $10 \times 24 \times 5$ finite cells. The polynomial order of the shape functions is $p = 5$. The constitutive relationship is assumed to be linear elastic. Constant pressure boundary condition is applied on the upper cylindrical hole and homogeneous Dirichlet boundary conditions constrain the two lower cylindrical holes (see Figure 2.54). On this configuration, an integration mesh was generated by the smart octree method, with a maximum subdivision depth $k_{max} = 4$. The octree refinement step was applied for 15% of the cells and the maximum refinement depth k_{max} of the octree subdivision was reached in 1% of the cells in the background mesh. In this small portion of cells the *fallback strategy* was applied, computing the integral on the leaves of the standard octree. The error of volume computed on the generated integration mesh is in the range of 0.1% when compared to the value computed by the commercial BREP software *Rhinoceros*, which computes the volume on a fine tessellation of the geometric model, employing the divergence theorem.

The integration mesh generated by the smart octree partitioner is depicted in Figure 2.55. Figure 2.56 provides a detailed illustration of the subcells lying inside the geometric domain. Figure 2.57 shows the deformed geometry as well as the von Mises stresses. When compared to a reference value of the strain energy which was obtained by an "overkill" FCM solution, the associated error in energy norm is in the range of 10%. This value is in the expected order of magnitude, because there is no local refinement of the mesh defining the shape functions around singularities along the edges representing reentrant corners.



(a)



(b)

Figure 2.54: *Connecting rod: (a) geometry and boundary conditions, (b) finite cell mesh. The surfaces of Neumann and Dirichlet boundary conditions are shown in dark blue and red color, respectively [17].*

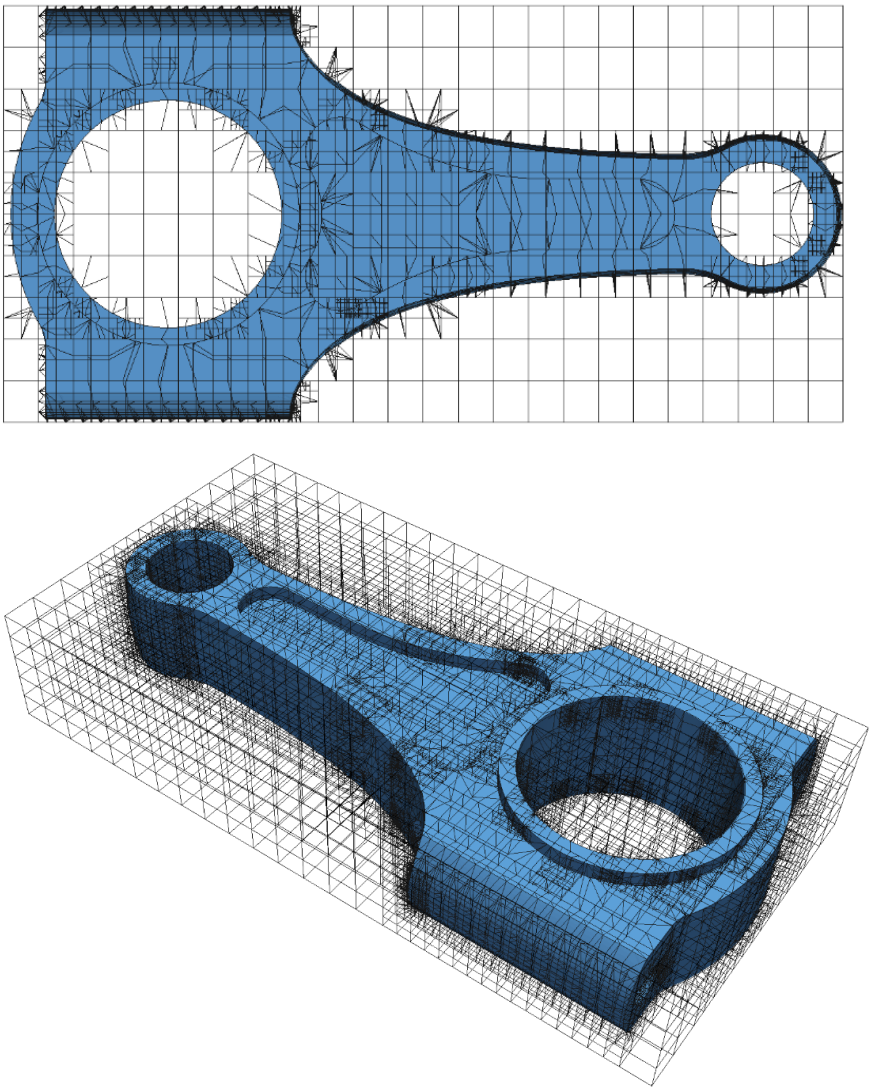


Figure 2.55: *Smart octree generated on the connecting rod [17].*

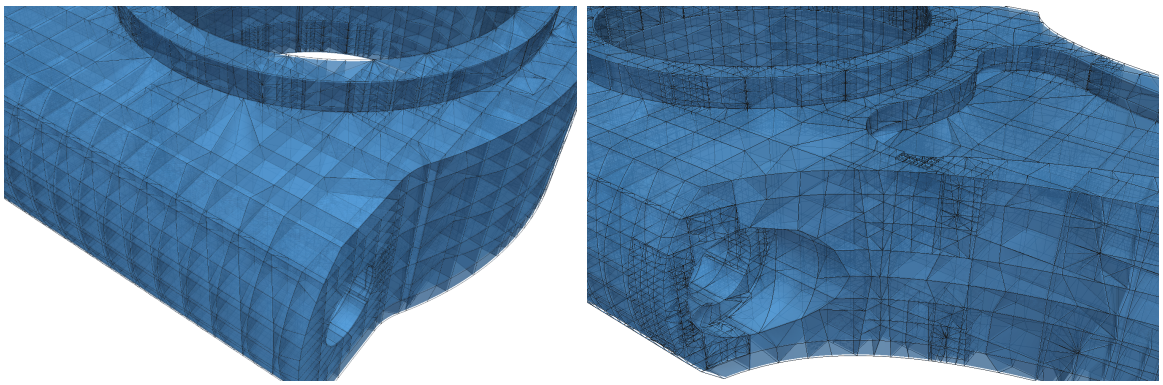


Figure 2.56: *Detailed view of the smart octree, only showing octants that lie inside the domain of computation [17].*

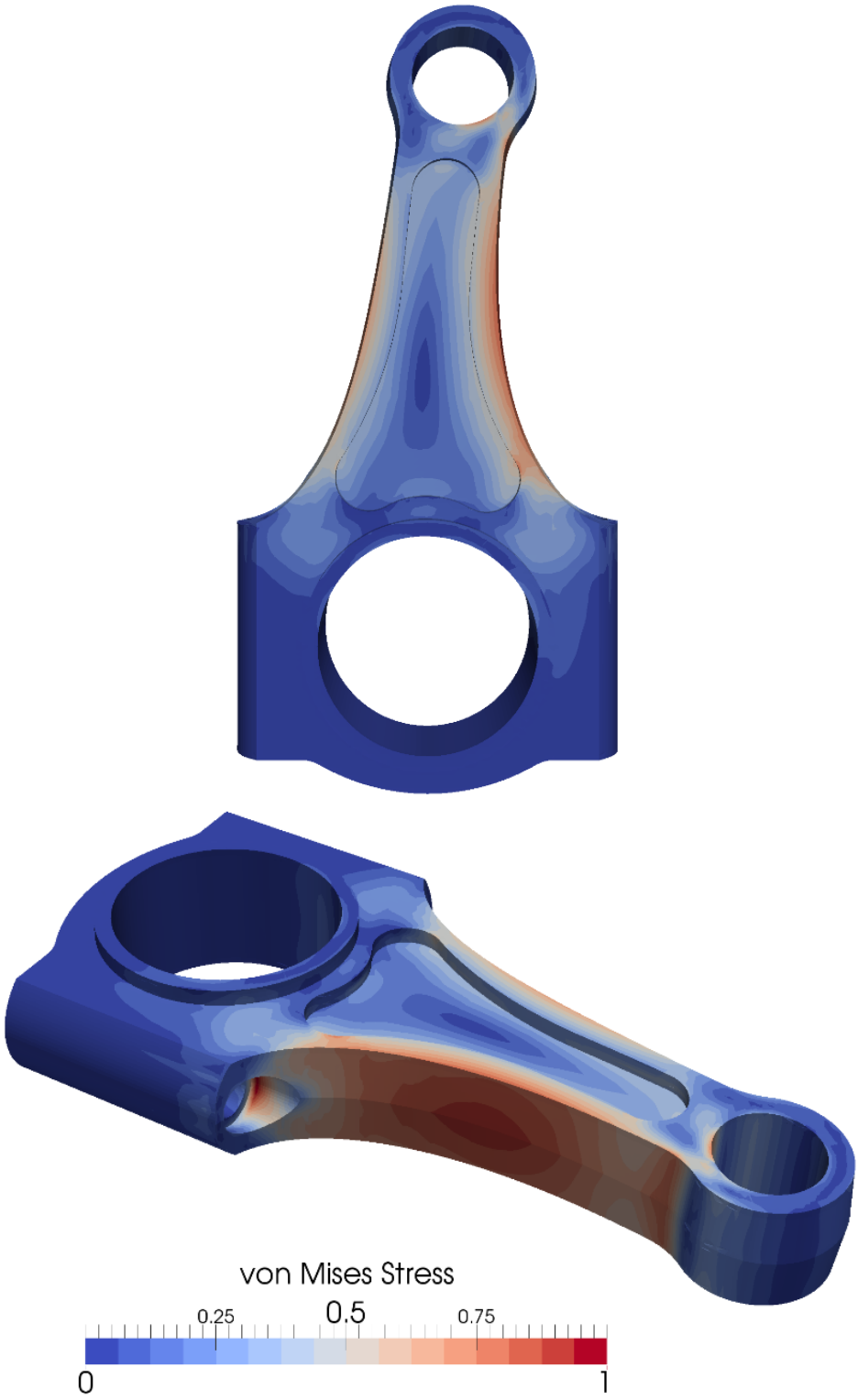


Figure 2.57: *Von Mises stress contours [17].*

Chapter 3

Image-based shape measurement and mesh generation

As seen in Chapter 2, both the FEM and the FCM rely on the availability of a digital geometric model, either for mesh generation or for the construction of the indicator function $\alpha(\boldsymbol{x})$. Although in most cases there is a digital model coming from a computer aided design software, there are application fields where such CAD models are not directly available. This situation typically arises when the geometries of interest represent "natural" shapes, e.g. biomechanical structures, landscapes, or cultural heritage artifacts, such as historical buildings and statues. In this context, the object under consideration needs to be digitized and converted into a CAD model.

The aim of this chapter is to give an overview over the possible approaches for solving this task. The first part focuses on shape acquisition techniques, which serve the purpose of recording the surface of an object of interest in the form of point clouds. Thereby, special attention is given to photo-based techniques. After outlining the essential ideas of these methods, two special cases are discussed: a formulation for shape acquisition from pictures recorded by moving a camera on a circular path, and a specialization of this formulation for the setting when the object of interest is immersed into a liquid with known refractive coefficients. The second part discusses the common steps taken to convert a point cloud into a geometric model and finally a finite element mesh.

3.1 Overview of digital shape acquisition techniques

The primary goal of digital shape acquisition is to measure the actual shape of a physical object, providing metrological information such as size, form, location and orientation [60]. This is achieved by collecting data about the surface at certain points or regions. The acquired data, a *"series of closely separated points on the surface"* [61] is usually stored by a set of three-dimensional coordinates, *point clouds*, allowing for further processing by computers.

Different possibilities are available to acquire the shape of a physical object. In essence, every shape acquisition method relies on some mechanism or phenomenon for interacting with the surface or volume of the object of interest. Depending on the physical principle that governs this interaction, data acquisition methods can be divided into two main categories: tactile and non-contact methods [62].

3.1.1 Tactile methods

In tactile methods, the measurement principle is based on mechanical contact. To determine the shape of the object to be measured, a touch-sensitive probe is moved along the surface of interest using mechanical arms. When the object is touched by the probe, the position of the probe tip is determined by reading the data from the sensors mounted in the joints of the moving arms. By performing this process repeatedly, the dimensions and the shape of the object can be determined. A widely applied implementation of the contact-based shape measurement technique is the coordinate measuring machine (CMM), see e.g. [63].

Not all kinds of shapes can be measured by CMMs efficiently: the motion of the probe tip is limited by the mechanical constraints of the moving arms, which might cause the measurement of concave surfaces to be especially challenging [62]. Further, as the probe tip needs to be moved along the surface of interest point-by-point, the measurement speed of CMMs is low compared to non-contact devices [64].

3.1.2 Non-contact methods

The limitations of CMMs can be overcome by non-contact approaches. These methods usually rely on measuring and processing the intensity of energy carried by a physical wave that is either reflected by or transmitted through the object of interest.

Acoustic sensors use sound waves as the primary carrier of energy. Members in this category include sonar imagery, e.g. for the mapping of seafloors [65], medical ultrasound imaging [66], and ultrasonic non-destructive testing [67]. In these methods, the distance of an object point to the sensor is usually derived from the time required for a pulse of sound wave to bounce back from the object.

If energy is transmitted by means of electromagnetic waves, a further distinction can be made depending on whether the processing of the recorded signal uses the reflected waves coming from the surface of the object of interest, or transmitted ones, where the governing phenomena occur when the wave passes through the volume of the structure. Examples for *volumetric imaging* based on transmitted electromagnetic waves include X-rays and CT-scans.

If the recorded signal is predominantly composed of reflected electromagnetic waves with wavelengths in the spectrum of visible light, the shape measurement technique belongs to the category of *optical shape measurement* methods.

Optical shape measurements record the intensity of light rays reflected by the object on a digital sensor, and process the recorded information further to recover the shape of the surface interest. The wide range of available 3D optical shape measurement methods and their further categorization is beyond the scope of this work. A discussion on further distinguishing aspects and categories of these approaches can be found in e.g. [68].

In the remaining parts of the thesis, attention is focused on methods that rely on recovering points on the object surface by *optical triangulation* [69]. Triangulation refers to the process of deriving the 3D location of a point in space using trigonometric calculations on triangles formed by two known points and the unknown point to be computed.

Laser triangulators are widely used optical measurement devices that rely on the triangulation principle. They use an active illumination component, a laser source, to sweep a plane of light across the object to illuminate its surface. During the movement of the light plane, the scene is observed by a light-sensitive sensor from an offset viewpoint. The intersection

of the light plane with the object of interest forms a space curve whose projection on the camera sensor is recorded. Knowing the positions of the active light source and the camera, optical triangulation allows to infer the 3D location of the individual pixels on the recorded stripes [70].

To sweep the light plane across the scene, the optical system forming the light sheet needs to be moved. This motion limits the speed of scanning by laser triangulation. To decrease the time required for scanning, *structured light* scanners can be employed. The working principle of these sensors is also based on the principle of optical triangulation. However, instead of a single light sheet, they project two-dimensional patterns of non-coherent light onto the object. Thus, the light plane is replaced by a bundle of planes covering a larger surface area. In order to guarantee that the individual planes and their projections can be uniquely identified, different projection patterns have been developed, see [71, 72].

Laser triangulation and structured light scanning fall in the category of *active* optical shape acquisition methods, as they use an external light source to illuminate the surface of the object. However, not all optical scanning methods necessarily require an active illumination of the object surface. The most prominent examples for such *passive* approaches are based on *stereo vision*. Here, the light source is replaced by another CCD sensor, such that the scene is simultaneously viewed from two different viewpoints. Due to the distance between the two sensors, the projection of a point on the first camera is different from its projection on the second camera. The difference between the locations–the *disparity*–allows for recovering the distance of the point to the sensors. As stereo vision records the object from two viewpoints, only a partial reconstruction of the object is possible. To recover more complete models, the idea of stereo vision can be extended to configurations with multiple viewpoints, leading to *multi-view stereo* methods. These allow for a more complete recovery of 3D models, from multiple images.

3.2 Photogrammetric acquisition of surface points

In the following, the process of acquiring 3D points of an object from 2D images is reviewed. Starting from the basic properties and mathematical models associated to the imaging process, the principal formulation of multi-view stereo reconstruction is recalled. This is followed by an introduction to the bundle adjustment formulation, an important concept in structure-from-motion, i.e. simultaneously recovering camera positions and structure points from image correspondences. Having formulated the bundle adjustment algorithm, two extensions of the approach are introduced: a constrained formulation allowing for reconstructing objects that were recorded under a circular motion of the camera, and a modified bundle adjustment formulation that can be employed for scenes that were recorded in refractive environments. The aim of this section is to provide a short introduction into these topics. For an in-depth discussion, the reader is referred to one of the textbooks in this field, such as [73] or [74].

3.2.1 Overview of the imaging process

When taking a picture with a camera, a 2D projection of the 3D world is formed. In this process, *depth* information, i.e. the distance between the camera and the points seen by the camera is lost. The imaging process is usually modeled as a *central projection*, in which a

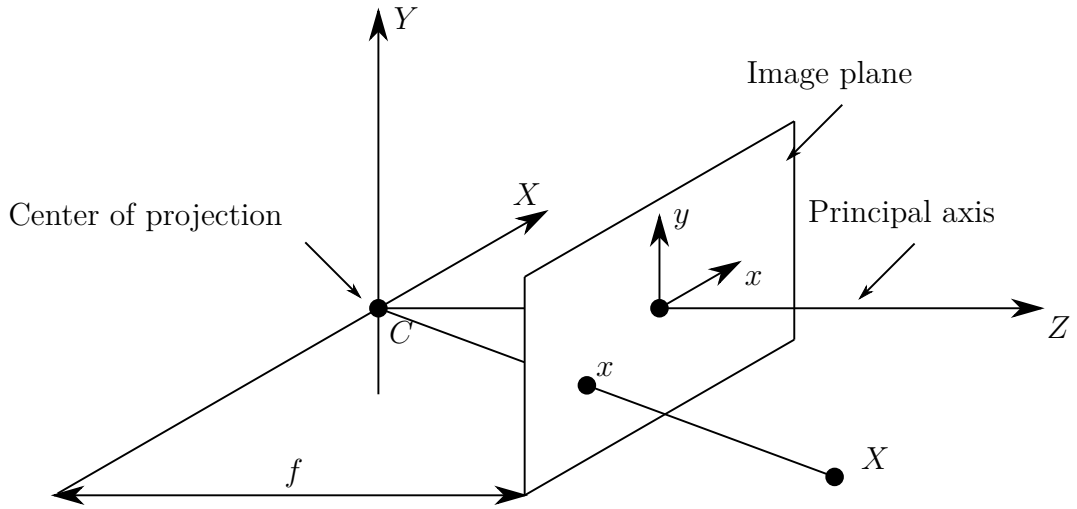


Figure 3.1: *Pinhole camera model. The center of projection is represented by the point* $\boldsymbol{C}$ *and the image plane is located at* $Z = f$*. The projection of the 3D point* $\boldsymbol{X}$ *is determined by the point of intersection between the image plane and the ray connecting* $\boldsymbol{X}$ *and* $\boldsymbol{C}$*.*

ray is drawn from a 3D world point towards a fixed point in space attached to the camera, the *center of projection*. This ray intersects the so-called *image plane* at a single location, which will form the image of the 3D point. This model corresponds to the simple camera configuration where light rays from the world are passing through the lens of the camera and land on a light-sensitive film or digital sensor.

The mathematical machinery that helps to formulate and eventually revert the imaging process is given by the tools of *projective geometry*. The simplest optical system which is used for modeling the projective behavior of cameras is the *pinhole camera model*.

Let the center of projection be located at the origin of the Euclidean coordinate system, and the plane defined at $Z = f$ be the *image plane*. Here, f denotes the focal distance of the camera. Using the pinhole camera model, the projection of a point $\boldsymbol{X} = [X, Y, Z]^{\mathrm{T}}$ in the 3D world coordinate system onto the image plane is the point of intersection between the image plane and the line that joins $\boldsymbol{X}$ with the center of projection. This projective relationship is depicted in Figure 3.1.

By similar triangles, one finds that projection of $\boldsymbol{X}$ on the image plane is given by:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ f \end{bmatrix}. \tag{3.1}$$

Omitting the last element yields the 2D coordinates of the projected point:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \end{bmatrix}, \tag{3.2}$$

which describes the mapping from world ($\mathbb{R}^3$) to image ($\mathbb{R}^2$) coordinates. The center of projection is often referred to as *camera center* or *optical center*. The ray starting from

the camera center perpendicular to the image plane (the Z axis in Figure 3.1) is called the *principal axis* or *optical axis*. The point where the principal axis intersects the image plane is the *principal point*.

The relation in Equation 3.2 can be expressed using homogeneous coordinates using a matrix-vector product:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.3}$$

Using the notation

$$\boldsymbol{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{3.4}$$

Equation 3.3 can be expressed as:

$$\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}. \tag{3.5}$$

The matrix in the expression above may be written as:

$$\boldsymbol{P} = \mathrm{diag}(f, f, 1)\,[\boldsymbol{I}|0]\,, \tag{3.6}$$

where $\mathrm{diag}(f, f, 1)$ represents a diagonal matrix, and $[\boldsymbol{I}|0]$ is a 4×3 matrix, divided into two blocks, the 3×3 identity matrix and a 3×1 column vector, which, in this case is the zero vector. The matrix $\boldsymbol{P}$ is referred to as the *camera projection matrix*.

Equation 3.3 assumes that the origin of the image coordinate system coincides with the principal point. However, this is not necessarily always the case, especially for images recorded on CCD sensors. In this situation, the origin of the image coordinate system is typically defined at the upper left corner of the image, with the x and y axes pointing right and downwards, respectively. To account for the offset between the origin of the coordinate system and the principal point, the third column in the matrix $\boldsymbol{P}$ carries the *principal point offset* p_x and p_y values:

$$\boldsymbol{P} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.7}$$

It follows that the projection matrix can be written as $\boldsymbol{P} = \boldsymbol{K}\,[\boldsymbol{I}|0]$, where the matrix $\boldsymbol{K}$ is:

$$\boldsymbol{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.8}$$

The matrix above contains the intrinsic properties of the camera and is thus usually referred to as the *camera intrinsic matrix*. The expression for the projection matrix in Equation 3.7

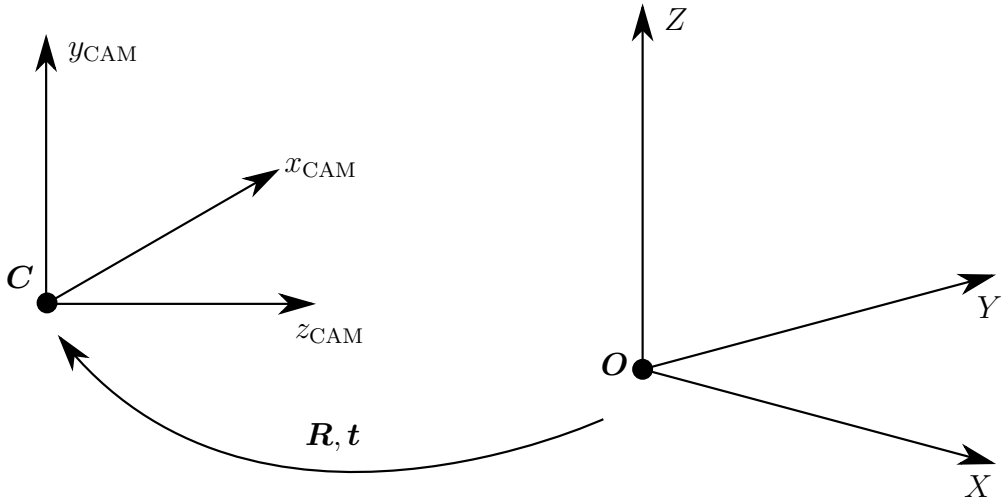


Figure 3.2: *Transformation between world- and camera coordinates.*

holds under the assumption that the world and image coordinates have equal scales in both axial directions. In the case of CCD cameras, however, a recorded image is usually stored in *pixel* coordinates. Therefore, in order to map the image from the world coordinates to pixel coordinates, the intrinsic matrix needs to be multiplied by an additional diagonal matrix $\mathrm{diag}\,(m_x, m_y, 1)$. Here, the parameters m_x and m_y represent the number of pixels per unit distance in image coordinates in the x and y directions, respectively. In the most general form, the intrinsic matrix of a CCD camera reads as [73]:

$$\boldsymbol{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.9}$$

where $\alpha_x = f m_x$ and $\alpha_y = f m_y$ are the focal distances in the x and y directions expressed in pixel units, and, similarly, $x_0 = m_x p_x$ and $y_0 = m_y p_y$ are the pixel coordinates of the principal point. Finally, s is the so-called *skew factor*, which is zero for most normal cameras.

In general, points in the 3D space are expressed in the world coordinate system, whose origin is not located at the principal point, and its axes are not necessarily parallel with the axes of the coordinate system fixed to the camera. Therefore, before the projective action of the camera represented by the matrix $\boldsymbol{K}$ is applied, points need to be transformed from the world coordinate system to the coordinate system attached to the principal point. This transformation can be expressed by a rotation followed by a translation. The overall setup of this a transformation is depicted in Figure 3.2.

Given a point $\boldsymbol{X}$ in world coordinates, its representation $\boldsymbol{X}_C$ in camera coordinates can be computed by:

$$\boldsymbol{X}_C = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R}\boldsymbol{C} \\ 0 & 1 \end{bmatrix}, \tag{3.10}$$

where $\boldsymbol{R}$ is a 3×3 rotation matrix representing the orientation of the camera in world coordinates, and $\boldsymbol{C}$ is the location of the camera center in world coordinates. Often, the camera center is not represented explicitly, but the following replacement is applied:

$$\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{C}. \tag{3.11}$$

It follows that the projection matrix of Equation 3.5 for a CCD camera located at $\boldsymbol{C}$ with an orientation $\boldsymbol{R}$ can be expressed as:

$$\boldsymbol{P} = \boldsymbol{K}\,[\boldsymbol{R}|\boldsymbol{t}]\,. \tag{3.12}$$

While the intrinsic matrix $\boldsymbol{K}$ contains the internal properties of the camera that are invariant under any translation or rotation, the term between the square brackets expresses those parameters that are dictated by such positional quantities. Therefore, the parameters $\boldsymbol{R}$ and $\boldsymbol{C}$ are usually referred to as *external parameters* or *exterior orientation*. For similar reasons, the matrix $[\boldsymbol{R}|\boldsymbol{t}]$ is called the *camera extrinsic matrix*.

The projection matrix $\boldsymbol{P}$ is of size 3×4 and condenses all the extrinsic and intrinsic properties into a single matrix representing a homogeneous transformation. It has 11 degrees of freedom, which is the same number as the degrees of freedom of a 3×4 matrix defined up to an arbitrary scale.

3.2.2 Multiple-view geometry

Multiple-view geometry (MVG) is the subject where relations between coordinates of feature points in different views are studied [75]. The main aim of MVG is to study the theory and methods relating to the structure and motion problem which is formulated in the following.

The structure and motion problem

Given a sequence of images with corresponding feature points $\boldsymbol{x}_{ij}$ taken by a set of perspective cameras, such that:

$$\boldsymbol{x}_{ij} = \boldsymbol{P}_i\boldsymbol{X}_j, \tag{3.13}$$

determine the camera matrices $\boldsymbol{P}_i$, and the 3D points $\boldsymbol{X}_j$, under different assumptions on the camera intrinsics and extrinsics.

In this problem statement, recovering the camera matrices is essentially equivalent to recovering the motion of the camera, while recovering the 3D points is equivalent to recovering the structure that the camera records during its motion. Hence the name of the problem: the structure and motion problem.

A basic building block of multiple-view geometry is the concept of *epipolar geometry*, a formalism that places constraints on the observations of structure points on different cameras. In this context, consider the images $\boldsymbol{x}$ and $\boldsymbol{x}'$ of the same point $\boldsymbol{X}$ on two different cameras, as depicted in 3.3. Obviously, the rays originating from the respective camera centers passing through $\boldsymbol{x}$ and $\boldsymbol{x}'$ intersect at $\boldsymbol{X}$. More importantly, the two rays are coplanar, and lie on the plane $\boldsymbol{\pi}$ that is defined by the two camera centers and the point $\boldsymbol{X}$. Supposing now that only $\boldsymbol{x}$ is known, the constraint on the location of $\boldsymbol{x}'$ is dictated by $\boldsymbol{\pi}$. This plane is always determined by the ray passing through $\boldsymbol{x}$ and the line connecting the two camera centers–the *baseline*. As previously discussed, $\boldsymbol{x}'$ needs to lie on $\boldsymbol{\pi}$, and therefore it has to be located on the intersection line $\boldsymbol{l}'$ between $\boldsymbol{\pi}$ and the second image plane. This line is also called the *epipolar line* corresponding to $\boldsymbol{x}$, which is the projection of the ray originating from the first camera associated to $\boldsymbol{x}$ onto the second image plane. The terminology associated to the epipolar geometry is depicted in Figure 3.3. The meaning of the different terms is:

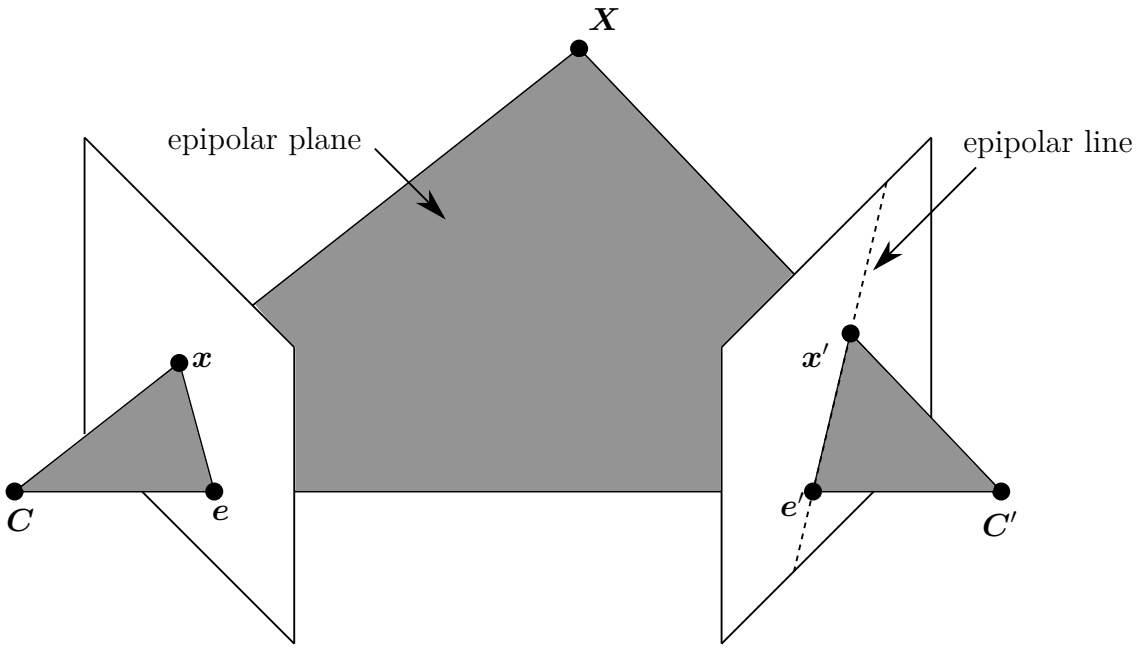


Figure 3.3: *Two cameras with centers* $\boldsymbol{C}$ *and* $\boldsymbol{C}'$ *viewing the same point* $\boldsymbol{X}$*. The epipolar plane is defined by the two camera centers and* $\boldsymbol{X}$*.*

- The *epipoles* $\boldsymbol{e}$ is the projection of the second camera center onto the image plane of the first camera. The other epipole $\boldsymbol{e}'$ is defined similarly.
- An *epipolar plane* is a plane containing the baseline, i.e. the line connecting the two camera centers. The set of epipolar planes forms a 1-parameter family.
- An *epipolar line* is the intersection of an epipolar plane with the image plane.

Given two camera views, there exists a 3×3 matrix $\boldsymbol{F}$, called the *fundamental matrix*, such that for any corresponding pair of points $\boldsymbol{x}$ and $\boldsymbol{x}'$, the following holds [73]:

$$\boldsymbol{x}'^{\mathrm{T}} \boldsymbol{F} \boldsymbol{x} = 0. \tag{3.14}$$

The importance of Equation 3.14 lies therein, that it provides a way to characterize the relationship between two camera views without explicitly using their projection matrices. Instead, the relationship between the views can be expressed in terms of corresponding image points. Therefore, the fundamental matrix can be computed from image correspondences alone. The fundamental matrix possesses the following properties:

- If $\boldsymbol{F}$ represents the fundamental matrix for a pair of cameras $\{\boldsymbol{P}, \boldsymbol{P}'\}$, the fundamental matrix of the same cameras in the opposite order $\{\boldsymbol{P}', \boldsymbol{P}\}$ is $\boldsymbol{F}^{\mathrm{T}}$.
- Given a point $\boldsymbol{x}$ in the "left" image, the corresponding epipolar line in the "right" image is given by $\boldsymbol{l}' = \boldsymbol{F}\boldsymbol{x}$. Conversely, the epipolar line in the "left" image for the point $\boldsymbol{x}'$ in the "right" image is $\boldsymbol{l} = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{x}'$.
- The epipoles on the left and right views satisfy $\boldsymbol{e}'\boldsymbol{F} = \boldsymbol{0}$ and $\boldsymbol{F}^{\mathrm{T}}\boldsymbol{e} = \boldsymbol{0}$, respectively.

Given two cameras represented by the projection matrices $\{\boldsymbol{P}, \boldsymbol{P}'\}$, the fundamental matrix can be computed as:

$$\boldsymbol{F} = [\boldsymbol{e}']_{\times} \boldsymbol{P}' \boldsymbol{P}^{+}, \tag{3.15}$$

where $\boldsymbol{P}^+$ is the pseudo-inverse of $\boldsymbol{P}$, and $\boldsymbol{e}' = \boldsymbol{P}'\boldsymbol{C}$.

While the formula above gives a way to compute the fundamental matrix from given camera projection matrices, it is equally important to be able to derive projection matrices from a given fundamental matrix. Even though a pair of camera matrices uniquely determines a fundamental matrix, this mapping is not one-to-one, as pairs of camera matrices that differ by a projective transformation result in the same fundamental matrix. Therefore, given $\boldsymbol{F}$, the respective camera matrices can be only determined up to a projective transformation. The camera matrices corresponding to a given fundamental matrix $\boldsymbol{F}$ may be chosen as [73]:

$$\begin{aligned} \boldsymbol{P} &= [\boldsymbol{I}|\boldsymbol{0}] \\ \boldsymbol{P}' &= \left[[\boldsymbol{e}']_\times \boldsymbol{F} | \boldsymbol{e}' \right]. \end{aligned} \tag{3.16}$$

Therefore, if the fundamental matrix is known for a pair of cameras, their projection matrices can be recovered, up to a projective transformation.

For image-based reconstructions, it is important to compute the fundamental matrix from image point correspondences. To compute the fundamental matrix, at least 7 matching points are required. Denoting matching points by $\boldsymbol{x} = [x, y, 1]$ and $\boldsymbol{x}' = [x', y', 1]$ and considering that $\boldsymbol{x}'^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x} = 0$, every point correspondence $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ gives rise to one equation in the unknown entries of $\boldsymbol{F}$. For one pair of corresponding points, the expansion of $\boldsymbol{x}'^{\mathrm{T}}\boldsymbol{F}\boldsymbol{x} = 0$ is:

$$xx'f_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0. \tag{3.17}$$

From this, when considering n matching pairs, a system of linear equations is obtained:

$$\begin{bmatrix} x_1x'_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx'_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \boldsymbol{f} = \boldsymbol{0}, \tag{3.18}$$

where $\boldsymbol{f}$ represents the entries of the fundamental matrix collected in a column vector in row-major order. For the computation of $\boldsymbol{F}$ the additional constraint $\det(\boldsymbol{F}) = 0$ may be used, i.e. its singularity property. Methods that solve the equation above under the singularity constraint can be found in [73].

Once the pair of camera projection matrices is known, the unknown structure points can be computed from the intersection of the back-projected rays associated to matching image features. The problem of recovering 3D point positions based on correspondences and known camera matrices is known as the *triangulation problem*. Triangulation is a well-studied question in the multi-view geometry literature, where various approaches are presented to tackle this problem, see e.g. [73, 74].

Given two views with camera matrices $\{\boldsymbol{P}, \boldsymbol{P}'\}$ and corresponding points $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$, there are essentially three main techniques for computing the location of the original 3D point $\boldsymbol{X}$:

- Minimization of the 3D error, i.e. finding the mid-point on the shortest line between the two back-projected rays originating from $\boldsymbol{x}$ and $\boldsymbol{x}'$, see [74].
- Minimization of the algebraic error: the two perspective projections $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}$ and $\boldsymbol{x}' = \boldsymbol{P}'\boldsymbol{X}$ give rise to four equations on the three unknown entries of $\boldsymbol{X}$. The resulting overdetermined system of equations can be solved by the *Direct Linear Transformation* (DLT) method, as discussed in [73].

- Minimization of the reprojection error: this method starts from the observation that corresponding points must lie on the same epipolar plane, which is a plane from the 1-parameter family of epipolar planes. Further, the corresponding epipolar lines also form a 1-parameter family. Thus, the task of finding the 3D point $\boldsymbol{X}$ that minimizes the reprojection error can be formulated as a single-parameter minimization problem. It turns out that the minimum can be found by solving for the real roots of a polynomial of order 6. More details on this approach can be found in [76].

Having formulated the most important ingredients of two-view reconstructions, the main steps of the solution process of the two-view structure and motion problem can be summarized as follows:

1. Starting from two views, compute corresponding image points $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$.
2. From the image correspondences, compute the fundamental matrix.
3. Compute the camera matrices from the fundamental matrix.
4. Compute the 3D points $\boldsymbol{X}$ from the correspondences and camera matrices by triangulation.

In order to establish $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$ correspondences, special points of interest, *keypoints* need to be extracted from the input images. This task is usually performed by feature extractors. Corners, sharp edges, highly textured regions classify as good keypoints. Feature extraction algorithms thus very often work on gradients of image intensities. Examples for feature extraction algorithms are the Harris corner detector [77], the scale-invariant feature transform (SIFT) [78], speeded up robust features (SURF) [79] and Oriented FAST and Rotated BRIEF (ORB) [80]. Very often, the feature detection and matching algorithm that is responsible for computing the correspondences returns false positives. These matches do not represent points that stem from the same 3D feature in reality. Such outlier points may draw the computation of the fundamental matrix towards a wrong solution. However, the detection and matching process usually provides much more than 7 matches per image pair. Furthermore, it can be assumed that the majority of the matching pairs represent true correspondences. To deal with the effect of the outliers, the computation of the fundamental matrix may be embedded in a RANSAC-based statistical fitting scheme, allowing for filtering out the effects of the wrong matches, as outlined in [75].

For the case of n-view reconstructions, the two-view process can be extended. In this setting, feature matches need to be determined amongst all the images taking part in the reconstruction. Once matches are available, an initial pair of views is selected. For the initial pair, the projection matrices and the corresponding structure points can be recovered following the two-view reconstruction steps described before. Then, additional views are added to the reconstruction in an iterative manner. In this process, for an additional view $i+1$, the pose towards the already existing reconstruction on i views is determined. To this end, the matches that correspond to 3D points that are already parts of the reconstructed scene are considered. Based on this, the projection matrix of $\boldsymbol{P}_{i+1}$ of the additional camera can be determined in a procedure similar to the computation of the fundamental matrix. By repeating this process until there is no remaining image to be added to the scene, the camera projection matrices are recovered for each view.

The general process of recovering the camera poses from a sequence of images is an extensively studied topic in the computer vision literature. Many shortcuts may be taken if prior

knowledge is available on the motion of the camera. For example, if images stem from a video sequence, it is a reasonable assumption that a feature appears only in nearby frames, which simplifies the searching process of feature matching. Examples of structure-from-motion algorithms working on image sequences may be found in e.g. [81–84].

3.2.3 Bundle adjustment

Following the steps explained in the previous section, the n-view structure and motion problem is solved, that is, the unknown structure points $\boldsymbol{X}_j$ and projection matrices $\boldsymbol{P}_i$ are found. Thus, ideally, the recovered quantities satisfy:

$$\tilde{\boldsymbol{x}}_{ij} = \boldsymbol{x}_{ij} = \boldsymbol{P}_i \boldsymbol{X}_j, \tag{3.19}$$

where $\tilde{\boldsymbol{x}}_{ij}$ is the observed position of the structure point $\boldsymbol{X}_j$ in the i-th image. In the usual case, however, the equation above is not satisfied exactly, due to the presence of measurement noise. The geometric error that corresponds to the distance between the projection of $\boldsymbol{X}_j$ and its observation $\tilde{\boldsymbol{x}}_{ij}$ is referred to as the *reprojection error*, defined as:

$$d(\tilde{\boldsymbol{x}}_{\boldsymbol{ij}}, \boldsymbol{x}_{ij}) = d(\tilde{\boldsymbol{x}}_{\boldsymbol{ij}}, \boldsymbol{P}_i \boldsymbol{X}_j), \tag{3.20}$$

where $d(.,.)$ denotes the Euclidean distance. In order to obtain a better reconstruction of the structure and the cameras, we wish to find a configuration of the projection matrices and structure points that minimizes the sum of the squares of the reprojection errors:

$$\min_{\boldsymbol{P}_i, \boldsymbol{X}_j} \sum_i \sum_j d(\tilde{\boldsymbol{x}}_{ij}, \boldsymbol{P}_i \boldsymbol{X}_j)^2. \tag{3.21}$$

This non-linear least-squares problem is known in the image processing literature as the *bundle adjustment problem* [85], as it aims at adjusting the bundle of rays from each camera center towards the structure points.

As discussed before, every camera has 11 degrees of freedom, while every structure point contributes with 3 parameters to the bundle adjustment problem. Thus, the total number of parameters to be minimized is $11m + 3n$, where m is the number of views and n is the number of structure points. Clearly, this results in a rather large minimization problem. For example, for a set of 30 views with 1,000 points/view the number of design parameters is over 90,000. Therefore, a direct solution becomes computationally infeasible. Fortunately, the special structure of the minimization problem can be exploited, reducing the computational effort tremendously. The key insight here is that an observation $\tilde{\boldsymbol{x}}_{ij}$ only depends on one structure point $\boldsymbol{X}_j$ and a single camera $\boldsymbol{P}_i$. It follows that most of the partial derivatives involved in the normal equations of the minimization problem are zero, which results in sparse Jacobian and Hessian matrices. Efficient solution techniques based on this sparse formulation can be found in [73, 74, 86]. The most popular algorithm of choice is the sparse variant of the Levenberg-Marquardt algorithm, also known as the Damped Least Squares method.

3.2.3.1 Constrained bundle adjustment

In certain cases, the number of parameters involved in bundle adjustment can be reduced, if prior knowledge about the scene structure is available. This situation arises for example if

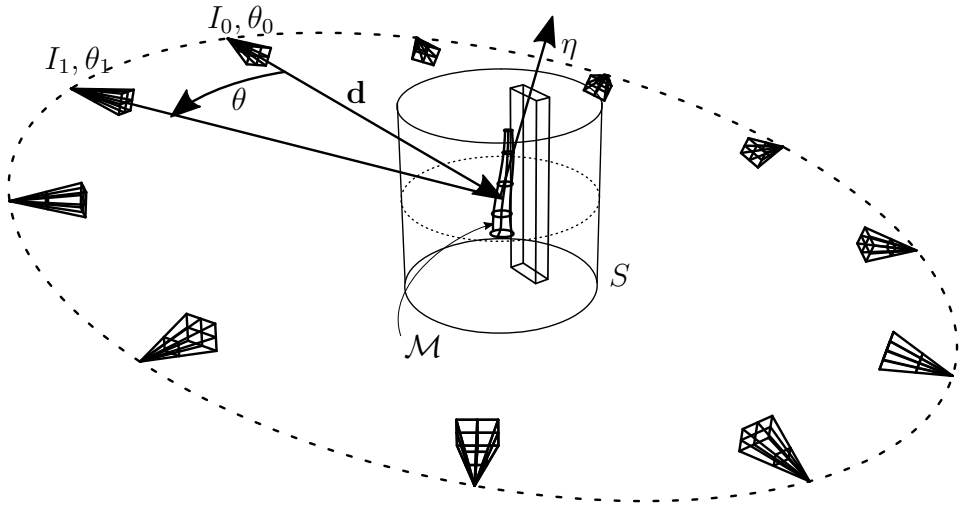


Figure 3.4: *Geometric setup for constrained bundle adjustment. The camera moves around the object of interest $\mathcal{M}$ along a prescribed circular path. The rectangular box in the vicinity of the object represents objects that do not belong to $\mathcal{M}$ and may interfere with the reconstruction [18].*

the camera is moved around the object of interest on an exact circular path, as depicted in Figure 3.4. In this case, the constraint on the movement of the camera allows to cast the extrinsic parameters into a more convenient form with less degrees of freedom, following the idea in [87].

Here, each camera is associated to a unique rotation angle θ_i around a common rotation axis, defined by a vector η and a translation vector $\mathbf{d}$. The rotation axis η is represented in spherical coordinates:

$$\begin{aligned} &\eta = [\sin(\alpha)\cos(\beta), \sin(\alpha)\sin(\beta), \cos(\alpha)]\,, \\ &\alpha \in [0, \pi]\,, \beta \in [0, 2\pi]\,. \end{aligned} \tag{3.22}$$

This way, there are 5 degrees of freedom associated to the circular path and the position of each camera is determined by a single degree of freedom: its angle around the rotation axis. Thus, the entire motion of the camera can be described by a total of $5 + n$ unknowns $\mathbf{M} = \{\mathbf{d}, \alpha, \beta, \theta_0, ..., \theta_n\}$. If the camera intrinsics do not change during the motion, the total number of parameters involved in the minimization problem is $10 + n$.

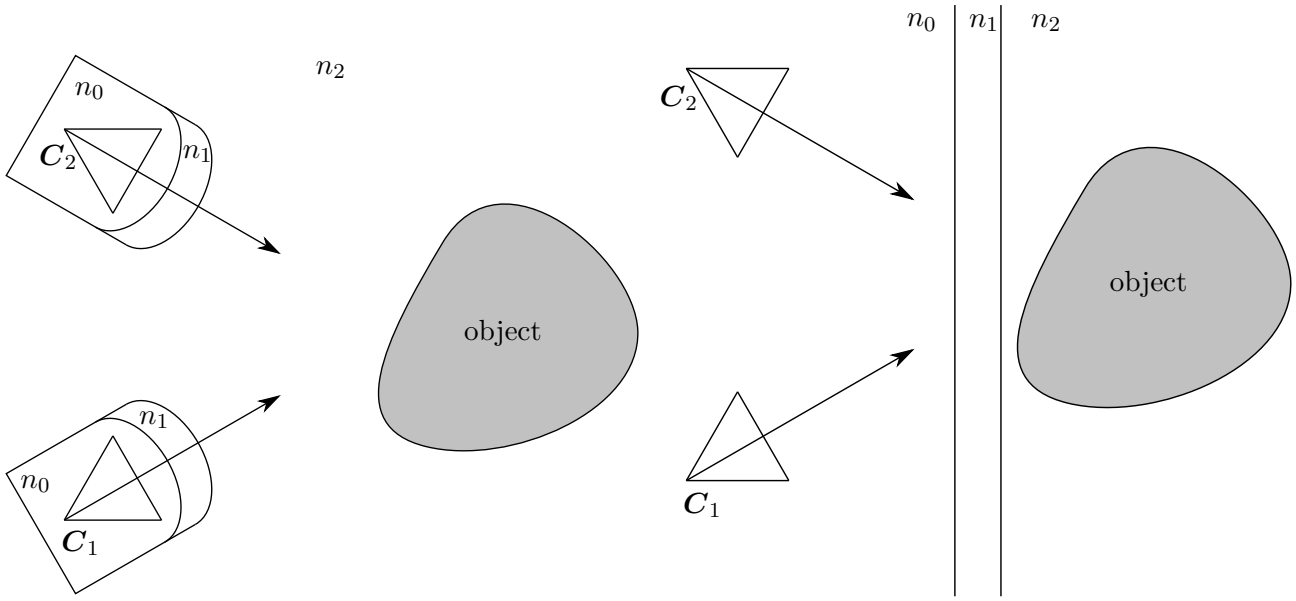
Here, the intrinsic part of the camera projection matrix takes 5 parameters, while the camera motion, which determines the extrinsic part of the projection is represented by $5 + n$ parameters, as discussed above. It follows that the bundle adjustment problem involving a single camera that moves on a circular path around the object of interest takes the following form:

$$\min_{\boldsymbol{K},\mathbf{d},\alpha,\beta,\theta_i \boldsymbol{X}_j} \sum_i \sum_j d(\tilde{\boldsymbol{x}}_{ij}, \boldsymbol{P}_i \boldsymbol{X}_j), \tag{3.23}$$

where the projection matrix $\boldsymbol{P}_i$ is constructed from the camera intrinsics $\boldsymbol{K}$ and the parameters describing the motion around the circular axis.

3.2.3.2 Bundle adjustment in refractive environments

As long as the assumption is satisfied that light rays propagate along a straight path between the camera and the object of interest, the concepts of projective geometry discussed so far



(a) Bundle invariant interface (b) Object invariant interface

Figure 3.5: *Bundle- and object-invariant interfaces in multimedia photogrammetry. For the bundle invariant case, the position of the refractive interface remains fixed with respect to the camera. In the object invariant case, the refractive interface remains fixed with respect to the object.*

hold. In some applications of image-based measurements, however, these assumptions cannot be used anymore. Generally, if light traverses through media with differing optical properties, the refraction phenomena occurring on the interfaces between different optical domains causes individual light rays to alter their directions. Photogrammetric measurements that involve the tracing of light rays passing through several optical media are referred to as *multimedia photogrammetry* [88]. Depending on the relative movement between the camera, the object and the refractive interface, two main categories of multimedia photogrammetry exist [89]:

1. **Bundle invariant interfaces:** refractive interfaces that do not change their position relative to the bundle of rays during the image acquisition process. This is usually the case in the context of underwater photogrammetry, where cameras are placed in a water-proof, pressurized housing.
2. **Object invariant interfaces:** refractive interfaces which do not change their position with respect to the world coordinate system during the image acquisition process. These interfaces typically occur if the object of interest is located in a closed, inaccessible area, allowing for observation from the outside.

In both cases, light rays incident on the image plane coming from the object undergo two refractions: one refraction on the water-glass interface and a subsequent one on the glass-air interface.

Typical applicaton areas of underwater photogrammetry involving bundle invariant interfaces are e.g. archaeology [90], marine biology [91] or measurements in seafloor geology [92]. Object invariant interfaces appear e.g., in the characterization of the movement of fluids

observed by cameras located outside of the observation vessel [93]. Accounting for object invariant interfaces is also required by some experimental methods for the study of the mechanical behavior of soft human tissues in an *in vitro* environment. In these experiments, it is important to be able to mimic the conditions that are present in the normal biological context of the investigated tissues. This usually requires the sample to be immersed in a physiological fluid. An example with such requirements is the study of the active biomechanical response of human arteries. Thus, the presence of the interface between the physiological fluid and the camera needs to be taken into account by the reconstruction algorithm.

Multi-media photogrammetry and the study of the influence of refractive interfaces have been the subject of extensive research in recent years. In the context of underwater photogrammetry, where the camera is immersed in a liquid medium, the main question is how to account for the distortion effects due to the camera housing [94–96]. Other applications aim at the reconstruction of the refractive interface itself, e.g. when measuring the shape of transparent or reflective surfaces [97, 98].

The behavior of an individual light ray passing through an optical interface that separates two media with different optical properties is governed by the law of refraction or Snell's law. If a light ray with the direction vector $\mathbf{r}_0$ intersects an optical interface with an outward facing unit normal vector $\mathbf{n}$, the direction of the refracted ray $\mathbf{r}_1$ is determined by [99]:

$$\mathbf{r}_1 = n_{01}\mathbf{r}_0 + \left[-n_{01}\mathbf{n}\cdot\mathbf{r}_0 - \sqrt{1 - n_{01}^2\left[1 - (-\mathbf{n}\cdot\mathbf{r}_0)^2\right]}\right]\mathbf{n}, \tag{3.24}$$

where $n_{01} = n_0/n_1$ is the ratio of the refractive indices of the incident and transmitted media.

Given a camera's intrinsic and extrinsic matrices $\boldsymbol{K}$ and $[\boldsymbol{R}|\boldsymbol{t}]$, an observation $\boldsymbol{x}$ on the image plane and the position and orientation of the refractive interface, Equation 3.24 can be used to determine the course of a single *back projected* ray. Further, if a scene is recorded by two or more cameras, and a pair of corresponding points $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ is given, the back-projected rays by Equation 3.24 can be intersected to find the structure point $\boldsymbol{X}$, which forms the multi-media equivalent of the triangulation procedure. This concept is also depicted in Figure 3.6.

In contrast to the ray tracing process that employs Equation 3.24 to trace the path of back-projected rays, the opposite problem, i.e. the projection of a point $\boldsymbol{X}$ from world coordinates onto the image plane of a camera cannot be computed directly. This is because no initial direction for the ray path can be given when starting from a point in world coordinates. However, the path of the ray can be computed in an iterative manner, following the *alternating forward ray tracing* (AFRT) algorithm, see in [100, 101]. The basic concepts of AFRT are outlined in the following, complemented by Figure 3.7.

Consider a scene with a refractive interface S and a feature point $\mathbf{X}_j$ to be projected onto a camera with the projection center $\mathbf{X}_C$. The AFRT procedure starts with an initial ray which is determined by connecting $\mathbf{X}_C$ and $\mathbf{X}_j$:

$$\mathbf{r}_0^k = \frac{\mathbf{X}_j - \mathbf{X}_C}{||\mathbf{X}_j - \mathbf{X}_C||}, \tag{3.25}$$

with $k = 1$ in the first iteration. Evaluating Equation 3.24 at the intersection point between the initial ray and the interface $\mathbf{X}_S^k = \mathbf{r}_0^k \cap S$ yields the direction of the refracted ray $\mathbf{r}_1^k$. Next, a ray with direction $-\mathbf{r}_1^k$ is traced from $\mathbf{X}_j$ towards the interface S. This gives a new

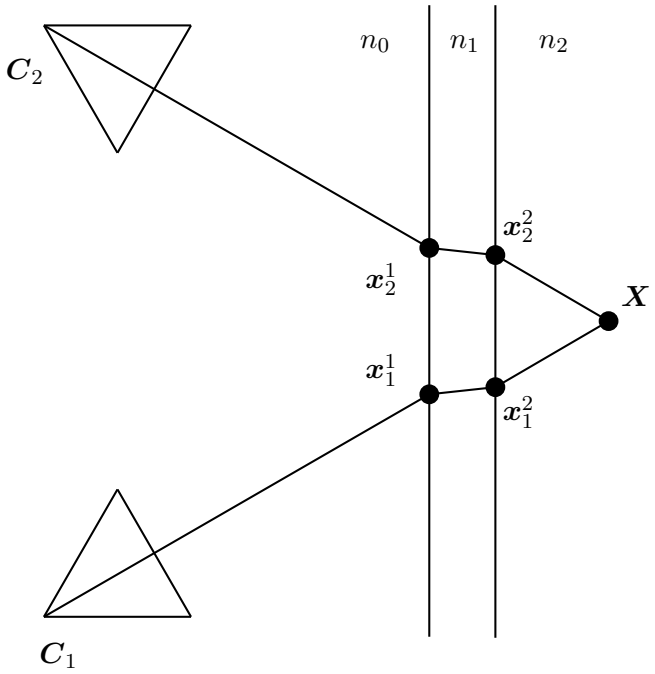


Figure 3.6: *Back-projection of individual rays in a multi-media environment. The law of refraction in Equation 3.24 is evaluated at every intersection point $\boldsymbol{x}_i^j$ between the light ray coming from the camera and the refractive interface, where i denotes the light ray coming from the camera $\boldsymbol{C}_i$ and j is the j-th intersection point with the refractive interface. The back projection of matching points is used to triangulate the structure point $\boldsymbol{X}$.*

intersection point $\mathbf{X}_S'^k$. If the distance $\Delta\mathbf{X}_k = ||\mathbf{X}_S^k - \mathbf{X}_S'^k||$ is larger than a tolerance value ϵ, the procedure is reiterated with:

$$\mathbf{r}_0^{k+1} = \frac{\mathbf{X}_S^{k+1} - \mathbf{X}_C}{||\mathbf{X}_S^{k+1} - \mathbf{X}_C||}, \tag{3.26}$$

where $\mathbf{X}_S^{k+1}$ is defined by the projection of $\frac{1}{2}\left(\mathbf{X}_S^k + \mathbf{X}_S'^k\right)$ onto S. The iteration terminates when $\Delta\mathbf{X}_k < \epsilon$. Finally, the projection of $\mathbf{X}_S^{k_{end}}$ onto the image plane by Equation 3.5 yields $\mathbf{x}_i^j$.

In order to incorporate the AFRT algorithm in the bundle adjustment formulation, the projective action of the AFRT on the feature point $\mathbf{X}_j$ for the i-th image will be denoted as:

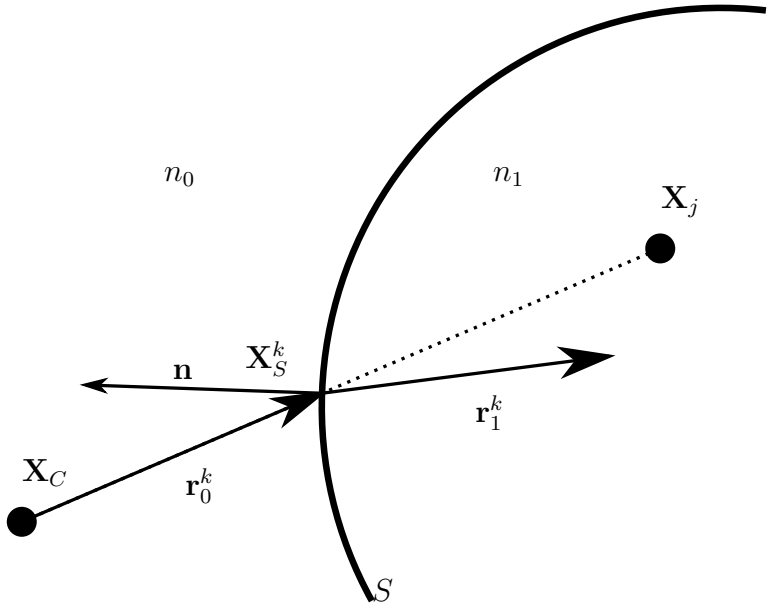
$$\mathbf{x}_i^j = P_{AFRT}\left(\mathbf{X}_j, \mathbf{K}, \eta, \mathbf{d}, \theta_i, \sigma\right), \tag{3.27}$$

where the vector σ represents the set of parameters that describes the geometry of the interface. If, for example, the shape of the interface is cylindrical, σ contains its orientation, location, radius, and the corresponding refractive coefficients.

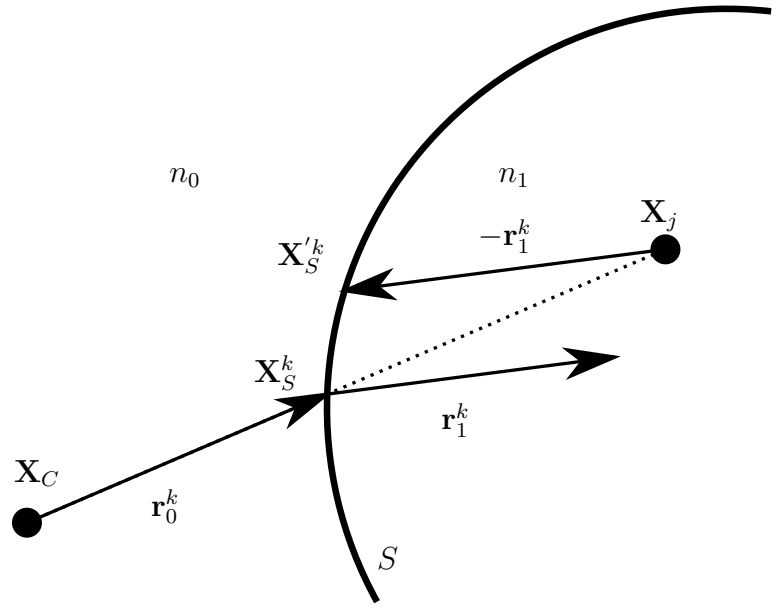
Under the assumption that the camera motion is cylindrical, the constrained refractive bundle adjustment problem takes on the following form:

$$\min_{\mathbf{X}_j,\eta,\mathbf{d},\mathbf{K},\theta_i,\sigma} \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - P_{AFRT}\left(\mathbf{X}_j, \mathbf{K}, \eta, \mathbf{d}, \theta_i, \sigma\right)\right)^2. \tag{3.28}$$

Note that here the projective relationship of Equation 3.23 is replaced by the AFRT procedure.



(a) The refracted direction $\mathbf{r}_1^k$ is computed at $\mathbf{X}_S^k$.



(b) The ray $-\mathbf{r}_1^k$ is traced towards S, resulting in $\mathbf{X}_S^{'k}$.

Figure 3.7: *Schematic of the AFRT algorithm. (Figure from [18], following the description in [100]).*

3.2.4 Multi-view stereo reconstruction

The so-far discussed concepts allow for recovering the motion, i.e. the camera positions at the time the images were acquired, and the structure, the 3D points in world coordinates that are back-triangulated from matching observations. Matching observations usually stem from regions that are characterized by high gradients of image intensity: corners, edges, highly textured areas. Therefore, the structure-from-motion step usually yields a cloud of points that is not necessarily dense enough in every region of the recorded object. These *sparse point clouds* may not be sufficient for many downstream applications, such as photorealistic rendering, virtual reality, or reverse engineering. In order to recover even more points, multiple-view stereo algorithms may be used. These approaches start from the 3D points recovered during the structure-from-motion step and perform an expansion step which results in a point cloud with orders of magnitude more points, usually referred to as a *dense cloud*.

The most fundamental method to recover a dense cloud from a set of oriented images is based on sliding a window along the one dimensional epipolar line looking for a pair of regions between images with a high correlation. Once a matching pair of windows has been found between two views, the 3D position of the originating portion of the body of interest can be back-triangulated using the methods discussed before [73, 74].

A more efficient variant of dense multi-view reconstruction is based on region-growing, starting from a set of seed points distributed on the images. Usually, corners or highly textured regions are chosen as seeds. One example following these ideas is the algorithm explained in [102]. Here, the initial matches over the recorded images are stored in a *seed set*. The measure that defines the "quality" of a match is computed using the zero-mean normalized cross correlation (ZNCC) between matching regions of an image pair. The initial matches are sorted in the seed set according to their ZNCC score. At every iteration of the algorithm, the top-most match is removed from the seed set, and potential matching candidates are sought for in the local neighborhoods of the match. A local neighborhood is usually a square window of 5×5 - 10×10 pixels in size. Matching candidates with a ZNCC score over a certain threshold get stored in the seed set and become seeds themselves. Finally, the seed set is sorted again according to the ZNCC score of the elements stored within.

An improved version of the region growing algorithm for dense multi-view reconstruction is given in [103]. Instead of treating the model to be reconstructed as a set of single points, it assumes that the model is composed of a set of quadrilateral *patches*. In contrast to a point that only has a spatial position, a patch is characterized by its *position and its normal vector*, that is, it possesses an orientation in the 3D world. Because of the normal vector, patches carry more geometric information than single points. To exploit this additional information, the NCC score is not computed between rectangular windows on image pairs, but over the projections of rectangular patches that are projected onto the image planes. The algorithm implements 3D dense reconstruction in three steps: *match, expand, filter*. In the *matching* step, points of interest on the input images are found. Then, the points of interest are used to establish a network of matches across multiple images. The identified matches are used to generate a set of initial patches. The *expansion* step works similarly as discussed before for the simpler algorithm. In this process, the initial patches are expanded by expansion candidates, as long as a candidate is consistent with the original patch. A candidate patch is accepted if it lies in the neighborhood of the original patch and their normal vectors are sufficiently close. Finally, the *filtering* step removes erroneous matches and makes sure that only those patches

are kept that satisfy some visibility criteria.

One advantage of using this algorithm is that it provides not only simple positions but also normal vectors associated to the points in the resulting point clouds. The obvious advantage of this extra geometric information is that oriented point clouds can produce better renderings than unoriented ones, and many surface reconstruction algorithms are based on exploiting the extra geometric information that normal vectors carry. Moreover, as will be shown in Section 4, point clouds equipped with normal information can also be used directly for numerical analysis, when combined with the Finite Cell Method.

3.3 Geometric model recovery from point clouds

The shape reconstruction methods discussed so far—especially multi-view reconstructions and laser scanning—reproduce the geometry of interest in a discrete manner: a set of points, a *point cloud* is generated which constitutes a discrete sampling of the continuous boundaries of the object. While point clouds are well-suited for quick dimensional inspections or visualization purposes [104], the requirements posed by many downstream applications lie beyond the capabilities of these discrete representations. Many applications require surface models for texturing and photorealistic renderings. Examples in this category include virtual reality models built upon three-dimensional recordings of cultural heritage sites [105] or geometric models of statues and other cultural heritage artifacts [106]. Recovering surfaces from point clouds is also a necessary step if one aims to perform numerical analysis: typically, a finite element mesh generator expects a CAD model in BREP or STL format. Thus, in order to make point clouds compatible with applications in numerical analysis, the key challenge is to transform the underlying surface information carried by the points into a geometric model.

This question has long been in the center of attention in the field of *reverse engineering* and *computer graphics*, resulting in solutions that tackle the problem using various considerations. Generally, the approaches pursuing the goal of recovering a surface model from a point cloud can be divided into two main categories: *recovery by geometric primitive identification* and *implicit function fitting*.

3.3.1 Recovery by geometric primitive identification

To the first category belong methods that rely on the observation that the majority of objects encountered in the everyday engineering practice can be approximately represented by a combination of simple geometric shapes, such as planes, spheres, cylinders, tori, etc. Therefore, these approaches start with a *geometric primitive identification* technique which segments the point cloud into subsets that can be classified as these base shapes. Following this *segmentation* step, a topology recovery method is applied to establish the associative relationships between the identified primitives. Finally, having identified the primitive shapes and their topological associations, a surface model is created and stored in a standard CAD format. An example for this approach can be found in [107], where the CAD representation of buildings is derived from point clouds recorded by terrestrial laser scanning. To this end, artificial neural networks are employed to identify the individual “building elements”, such as walls, columns or doors.

The basic terminology and challenges associated to primitive segmentation methods can be found in [62]. In this work, the authors divide the segmentation problem into two main classes: edge-based and face-based methods. For edge-based methods, the reconstruction approach first identifies boundaries between neighboring primitive surfaces by looking for sharp edges in the input point cloud. Then, individual surfaces can be found by the implicit segmentation provided by the sharp edge data. On the other hand, face-based approaches work in an opposite manner: they infer regions in the cloud with similar properties, thereby identifying connected regions representing the same primitive surfaces. Once these surfaces are identified, computing their intersection allows to recover their bounding sharp edges. In [62], the authors propose a region-growing approach to implement a face-based segmentation technique.

Alternatives to region growing methods are techniques based on direct segmentation, where clusters of regions are identified by testing against hypotheses about the characteristics of the underlying surfaces of separate regions. See, e.g. [108] for a method that works directly on point clouds, and [109] for an example that works on a triangulation. The challenging aspect of these methods is that they may not necessarily work if measurement noise is present. In such cases, the outlier points may draw the segmentation algorithms towards false positives. Outliers may be identified as actual parts of the recovered surfaces, or, even worse, they may alter the identified type of the primitive which is recovered during the segmentation process.

To overcome these problems, [110] proposed to employ the *random sample consensus* (RANSAC) algorithm. RANSAC, introduced in [111] is an iterative, voting-based approach that aims at fitting mathematical models onto scattered data in the presence of outliers. The algorithm extracts shape primitives from the point cloud by randomly drawing minimal point sets from the point data that are sufficient to define minimal shape primitives. A minimal point set is the amount of points that is required to uniquely define a given geometric primitive: e.g. 3 points define the location and orientation of a plane. In every iteration of the algorithm, the candidate shapes defined by the minimal sets are tested against the remaining points in the cloud to see how many of them are well approximated by the primitive. The measure that is used to characterize how well the remainder of points is approximated by the primitive is called the *score* of the shape. After a pre-defined amount of trials, the shape which possesses the highest score is extracted from the cloud and the algorithm continues on the remaining data. In the work of [110], five basic geometric shapes are considered: plane, sphere, cylinder, torus and cone. In this work, to speed up the convergence of the RANSAC algorithm while maintaining a maximal detection probability, the authors propose an octree-based sampling strategy, which starts from the observation that those samples that belong to the same geometric primitive are more likely to be located close to each other. While this method seems to work for clouds that contain only simple primitives, a similar, voting-based approach is presented in [112], able to recognize 3D free-form objects in point clouds.

The identified primitives may be used in further processing steps to achieve completion of missing parts, as explained in e.g. [113] or [114].

While the set of shape types that can be recovered this way is restricted to basic geometric primitives, these methods can be easily extended to more complex objects. To this end, the primitives are converted from their analytical representation into B-Spline or NURBS surface descriptions, which is then used as an initial guess for a subsequent surface fitting procedure, e.g. surface fitting by least squares. In this process, care needs to be taken that the geometric model is kept in a valid state. This means that the fitting algorithms need to ensure that the data structure containing the topological relationships is always updated if a change in the

topology occurs during the fitting process.

While primitive identification approaches rely on the observation that many engineering geometries are composed of basic geometric primitives, such as planes, spheres, tori, etc. the type of the primitives is not the only *shape prior* that may be exploited during the reconstruction process. For example, very often, CAD models are not only composed of such primitives, but the relationship between these geometric components also tends to follow certain patterns, such as coplanarity, orthogonality, concentricity, etc. These relationship priors can also be exploited at the time of the reconstruction.

In facade models for example, the Manhattan world (MW) assumption can be used: all planar primitives in a cloud belong to one of three mutually orthogonal planes, see e.g the method in [115], or [116] for a comprehensive survey of other methods in urban reconstruction. Other than simple MW assumptions, further relationships between the primitives such as parallelism, orthogonality and equal angles may be detected and enforced in order to help recovering the underlying geometry from the point cloud scene. This is the approach taken by the method explained in [117], which, starting from a RANSAC identification step, enforces the above constraints to attenuate the effects of structured measurement noise, such as scan misalignment.

The segmentation and subsequent reconstruction of urban environments can be further improved when prior knowledge is available about the objects and object-relationships that are expected to appear in a given scene. For example, [118] presents a methodology for monitoring the progress of construction sites by combining spatio-temporal Building Information Modeling (BIM) data and point clouds acquired by photogrammetric techniques. Another promising direction of research concerning the identification and classification of urban objects is provided by machine learning techniques: see e.g. in [119] for an approach that works on aerial images or in [120] for a classification algorithm working with point clouds.

The approaches listed in this section represent only a small part of all the methods that are available for recovering a geometric model starting from primitive identification techniques. For a more extensive survey, see the overview provided in [121].

3.3.2 From primitives to best-fit surfaces

There are structures that are difficult to be described by a combination of basic geometric primitives. Such non-standard models often arise when the shape of interest is "natural": rock formations, landscapes, sculptures, objects stemming from medical imaging techniques are models whose geometries are usually much more complex than (a combination of) spheres, cones, tori, etc... Such general shapes can be best modeled by free-form surfaces and curves, such as B-Splines and NURBS, see Section 2.1.5. Fitting B-Spline curves and surfaces onto point clouds is a well-studied problem in the literature, and a complete discussion of this topic lies beyond the scope of this thesis. Here, only the key ideas are listed - the interested reader may refer to the overview found in [122].

The problem of fitting a parametric planar B-Spline curve onto a point cloud is formulated as follows. Let $\boldsymbol{p}_k \in \mathcal{R}^2$, $k = 1...m$ be a set of unorganized points onto which a parametric spline curve $\boldsymbol{C}(t)$ shall be fitted. Following Section 2.1.5, the B-Spline curve is represented as

a linear combination of control points $\boldsymbol{P}_i$ and the associated basis functions $N_i(t)$ [24]:

$$\boldsymbol{C}(t) = \sum_{i=0}^{n} N_i(t)\boldsymbol{P}_i. \tag{3.29}$$

Without loss of generality, it is assumed that the B-Spline curve is parametrized such that $t \in [0, 1]$. The goal of the curve fitting procedure is to find the set of control points that minimizes:

$$f = \frac{1}{2}\sum_{k=0}^{m} ||\boldsymbol{C}(t_k) - \boldsymbol{p}_k||^2, \tag{3.30}$$

where each $t_k \in [0, 1]$ is a parametric location associated to one point $\boldsymbol{p}_k$ in the input point cloud. Substituting Equation 3.29 into the expression above yields:

$$f = \frac{1}{2}\sum_{k=0}^{m} \left|\left| \sum_{i=0}^{m} N_j(t_k)\boldsymbol{P}_i - \boldsymbol{p}_k \right|\right|^2. \tag{3.31}$$

The set of control points that minimizes the cost function above is found by setting the derivative with respect to the control points' position to zero.

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{P}_j} &= \sum_{k=0}^{m} \left(\sum_{i=0}^{n} N_i(t_k)\boldsymbol{P}_i - \boldsymbol{p}_k \right) N_j(t_k) = 0, \qquad i = 0...m, \\ &= -\sum_{k=0}^{m} \boldsymbol{p}_k N_j(t_k) + \sum_{k=0}^{m}\sum_{i=0}^{n} N_i(t_k)N_j(t_k)\boldsymbol{P}_i, \end{aligned} \tag{3.32}$$

which yields $2 \times n$ or $3 \times n$ equations for the n control points in 2D and 3D, respectively. Equation 3.32 can be written more compactly as:

$$\boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{P} = \boldsymbol{A}\boldsymbol{p}, \tag{3.33}$$

where the coefficient matrix $\boldsymbol{A}$ collects the shape functions evaluated at the t_k locations:

$$\boldsymbol{A} = [a_{ik}] = [N_i(t_k)]\,. \tag{3.34}$$

The solution of the equation above yields the set of control points that minimize the cost function of 3.31. In practice, the fitting procedure requires that an initial approximation of the curve $\boldsymbol{C}(t)$ is available, in order to be able to compute the error term $||\boldsymbol{C}(t_k) - \boldsymbol{p}_k||$. Even if this initial guess is available, the parametric locations t_k still need to be chosen. A commonly employed approach is to compute the closest projection of the point $\boldsymbol{p}_k$ onto the curve $\boldsymbol{C}(t)$ yielding a parametric value t_k for every point $\boldsymbol{p}_k$.

As the error term in Equation 3.31 is based directly on the distance between the curve and the points to be fitted, this procedure is commonly referred to as *Point Distance Minimization* (PDM). Thanks to its simplicity, it is a frequently employed approach for fitting B-Spline curves as well as surfaces onto a set of unstructured points, see, e.g. [123, 124]. However, the PDM method suffers from some drawbacks, e.g. slow convergence, making it not the most optimal solution when it comes to fitting curves or surfaces, see [122].

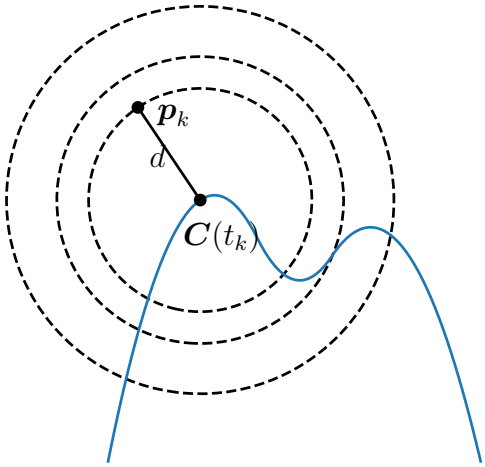


Figure 3.8: *Iso-curves of the point distance (PDM) error term.*

To avoid the difficulties associated to the PDM-based minimization, a frequently employed approach is to use a different error term, which takes into account more information about the local geometry of the curve to be fitted. This technique is called *tangent distance minimization* (TDM), and its error term is formulated as follows [125]:

$$f = \frac{1}{2}\sum_{k=0}^{m} |(\boldsymbol{C}(t_k) - \boldsymbol{p}_k)\boldsymbol{N}_k|, \tag{3.35}$$

where $\boldsymbol{N}_k$ is the unit normal vector of the curve $\boldsymbol{C}(t)$ in the current iteration, evaluated at the parameter t_k. While this formulation involves more geometric information about the curve geometry than a simple point-to-point measure, it still remains a poor approximation of the surface, creating instabilities in the curve fitting process in regions with high curvature.

To resolve this problem, the following observation about the iso-contours of the PDM and TDM approaches come to help: for the PDM approach, the error term is constant for any $\boldsymbol{p}_k$ satisfying $e_{\mathrm{PD},k} = ||N_j(t_k)\boldsymbol{P}_i - \boldsymbol{p}_k||^2 = c$. The contour of such curve corresponds to a circle, as depicted in Figure 3.8. In contrast, the iso-contour for the TDM-based fitting is defined by $|(N_j(t_k)\boldsymbol{P}_i - \boldsymbol{p}_k)\boldsymbol{N}_k| = c$, representing a pair of parallel lines, which can be regarded as the contour of a degenerate ellipse (Figure 3.9). The intuitive observation is that if the PDM approach with the circular iso-contours leads to slow convergence, while the TDM with the parallel lines produces fast, but unstable convergence, an optimal error term should be one whose iso-contours lie between the two, represented by ellipses. An error term that creates elliptical iso-curves of errors is provided naturally by a curvature-based approximation of the squared distance function, as described in [126].

Given a curve $\boldsymbol{C}(t)$ in $\mathcal{R}^2$, the squared distance function assigns to each point $\boldsymbol{p} \in \mathcal{R}^2$ the squared distance from $\boldsymbol{p}$ to $\boldsymbol{C}(t)$. Let $\mathcal{X}$ be the closest point on the curve $\boldsymbol{C}(t)$ to the point $\boldsymbol{p}$. Let ρ denote the radius of curvature of the curve at the point $\mathcal{X}$. In the local Frenet frame of the curve at $\mathcal{X}$, the center of curvature is located at $(0, \rho)$, while the coordinates of the point $\boldsymbol{p}$ are given by $(0, d)$, where d denotes the distance between $\boldsymbol{p}$ and $\mathcal{X}$. In the Frenet frame, the second order approximant of the squared distance function is given by:

$$F_d(x_1, x_2) = \frac{d}{d-\rho}x_1^2 + x_2^2, \tag{3.36}$$

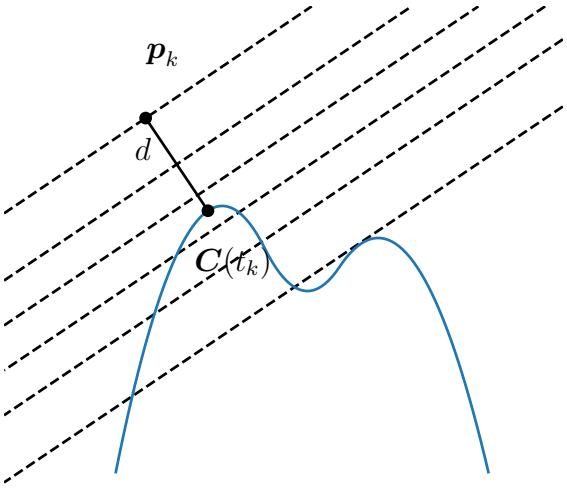


Figure 3.9: *Iso-curves of the tangent distance (TDM) error term.*

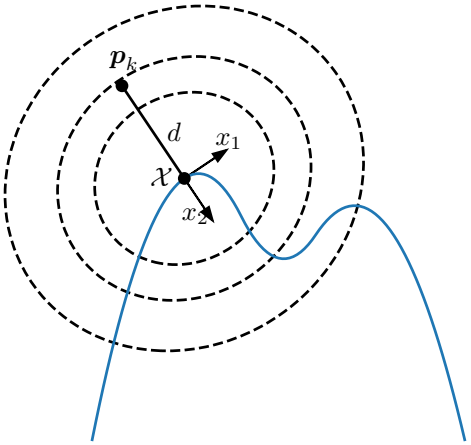


Figure 3.10: *Iso-curves of the approximated squared distance function at* $\boldsymbol{p}_k$.

where x_1 and x_2 are the coordinates measured along the coordinate axes of the local Frenet frame. An example for the approximant of the squared distance function is plotted in Figure 3.10. The concept of quadratic approximants may be used for curve fitting as described in [126]. In the following, a summary of this procedure is given.

Given a model shape $\mathcal{M}$ which needs to be approximated by a parametric B-Spline curve $\boldsymbol{C}(t)$, the process first generates a set of sample points on the curve at the current iteration, called *active curve points*, denoted by $\{\boldsymbol{s}_k\}, k = 1...M$. Then, for every $\boldsymbol{s}_k$, the closest point on the target shape $\mathcal{M}$ is evaluated. For every active point together with its closest point on $\mathcal{M}$, the approximate squared distance function can be computed, denoted by F_d^k. The surface fitting problem amounts to finding a displacement vector $\boldsymbol{c}_i$ for every control point $\boldsymbol{P}_i$, such that the sum of F_d^k's is minimized:

$$F = \sum_{k=0}^{M} F_d^k \left(\sum_{i=0}^{n} N_i(t_k) \left(\mathbf{P}_i + \mathbf{c}_i \right) \right). \tag{3.37}$$

It was shown in [126] that minimizing F amounts to solving a system of linear equations:

$$2\mathbf{A}\mathbf{c} + \mathbf{b} = 0, \tag{3.38}$$

where the vector $\mathbf{c}$ (of size $2n \times 1$) collects the displacement vectors of the control points: $\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n]^T$. The next iteration starts with an updated surface whose control points are shifted by the values in $\mathbf{c}$, and the procedure is repeated until the root mean squared approximation error is less than a tolerance value. For point clouds, the curvature of the underlying surface can be estimated by applying the method of osculating jets [127] on the k-neighborhood of the closest point to $\boldsymbol{x}_k$. One step of this process is depicted in Figure 3.11.

For surfaces, the fitting procedure can be executed similarly. In order to compute the quadratic approximant of the squared distance function in this case, [126] provides the following formula:

$$F_d^k(x_1, x_2, x_3) = \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2, \tag{3.39}$$

where ρ_1 and ρ_2 denote the radius of curvatures associated to the principal curvatures of the surface evaluated at the closest point.

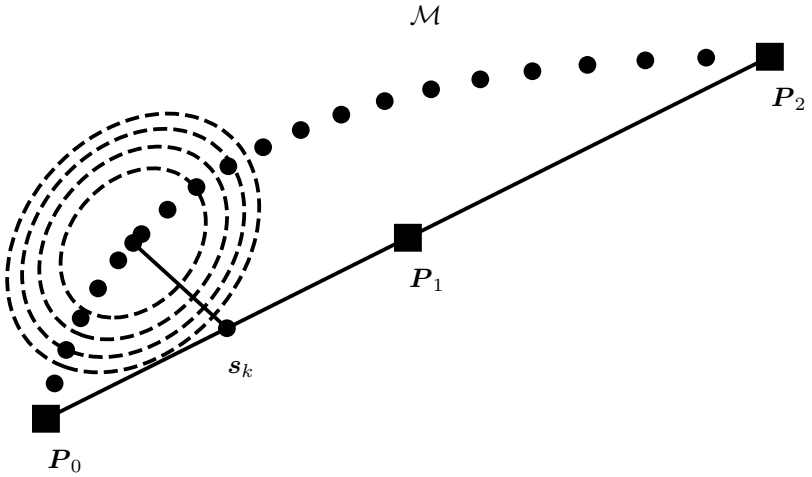
3.3.3 Methods based on implicit function fitting

The second major class of geometry recovery techniques formulates the reconstruction problem in terms of level-set surfaces. Geometric models defined by level-set surfaces are particularly popular in the field of computer graphics [128]. Unlike recovery methods that are based on primitive identification, these procedures do not require the handling of topological changes occurring during the fitting process. Instead, their formulation allows for representing models of almost arbitrary geometric and topological complexity, without having to handle the associative relationships between the geometric elements that compose the model. Usually, surface recovery following implicit function fitting follows a two-step procedure. In the first step, a scalar function $\Phi(\boldsymbol{x}) : \mathbb{R} \to \mathbb{R}^3$ is sought, whose level set $\partial\mathcal{M} = \{x \in \mathbb{R}^3 | \Phi(\boldsymbol{x}) = \Phi_c\}$ approximates the surface which is represented by the input point cloud. Then, in the second step, a contouring method is employed to extract a tessellation – usually consisting of triangles – from the level-set surface of the first stage.

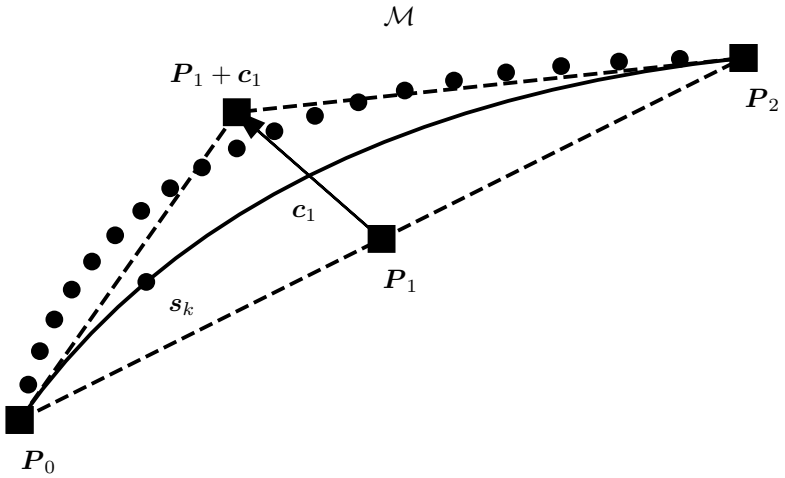
Different approaches have emerged to formulate the problem of finding $\Phi(\boldsymbol{x})$. The first important method in this class is the pioneering approach of [129]. This method seeks to find $\Phi(\boldsymbol{x})$ by computing the signed projection of every point $\boldsymbol{x} \in \mathbb{R}^3$ onto the tangent plane defined by the closest point and the associated normal vector in the point cloud. While this method is easy to implement, it produces noisy and unreliable output if the normals in the cloud are not consistently oriented or when the sampling density is not uniform.

Subsequent approaches aim at overcoming this problem by not only considering the closest point but also its local neighborhood. The most notable methods relying on surface information extracted from local neighborhoods are based on the method of Moving Least Squares (MLS). MLS approaches reconstruct $\Phi(\boldsymbol{x})$ by fitting a multivariate polynomial onto the closest neighborhood of $\boldsymbol{x}$ in a weighted least-squares sense. The techniques described in [130, 131] use low-degree bivariate polynomials defined on the local reference frame computed by principal component analysis. Subsequently, the projection of $\boldsymbol{x}$ onto the MLS surface can be defined as the closest point on the local polynomial approximation. Finally, the MLS surface is implicitly defined as the stationary set of the projection operator.

The use of polynomials is not absolutely necessary to represent the local approximant of the surface in the MLS framework. As shown in [132, 133], it is sufficient if a planar approximation



(a) For an active point $\mathbf{s}_k$, the approximant of the squared distance function is computed by finding $\mathbf{p}_k$ and its k-neighborhood.



(b) After solving Equation 3.38, the control point $\mathbf{P}_1$ is shifted by the vector $\mathbf{c}_1$.

Figure 3.11: *Conceptual sketch of the active surface fitting algorithm in 2D. The circular black dots represent the point cloud $\mathbf{X}_j^{\mathcal{M}}$, while the solid black curve depicts the active curve which is iteratively deformed towards $\mathbf{X}_j^{\mathcal{M}}$. This concept stems from [126].*

is employed instead of polynomials. In terms of MLS fitting, this means that the tangent plane is computed by a weighted average of neighboring tangent planes found in some neighborhood of the query point $\boldsymbol{x}$.

While MLS-like approaches take a local view of the problem by looking only on individual points and their neighborhoods, there are formulations that operate on a global level. Historically, the first notable method following this idea is the method of [134], based on radial basis functions (RBFs). RBF methods, popular in the context of scattered data interpolation, aim at reconstructing the level-set surface by taking a linear combination of radially symmetric basis functions, $\phi(\boldsymbol{x})$'s. Then, the implicit function Φ is expressed as:

$$\Phi(\boldsymbol{x}) = g(\boldsymbol{x}) + \sum_j \lambda_j \phi(||\boldsymbol{x} - \boldsymbol{q}_j||), \tag{3.40}$$

where $g(\boldsymbol{x})$ is a globally supported, low-degree polynomial and the basis functions $\phi(\boldsymbol{x})$ are centered at the nodes $\boldsymbol{q}_j$. In order to find the unknown λ_j coefficients, the function is prescribed to take on the value 0 at the locations of the points in the point cloud S. Furthermore, to avoid the trivial solution where $\Phi(\boldsymbol{x})$ is zero everywhere, additional off-surface constraints are imposed. Usually, these constraints are created by shifting the original points in the cloud in the direction of the normal vectors by a small distance ε. Solving the resulting dense system of linear equations yields the value of the unknown coefficients λ_j. Once the values of the coefficients are found, the surface of interest is recovered as the zero level set of the global function $\Phi(\boldsymbol{x})$.

The need for applying off-surface constraints can be overcome by additionally requiring that the gradient of $\Phi(\boldsymbol{x})$ interpolates the normal vectors $\boldsymbol{n}_i$ given at the data points $\boldsymbol{p}_i$, as described in [135]. An application of this approach for 3D reconstructions can be found in [136].

The other popular category of global approaches in the context of reconstruction by implicit functions is based on recovering the indicator function itself. These *volumetric labeling* methods rely on the key observation that the indicator function can be found by ensuring that its gradient matches the normal vectors that are given at the locations of the points in the point cloud S. Thus, these methods find an indicator function that minimizes the following functional:

$$\int_\Omega ||\nabla\Phi(\boldsymbol{x}) - \boldsymbol{n}||^2 \mathrm{d}\Omega. \tag{3.41}$$

Applying the Euler-Lagrange equation, this minimization problem can be cast in the form of a standard Poisson equation, i.e. the minimizer of Equation 3.41 satisfies:

$$\Delta\Phi(\boldsymbol{x}) = \nabla \cdot \boldsymbol{n}. \tag{3.42}$$

This Poisson problem may be solved in the frequency domain, by taking the Fourier transform of both sides of the equation, as discussed in [137]. The Fourier transform approach is restricted to regular grids, thereby limiting the spatial resolution of the reconstructed surface. This limitation can be overcome by solving the Poisson problem using a hierarchical formulation known from the Finite Element Method and a multigrid solver, as explained in [10]. This approach is widely popular in state-of-the art 3D surface reconstructions, and will be discussed in more detail in Section 4.4.3.

To avoid oversmoothing, the Poisson formulation can be extended by incorporating additional positional constraints in the energy functional of Equation 3.41, resulting in the *screened Poisson* problem [11]:

$$\int_{\Omega} ||\nabla\Phi(\boldsymbol{x}) - \boldsymbol{n}||^2 \mathrm{d}\Omega + \lambda \sum_{\boldsymbol{p}_i \in S} \Phi^2(\boldsymbol{p}_i), \tag{3.43}$$

where a higher value of the parameter λ draws the zero level-set of Φ closer to the input points in the cloud S. This can reduce the amount of oversmoothing, but, on the other hand, choosing a value of λ that is too high may result in overfitting, just like in the case of interpolatory methods.

One complication that occurs when employing Poisson-based reconstruction methods is the observation that the gradient of the indicator function $\nabla\Phi(\boldsymbol{x})$ cannot be defined in the traditional sense at the sample points, as $\Phi(\boldsymbol{x})$ exhibits a discontinuity at these locations. Therefore, these methods essentially aim to recover a low-pass-filtered version of the indicator function. As also noted in [10], this smoothing filter needs to be chosen carefully. Moreover, in case of non-uniformly sampled points, the width of the smoothing kernel needs to be adapted, which introduces and additional complexity in the implementation of these methods.

A related technique that aims at circumventing this problem formulates the reconstruction problem in terms of recovering a *smooth signed distance* function $\Phi(\boldsymbol{x})$ which is negative inside and positive outside the object. However, contrary to interpolatory approaches and similar to the Poisson-based methods, this goal is achieved by minimizing the following energy:

$$\lambda_0 \sum_i \Phi(\boldsymbol{p}_i)^2 + \lambda_1 \sum_i ||\nabla\Phi(\boldsymbol{p}_i) - \boldsymbol{n}_i)||^2 + \lambda_2 \int_{\Omega} ||H\Phi(\boldsymbol{x})||^2 \mathrm{d}\Omega, \tag{3.44}$$

where $H\Phi(\boldsymbol{x})$ denotes the Hessian, a 3×3 matrix containing the second-order partial derivatives of the signed distance function $\Phi(\boldsymbol{x})$, and the norm of this matrix is its Frobenius norm. Then, Equation 3.44 is discretized by finite differences, to find the unknown function $\Phi(\boldsymbol{x})$. As a result of this formulation the gradient of $\Phi(\boldsymbol{x})$ tends to be constant away from the data points, where the Hessian term dominates. In the vicinity of the data points, however, where the contribution of the first two parts of the energy is stronger, the function approximates the signed distance function.

In the final step of implicit reconstruction approaches, the level-set surface is converted into a tessellation by a contouring algorithm. Usually, this contouring step is performed by the marching cubes method introduced in [55]. To enhance the efficiency of marching cubes, its octree-based modifications may be employed, such as the methods found in [138–140]. The application of marching cubes might result in loss of geometry information: thin or sharp features may get lost in the contouring process. To avoid these complications, the extensions of the original marching cubes algorithm such as *extended marching cubes* [141] or *dual marching cubes* [142] variants may be employed. A survey on the marching cubes algorithm and its extensions can be found in [143].

3.4 Application: mesh generation on a tubular geometry[c]

In the following, it is presented how the techniques described previously in this chapter can be employed to generate a mesh of high order hexahedral finite elements on a tubular geometry, starting from images that were taken on a circular path around the object of interest. The chain of steps which are followed by the method within this section follow the stages depicted in Figure 3.12.
One application area of the method is to be seen in experimental studies of the mechanical behavior of soft human tissues in an *in vitro* environment. In these experiments, it is important to be able to mimic the conditions that are present in the normal biological context of the investigated tissues. This usually requires the sample to be immersed in a physiological fluid. An example with such requirements is the study of the active biomechanical response of human arteries. These measurements are challenging for traditional image-based reconstruction methods, because the assumption that light rays propagate along a straight path becomes invalid. Thus, the presence of the interface between the physiological fluid and the camera needs to be taken into account by the reconstruction algorithm.
The goal of generating a high order finite element mesh is motivated by the superior approximation properties of the high order finite element method (p-FEM) compared to standard finite elements. Instead of linear shape functions, p-FEM employs high order polynomials to approximate the solution of physical problems described by partial differential equations [22]. Especially for smooth problems, p-FEM delivers very accurate results with drastically less degrees of freedom compared to linear FEM. The advantages of p-FEM can be fully exploited if it is combined with a high-order discretization of surfaces and volumes.

Using a high-order representation of the geometry is not only useful when the computation is done by the p-FEM, but also in the case of other numerical discretization techniques, like the method of Isogeometric Analysis. For IGA, the last, *mesh generation* step in Figure 3.12 may be omitted, as the method uses the shape functions of the geometric model directly for the discretization of the solution space.

3.4.1 Effects of refraction

Before demonstrating the entire mesh generation procedure, the importance of including the refractive effects in the bundle adjustment formulation is demonstrated. To this end, we consider a scenario where a planar object of size $25mm \times 19mm$ with a checkerboard pattern is immersed in a cylindrical container filled with water. The image of the checkerboard is recorded on a single camera using a ray-tracer software. The imaging setup is depicted in Figure 3.13. The positions of the individual checkerboard corners on the synthetically generated images are detected using the checkerboard detector algorithm of OpenCV [144]. Let $\mathbf{X}_j^G$ denote the coordinates of checkerboard corners in world coordinates, $\tilde{\mathbf{x}}_j$ the image coordinates of the detected corners, $\mathbf{x}_j^P$ and $\mathbf{x}_j^{AFRT}$ the projection of the checkerboard corners onto the image plane of the camera using standard perspective projection and the AFRT algorithm, respectively. The norm of the reprojection error for an individual corner with perspective

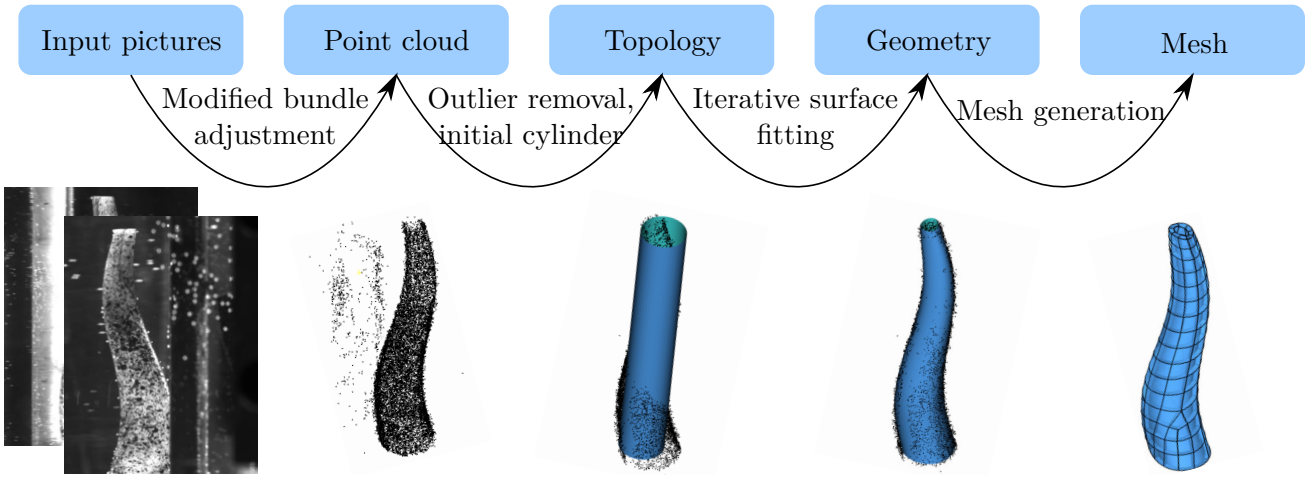


Figure 3.12: *From images to a boundary-conforming finite element mesh, for tubular geometries [18].*

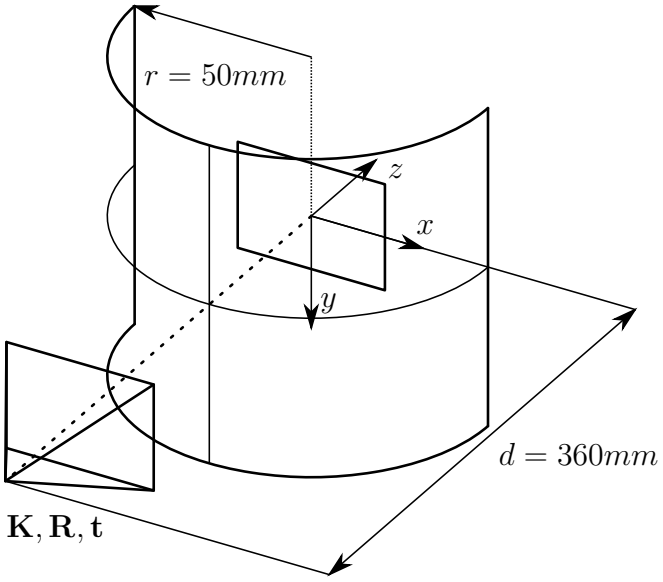


Figure 3.13: *Setup of the test example with a single camera and a checkerboard pattern immersed in a cylindrical water container [18].*

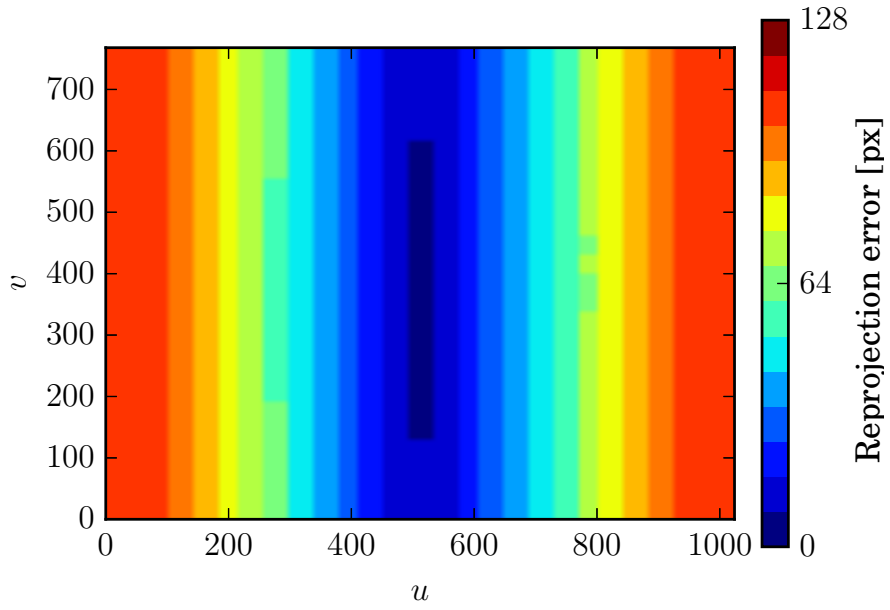


Figure 3.14: *Reprojection errors due to refractive effects for the checkerboard example. The error is small around the camera center and increases towards the image boundaries. Both coordinate axes are in pixel units [18].*

projection can be defined as:

$$e_j = \left|\left|\tilde{\mathbf{x}}_j - \mathbf{x}_j^P\right|\right|_2 . \tag{3.45}$$

The reprojection error of the AFRT algorithm can be computed similarly. Figure 3.14 shows the error of the standard perspective projection in the image coordinate space. In this imaging configuration, the light rays associated to the pixels around the image center enter the refractive medium almost orthogonally, which means that they are not subject to significant refraction and their direction remains almost unchanged. Therefore, the error associated to the pixels around the center of the image is low, as also shown in the figure. For image points that are further away from the imaging center, however, the refractive effects dominate the error. Compared to the values obtained by the AFRT algorithm, which remain in the sub-pixel accuracy range, this error can be orders of magnitudes higher.
This example emphasizes that if the refractive effects are not accounted for, the value of the objective function in the bundle adjustment problem (Equation 3.21) remains high – even if the exact values of the structure points $\mathbf{X}_j^G$ and the camera parameters $\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i$ are substituted into the equation. Moreover, the minimum of the objective function does not occur at these set of parameters at all. Instead, the non-linear least squares solver that is used to solve the bundle adjustment problem is drawn to a minimum, which is physically wrong.
To demonstrate this effect, we consider a simplified bundle adjustment problem of the imaging setup of Figure 3.13. In this case, the camera is allowed to move only along the z axis, while all the other parameters – including the structure points – are fixed. For different distance

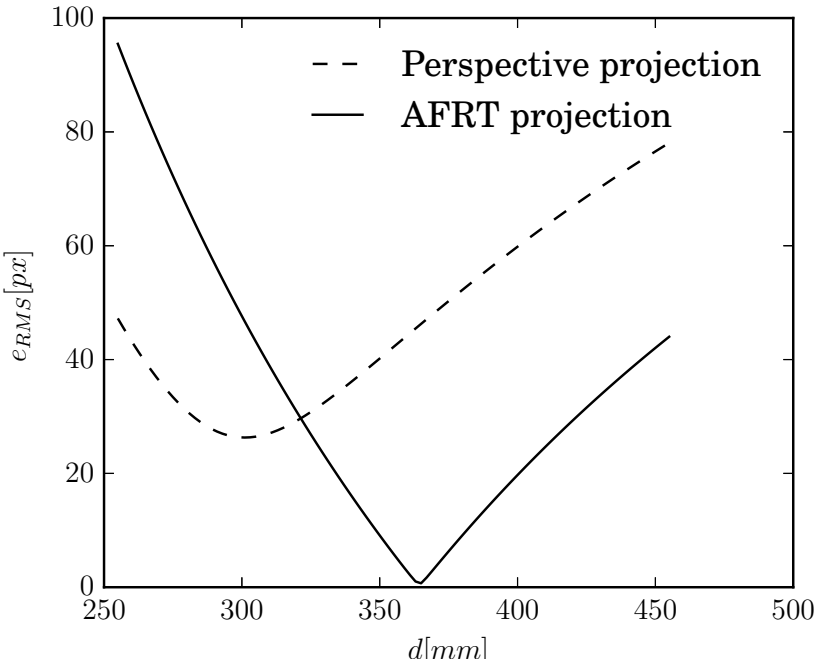


Figure 3.15: *Root mean squared reprojection error for different camera distances for the checkerboard example. With standard perspective projection, the minimum occurs at the incorrect distance $d = 300\,mm$. The AFRT algorithm delivers a well-defined minimum at $d = 360\,mm$, which corresponds to the reference configuration [18].*

values d, the root mean squared (RMS) error of the reprojection is computed as:

$$e_{RMS}^{P} = \sqrt{\frac{1}{n}\sum_{j}\left(\tilde{\mathbf{x}}_j - \mathbf{x}_j^P\right)^2}, \tag{3.46}$$

where n denotes the number of checkerboard corners. The root mean squared error for the AFRT projection can be computed similarly. Figure 3.15 shows the RMS error for different distance values. As expected, when using the AFRT algorithm, the minimum of the RMS error occurs when d is equal to the camera distance used in the ray tracing software. In contrast, the standard perspective projection attains its minimum at a different distance value d, which does not correspond to the real location of the camera. A similar effect can be observed if the camera parameters are kept fixed and the structure points $\mathbf{X}_j$ are allowed to move. To this end, the setup of Figure 3.13 is extended by two additional cameras, as depicted in Figure 3.16. If the AFRT algorithm in the bundle adjustment problem, the resulting minimizer set of structure points matches the reference points $\mathbf{X}_j^G$ with high accuracy. In contrast, if the refractive effects are disregarded, the structure points differ significantly from the reference configuration, as depicted in Figure 3.17. Table 3.1 summarizes the errors with respect to the ground truth geometry along the main directions of the world coordinate system as well as the root mean square of the reprojection error obtained from the solution of the bundle adjustment problem. Note that even though the values of the reprojection errors are in the same order of magnitude, there is a large difference in the error of the structure points. This implies that neglecting the refraction of light in the bundle adjustment formulation leads to

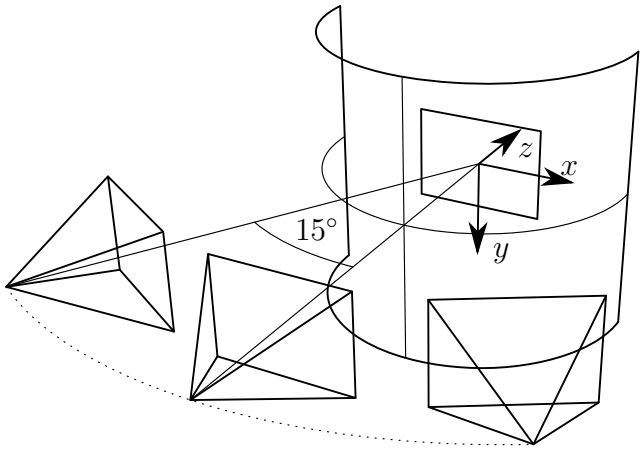


Figure 3.16: *Multi-camera setup of the checkerboard test example [18].*

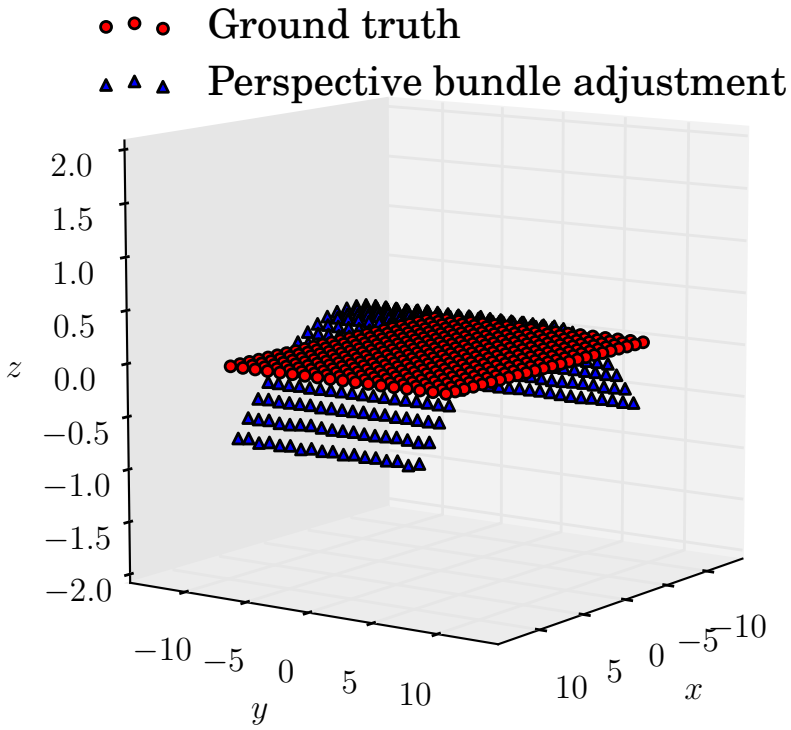


Figure 3.17: *Ground truth points and structure points found by bundle adjustment using perspective projection, for the multi-view checkerboard example [18].*

Method	ΔX_{RMS}	ΔY_{RMS}	ΔZ_{RMS}	$\Delta \mathbf{X}_{RMS}$	e_{RMS}
Perspective projection	0.601 mm	0.8052 mm	0.3193 mm	0.609 mm	0.59 px
AFRT projection	0.0055 mm	0.0090 mm	0.0052 mm	0.0068 mm	0.84 px

Table 3.1: *Summary of error values for the checkerboard example. Even though the reprojection error e_{RMS} is in the same order of magnitude for both methods, there is a significant difference in the error of the structure points [18].*

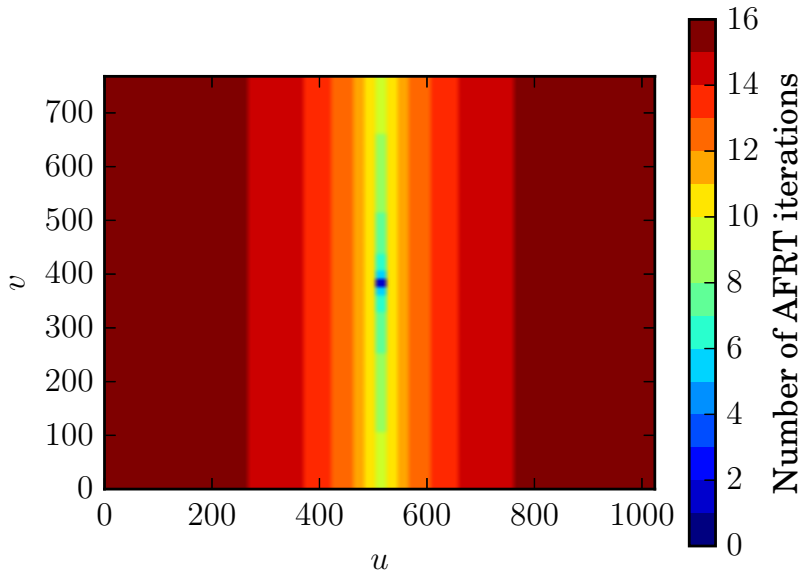


Figure 3.18: *Number of AFRT iterations across the image coordinates for the checkerboard example. Both coordinate axes are in pixel units [18].*

an error both in the estimated structure points as well as in the estimated camera positions. Another important aspect that needs to be considered when using the AFRT algorithm is the increase in time needed to solve the bundle adjustment problem. With standard perspective projection, the image of a structure point on the image plane can be computed directly by evaluating Equation 3.5. The AFRT procedure, however, requires an evaluation of the ray-surface intersections and the law of refraction across many iterations. Obviously, the higher the refractive effects, the more AFRT iterations are required. For the checkerboard example, this relationship is depicted in Figure 3.18. As expected, the rays that are associated to pixels around the image center – where refraction plays a minor role – require less computation. For pixels that are further away from the center, the required number of iterations increases, which directly affects the overall runtime of the bundle adjustment problem. These considerations need to be taken into account when deciding how the object is to be positioned inside the fluid container. Concerning tubular structures, this usually means that the orientation of the structure should be chosen such that it is aligned with the principal axis of the cylindrical container.

3.4.2 Tubular object in synthetic images

This section demonstrates the reconstruction of a tubular object on a set of pictures obtained from a ray tracer software. In this setup, an object $\mathcal{M}$ (depicted in Figure 3.19[†]) is immersed in a liquid with known optical properties. The geometry was constructed by sweeping a circle along a parametric generator curve defined by:

$$\mathbf{g}(t) = \left[t, sin\left(\frac{2\pi}{30}t\right), 0\right]^T, t \in [0, 30]\,. \tag{3.47}$$

In the sweeping process, the radius of the circle is increased linearly along the generator curve, such that $r(t = 0) = 1$ and $r(t = 30) = 3$. The shape of the refractive interface is cylindrical, with a radius $R = 50$mm and the camera moves along a circular path around the object as shown in Figure 3.4. To investigate how the method behaves for various object poses, the reference geometry was placed inside the water container in three different configurations, which are depicted in Figure 3.20. In these configurations, the angle between the principal axis of the object and the vertical direction is 0°, 45° and 90°, respectively. The center and the axis of the camera path was deliberately chosen in such a way that it does not coincide with the location and the axis of the refractive interface, nor with the object. Figure 3.21 shows examples of the captured images for pose A at different camera positions.

The solution of the bundle adjustment problem of Equation 3.28 yields a set of parameters that minimize the sum of the squares of the reprojection errors for all feature points over all images. To measure how accurately the bundle adjustment method is able to fit the model parameters to the observations, we look at the root mean square of the reprojection errors:

$$e_{RMS} = \sqrt{\frac{1}{n}\sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - P_{AFRT}\left(\mathbf{X}_j, \mathbf{K}, \mathbf{n}, \mathbf{d}, \theta_i, \sigma\right)\right)^2}, \tag{3.48}$$

where n denotes the total number of residuals. Secondly, knowing the geometry of $\mathcal{M}$ allows to investigate the accuracy of the optimized set of structure points $\{\mathbf{X}_i\}$, by computing the signed distance of each $\mathbf{X}_i$ with respect to the model $\mathcal{M}$, denoted by $d(\mathbf{X}_i, \mathcal{M})$. To calculate the signed distance, the closest orthogonal projection of every $\mathbf{X}_i$ onto $\mathcal{M}$ is computed. This way, the mean signed distance with respect to the ground truth is calculated as:

$$e_{\mathcal{M}} = \frac{1}{N}\sum_i d(\mathbf{X}_i, \mathcal{M}). \tag{3.49}$$

In order to compare the method to the standard bundle adjustment, the reconstruction of the artificially generated images was also performed without taking the refractive effects into account, and allowing the cameras to move arbitrarily.
Table 3.3 summarizes the results for the different configurations. Concerning the number of points and the precision of the resulting point cloud, pose A outperforms the two other settings. On the other hand, similarly to pose C, it requires more iterations in the solution process than pose B. The reason for this is that the projection of the object in the first and third configuration only covers a rather narrow region on the captured images as the camera

[†]The reference geometry was designed and manufactured by Ofry Yossef at the Experimental Biomechanics Lab of the Ben-Gurion University of the Negev, Israel. This support is gratefully acknowledged.

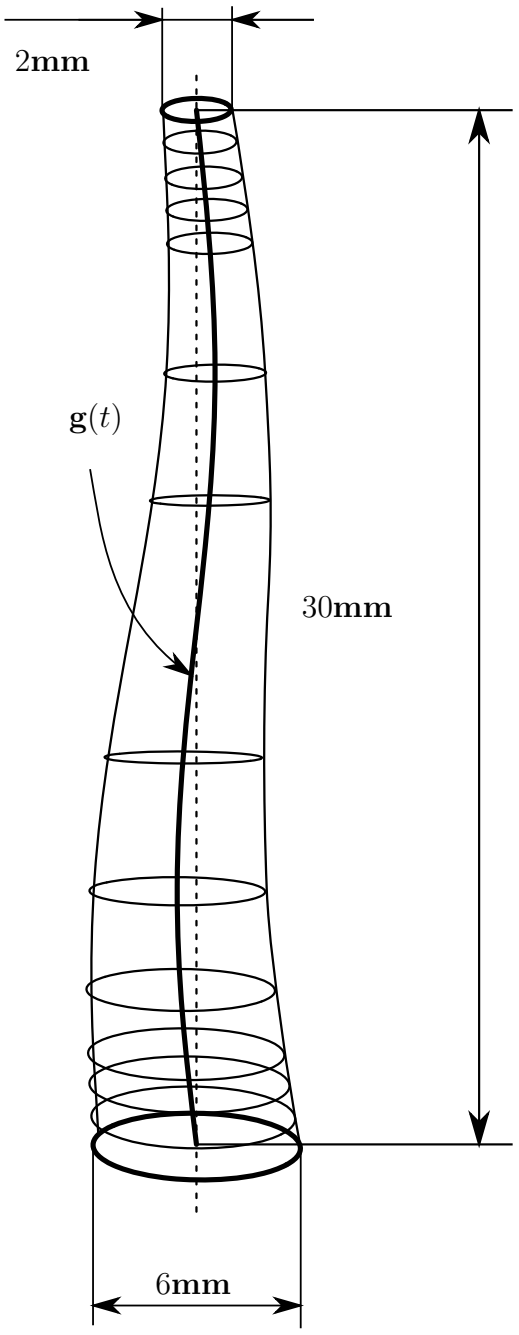
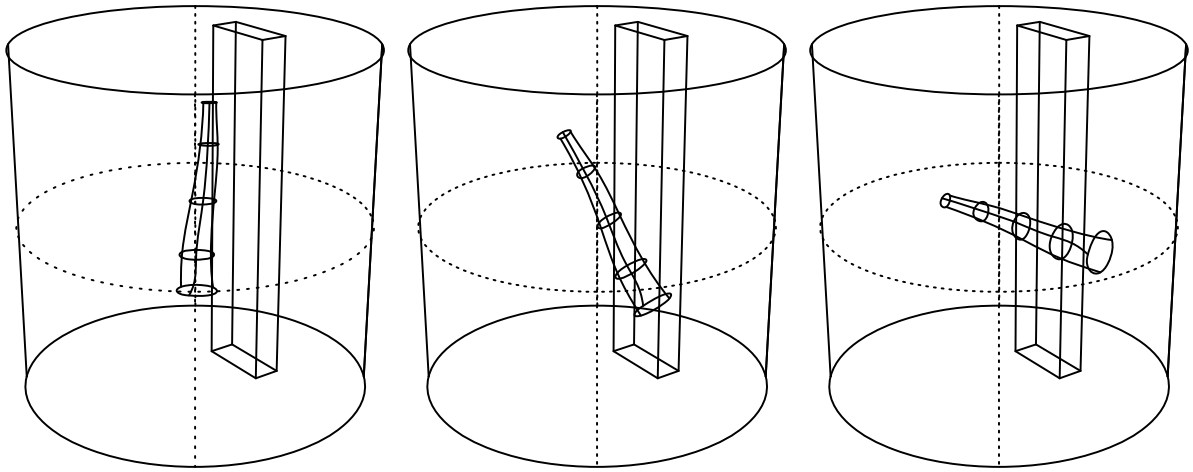
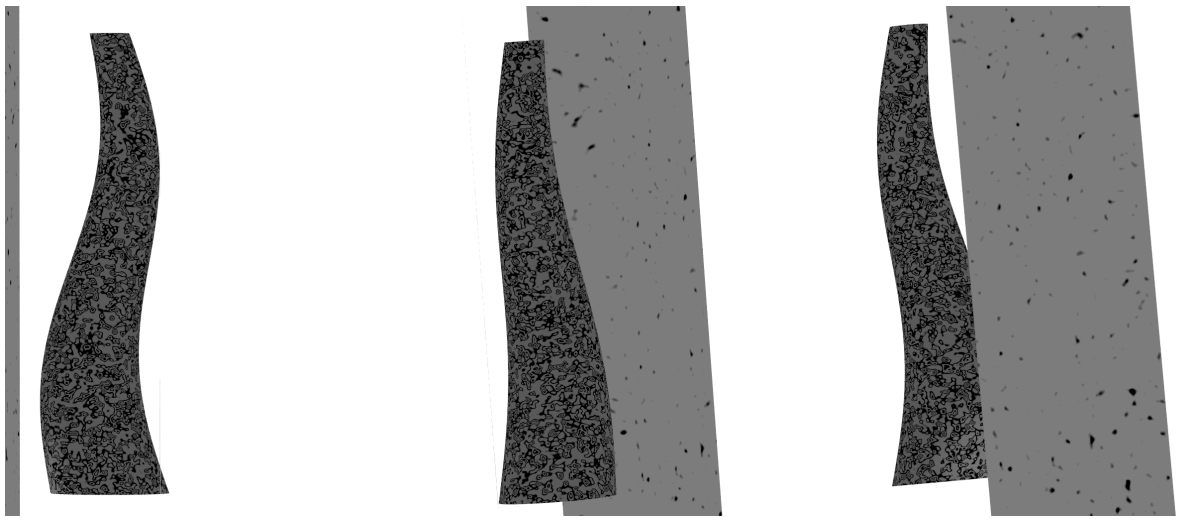


Figure 3.19: *Reference geometry $\mathcal{M}$ used in the verification example [18].*



(a) Pose A (b) Pose B (c) Pose C

Figure 3.20: *Different object poses used in the synthetic example [18].*



(a) (b) (c)

Figure 3.21: *Synthetic images of pose A. The object was immersed in water and the pictures were recorded under different rotations θ of the camera: (a) $\theta = 0°$, (b) $\theta = 108°$, (c) $\theta = 228°$ [18].*

Modified bundle adjustment							
Pose	e_{RMS}	$e_{\mathcal{M}}$	# points	# residuals	# iterations	Avg. # AFRT iterations	Time per residual
A	$0.3987\,px$	$0.0059\,mm$	14467	197428	27	18	$2.56 \cdot 10^{-5}\,s$
B	$0.2575\,px$	$0.026\,mm$	14011	148042	16	21	$2.76 \cdot 10^{-5}\,s$
C	$0.306\,px$	$0.1575\,mm$	9185	101672	27	23	$3.42 \cdot 10^{-5}\,s$
Standard bundle adjustment							
A	$0.2551\,px$	$0.8701\,mm$	14399	196276	35	–	$5.72 \cdot 10^{-7}\,s$
B	$0.7366\,px$	$0.6772\,mm$	13917	143854	13	–	$5.73 \cdot 10^{-7}\,s$
C	$0.702\,px$	$0.430603\,mm$	9185	101672	13	–	$5.62 \cdot 10^{-7}\,s$

Table 3.2: *Comparison of bundle adjustments for different poses of the verification geometry [18].*

rotates, leaving most of the image area unexploited. As demonstrated by the checkerboard example in Section 3.4.1, the least amount of AFRT iterations is needed if the object is oriented along the principal axis of the water container. In contrast, if the object is oriented perpendicularly – as in the case of pose C – the number of iterations of the AFRT algorithm increases. This also correlates with the average cost of evaluating a single residual.
Similarly to the checkerboard example in Section 3.4.1, when the refractive effects are neglected, the average mean distance of the resulting structure points $\{\mathbf{X}_i\}$ is rather high, even though the standard bundle adjustment yields a set of parameters that minimize the reprojection errors to a sub-pixel accuracy. This is due to the fact that in the standard case, the standard bundle adjustment finds a set of parameters that minimizes the cost function of Equation 3.21, but this optimal set of parameters does not represent the real structure at all. In contrast, the structure points resulting from the modified bundle adjustment approach – which takes the refractive effects into account – are orders of magnitude closer to the ground truth shape $\mathcal{M}$, while the reprojection errors are in the same order of magnitude. This high difference in the mean signed distances can be observed both for the synthetically generated images and for those recorded in the real setting, as will be shown in the next section.

3.4.3 Tubular objects represented on real images

For comparison, the experiment of the previous section was repeated in a real setting. In this case, the geometry was manufactured using a 3D printer and it was immersed into a cylindrical water container, according to pose A from the previous section. To enhance the quality of the surface texture, the model was sprayed with black ink using an airbrush. The resulting dot-like pattern covers approximately 40% of the surface. A camera was moved around the object using an automatic device (Figure 3.22[†]), allowing for accurate angular positioning and keeping the radius of the camera's path constant. This automatic device is part of an experimental apparatus built for the purpose of investigating the active biomechanical behavior of human

[†]The experimental device was designed and constructed at the Experimental Biomechanics Lab of the Ben Gurion University of the Negev, Israel. The image was taken by Avihai Uzan. This effort is greatly acknowledged.

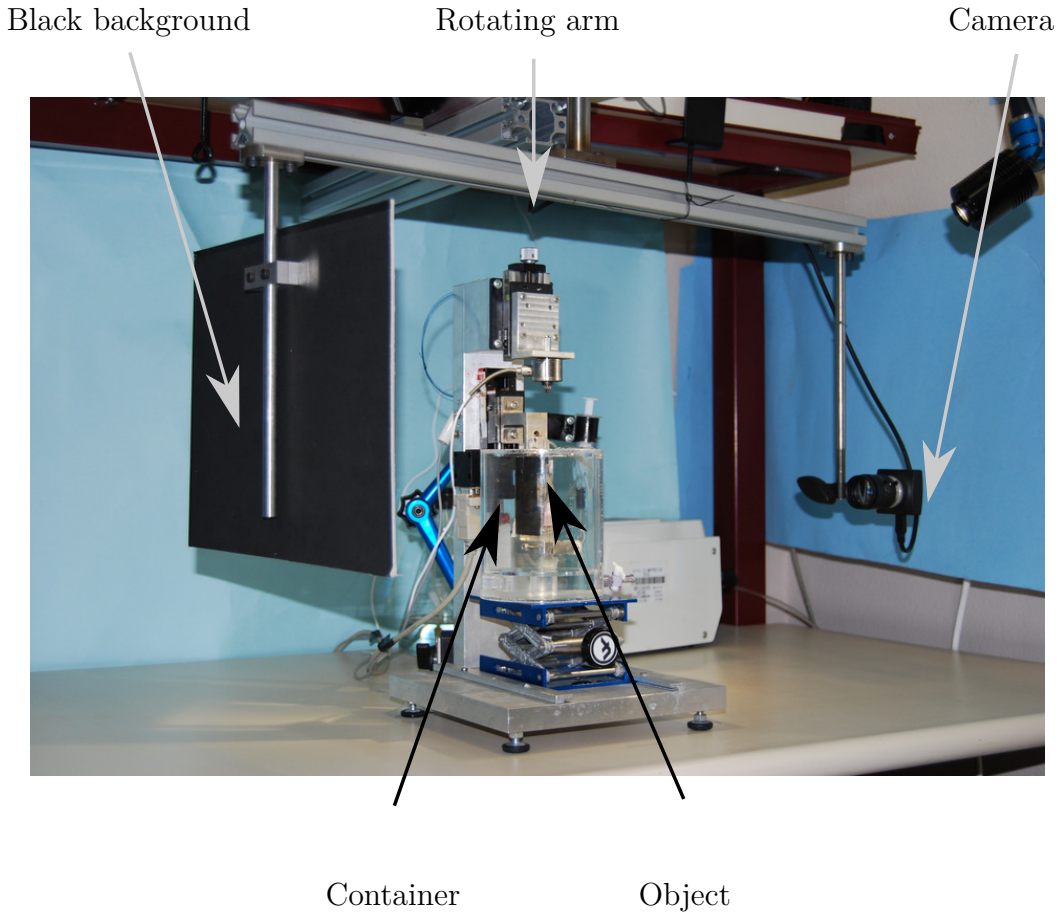
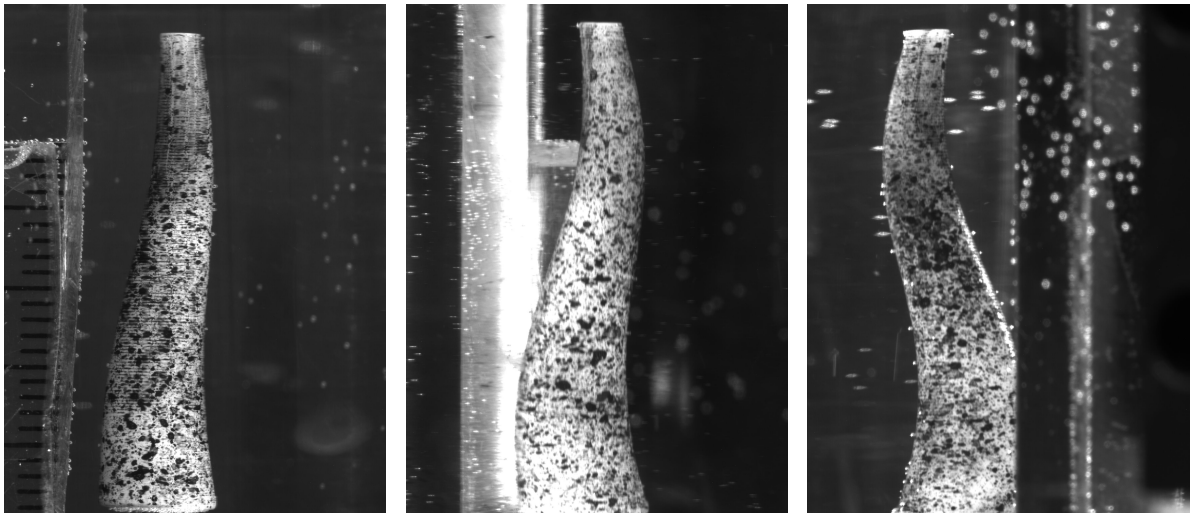


Figure 3.22: *Laboratory setup of the verification example. The camera is mounted on an arm that rotates around a cylindrical container. To avoid unwanted background features, a black sheet of light-absorbing material is mounted on an arm opposite to the camera. The 3D printed reference object is placed inside the container [18].*



(a) (b) (c)

Figure 3.23: *Images taken of the object immersed in water, under different rotation angles θ of the camera: (a) $\theta = 0°$, (b) $\theta = 108°$, (c) $\theta = 228°$ [18].*

Method	e_{RMS}	$e_{\mathcal{M}}$	# points
Standard B.A.	0.2551 px	0.8701 mm	7727
Modified B.A.	0.3987 px	0.0059 mm	8800

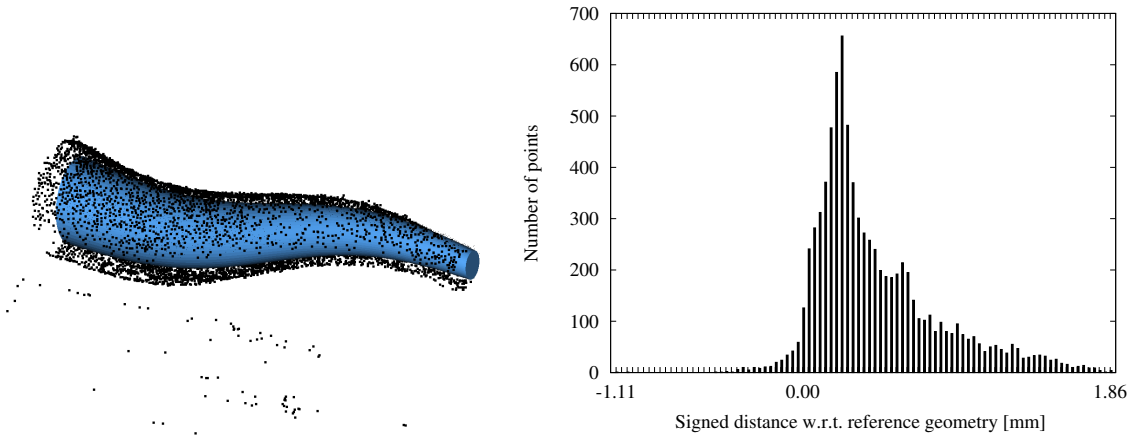
Table 3.3: *Comparison of the results of standard bundle adjustment and its modified version for the real image set.*

arteries. A detailed description of these experiments and the apparatus can be found in [145]. Prior to the experiment, the intrinsic parameters were calibrated using the method of [146]. With an increment of 10.8° in the angular position of the camera, 22 images of the object were captured. Examples of the images taken in the real setting are depicted in Figure 3.23†. The results of the modified bundle adjustment on the real image data are summarized in Table 3.3. Similar to the examples with the synthetic images, the root mean squared of the reprojection error is in the same order of magnitude for both methods. However, the modified bundle adjustment formulation yields points that lie orders of magnitudes closer to the ground truth geometry. The point sets and histograms of the signed distance errors for this scenario are depicted in Figure 3.24.

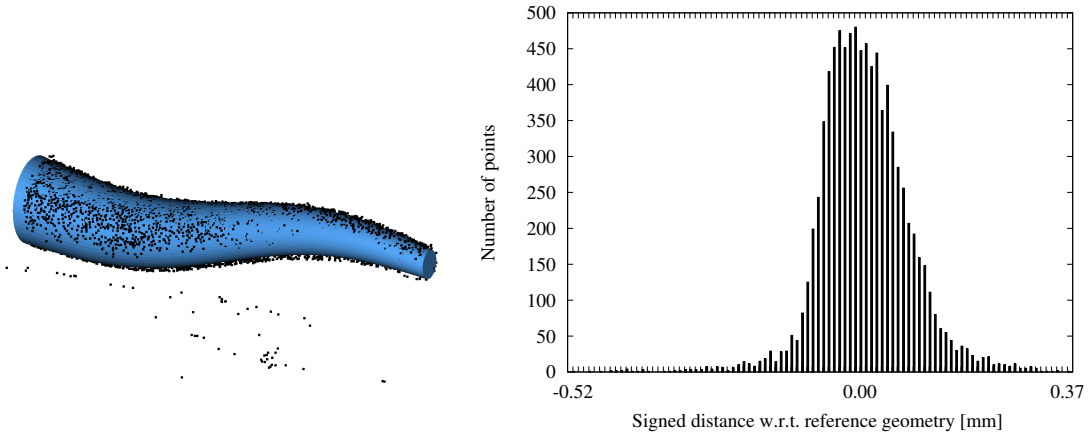
As shown in Figure 3.24, the resulting point set $\{\mathbf{X}_i\}$ contains points that belong to the experimental environment. These outlier points are filtered out using the RANSAC algorithm. Apart from the remaining inlier points $\left\{\mathbf{X}_i^{\mathcal{M}}\right\}$, RANSAC also provides the parameters of the cylinder which is the best fit to these points.

Concerning the points computed by the modified bundle adjustment method, the estimated diameter given by the RANSAC algorithm is 2.47763 mm. This value corresponds to a difference of 1% compared to the average radius of the reference geometry $\mathcal{M}$. In contrast, the points resulting from the standard bundle adjustment have a diameter of 3.40295 mm, which

†These images were taken by Ofry Yossef at the Experimental Biomechanics Lab of the Ben Gurion University of the Negev, Israel. This effort is greatly acknowledged.

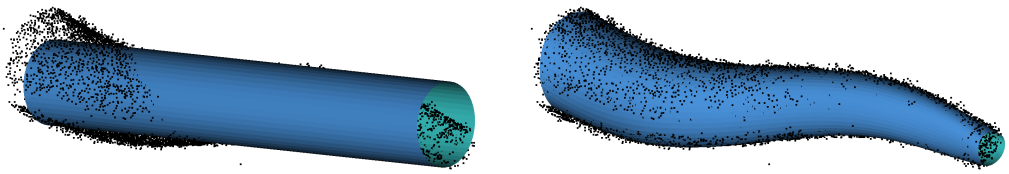


(a) Standard bundle adjustment



(b) Modified bundle adjustment

Figure 3.24: *Point clouds resulting from the real image set and the respective histograms of signed distance errors, for (a) standard bundle adjustment and (b) modified bundle adjustment [18].*



(a) Inlier points and best-fit cylinder (b) Surface fitted onto the inlier points

Figure 3.25: *Initial estimate and fitted surface [18].*

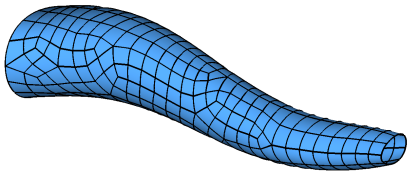


Figure 3.26: *Surface mesh: quadratic quadrilateral elements [18].*

means a difference of 40%.
The best-fit cylinder is used as the initial estimate for the active surface fitting algorithm of [126], described in Section 3.3.2. Starting from an initial 300 points, the number of active surface points is increased by 100 in every iteration, and the iteration is repeated until the root mean squared distance of the active surface with respect to the point cloud falls below 0.01 mm. The resulting surface $\mathbf{S}_{\mathcal{M}}(u, v)$ is a tensor product B-Spline surface approximating $\mathcal{M}$. This surface is stored as a boundary representation model (BRep) for the final step. The initial estimate and the result of the surface fitting algorithm are depicted in Figure 3.25. In the final step of the reconstruction procedure, a mesh of curved hexahedral finite elements is created by employing the method of [28]. Figure 3.26 shows an example mesh with quadratic elements. For this mesh, the average signed distance of the interpolation points with respect to the reference geometry $\mathcal{M}$ is 0.009 mm.

To obtain a volumetric mesh of high order finite elements, the quadrilateral elements are swept in the direction of the local surface normals of $\mathbf{S}_{\mathcal{M}}(u, v)$. In this example, the thickness of the geometry was chosen to be a constant of $t = 0.5$ mm throughout the entire domain. Examples of quadratic and quartic hexahedral elements are depicted in Figure 3.27.

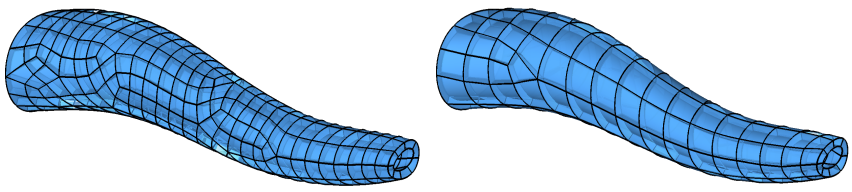


Figure 3.27: *Volumetric mesh: quadratic and quartic hexahedra [18].*

3.4.4 Tubular object with a branch represented on real images

As a final demonstration of the entire mesh generation procedure, we consider a branched geometry – represented by the intersection of two tubular objects, as depicted in Figure 3.28. The length of the object is $150mm$, an order of magnitude bigger than in the previous example. The texture of the object was enhanced by spraying black ink on its surface. Similar to the previous example, the object was placed in a cylindrical water container, and 36 pictures were recorded by moving the camera around it along a circular path. As an example, Figure 3.29 depicts one of the images captured by the camera. As the first step in the procedure, a point cloud is generated using the modified bundle adjustment formulation. The resulting points are depicted in Figure 3.30. The average distance of the resulting 10626 structure points to the ground truth geometry is $0.19\,mm$.

Figure 3.28: *Tubular geometry with a branch [18].*

Figure 3.29: *Image taken of the branched geometry in the experimental system [18].*

After solving the modified bundle adjustment problem, the RANSAC algorithm identifies the two main cylindrical components in the cloud. The points are separated according to the cylindrical model they belong to. The resulting cylinders represent the topology of the object, as shown in Figure 3.31 as well. The two cylinders resulting from the RANSAC algorithm are converted to separate B-Spline surfaces and used as initial guesses for the iterative procedure of Section 3.3.2. This process finds the best-fit surface to the separated points. The resulting surfaces are intersected and joined together to form the final geometric model, which is depicted in Figure 3.32.

Finally, a high order finite element mesh is generated on the surface using the algorithm of [28]. An example mesh with quadratic hexahedral elements is depicted in Figure 3.33. The

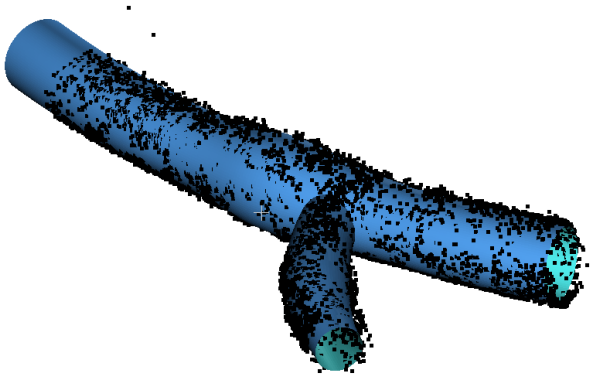


Figure 3.30: *Point cloud resulting from bundle adjustment, and the ground truth geometry [18].*

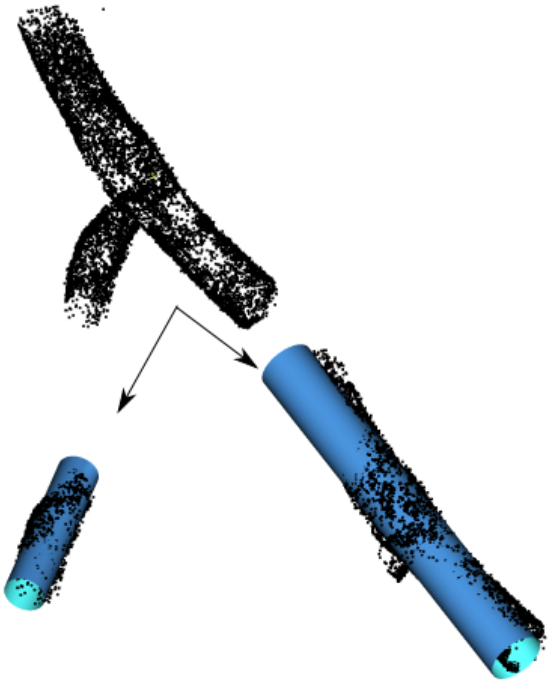


Figure 3.31: *The RANSAC algorithm separates the point cloud according to the underlying cylindrical models that describe its topology [18].*

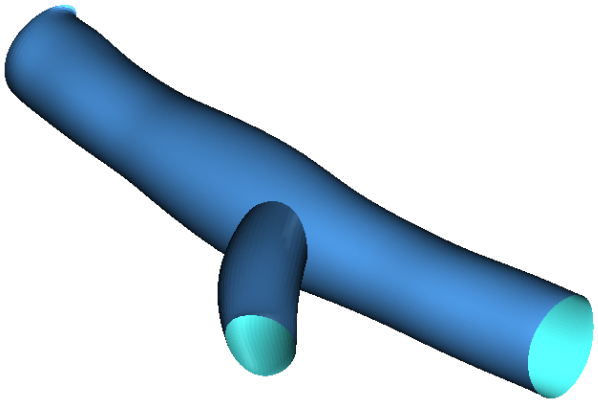


Figure 3.32: *B-Spline surfaces fitted onto the points separated by the RANSAC step [18].*

nodes lie within 0.25 mm of the ground truth geometry, the bounding box of which measures approximately 150 mm.

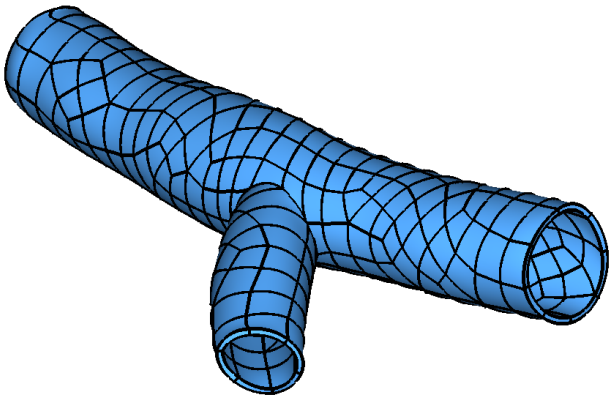


Figure 3.33: *Mesh of quadratic hexahedral finite elements, generated on the surfaces resulting from the surface fitting [18].*

Chapter 4

The Finite Cell Method combined with oriented point clouds

4.1 The role of geometric representations

The Finite Cell Method has been applied in combination with a large variety geometric representations. If the geometric model provides detailed, high order surface information (which is the case for geometries stored in IGES or STEP file formats), the accurate decomposition algorithm described in Section 2.2.2.4 may be applied, which delivers highly accurate integrals with an almost minimal number of integration points. However, even if no explicit surface information is available in the geometric model, (which is the case for voxel models for example), numerical integration can still be performed by using the spacetree decomposition of Section 2.2.2.1. As seen there, in order to generate a spacetree, it is sufficient if the algorithm can determine whether a cell is cut by the boundary of the physical domain or not. To this end, computing the inside-outside state of a set of test points and comparing them pairwise is necessary. Furthermore, at every integration point, the indicator function α needs to be evaluated, which also requires a point-membership classification from the geometric model.

Following these considerations, it can be seen that in its simplest implementation, the only requirement of the finite cell method from a geometric model is the ability to determine the inside-outside state: given a point $\boldsymbol{x} \in \mathbb{R}^d$ in space, does this point belong to the physical or the fictitious part of the domain? As long as a geometric model is able to provide such *point-membership tests*, it is possible to conduct a finite cell analysis.

The point clouds that result from the shape measurement procedures of Chapter 3 do not only contain points, but also the normal vectors at these points. These normals carry information about the orientation of the local tangent planes of the object of interest. While this normal information is an essential ingredient of many surface reconstruction algorithms (see in Section 3.3), it can also be used to carry out the point membership test, which is sufficient for the finite cell method. This allows for taking a major shortcut in the measurement-to-analysis pipeline: instead of having to recover a geometric model and a boundary-conforming finite element mesh, the finite cell method can be used to compute the structural behavior directly on domains defined by oriented point clouds. This idea is explored in the following parts of this chapter, demonstrating how the method can be applied on geometries encountered in standard engineering practice.

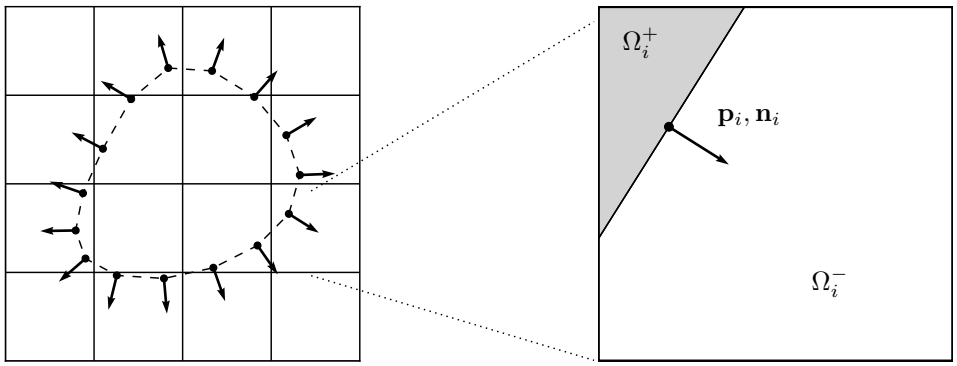


Figure 4.1: *Point membership classification on oriented point clouds. The domain is represented by a set of points* $\mathbf{p}_i$ *and associated normals* $\mathbf{n}_i$*. Every such pair locally separates the space along a hyperplane into two half-spaces:* Ω_i^- *and* Ω_i^+ *[4].*

4.2 Point membership tests on oriented point clouds[d]

In point cloud-based simulations, the domain Ω_{phy} is represented by a set of sample points $\mathbf{p}_i$ and their associated normal vectors $\mathbf{n}_i$. If no outliers are present, the set of pairs $S = \{\mathbf{p}_i, \mathbf{n}_i\}$ represent a discrete sampling of the boundary $\partial\Omega_{\text{phy}}$ of the domain.

Each element in S defines a hyperplane that separates the space in two half spaces: the open half-space Ω_i^- lying on the side of the hyperplane where the normal vector $\mathbf{n}_i$ points, and the closed half-space Ω_i^+ lying on the other side. This concept is depicted in Figure 4.1. For every $\mathbf{x} \in \Omega_i^+$, the following holds:

$$(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i \geq 0. \tag{4.1}$$

Therefore, a simple way to estimate whether a quadrature point $\mathbf{q}$ lies inside or outside the domain is to find the $\mathbf{p}_i$ and the associated $\mathbf{n}_i$ in S that lies closest to $\mathbf{q}$ and evaluate the scalar product of Equation 4.1. While the heuristic nature of this approach may give the impression that it only works for simple shapes, the practical examples in Section 4.6 on more complex geometries show that the recovered indicator function is suited for performing a finite cell analysis in these cases as well. The algorithm requires an efficient nearest neighbor query. To this end, the k-d tree implementation of the C++ library *nanoflann* [147] is used, while the clouds themselves are represented by the data structures of the Point Cloud Library [148]. The point membership classification method is summarized in Algorithm 1.

Conceptually, this approach is a simplified version of the method explained in [129]. In this work, the authors propose to reconstruct the surface of an object represented by an unorganized set of points by extracting the zero level-set of a signed geometric distance function. The sign of the distance function is determined by evaluating the scalar product of Equation 4.1, and its absolute value by computing the distance to the closest point. The difference of Algorithm 1 to the method proposed therein is that for the finite cell method, only the inside-outside state of a query point is required. Thus, for the purpose of an FCM simulation, only the sign of the signed distance function is relevant, while its absolute value, which would be needed for a subsequent surface recovery step, can be disregarded.

Algorithm 1: *Point membership test for oriented point clouds*

```
function isPointInside (q, S) ;
  Input  : Quadrature point q and oriented point cloud S = {p_i, n_i}
  Output: Boolean true if q lies inside the domain represented by S, false otherwise
p_i, n_i = getClosestPointInCloud(q, S);
v = p_i − q;
d = v · n_i;
if d ≥ 0 then
    return true;
end
return false;
```

4.2.1 Connection to Voronoi diagrams

Starting from the points $\boldsymbol{p}_i$ in the input point cloud, the embedding space $\mathbb{R}^d$ can be partitioned into disjoint regions, each denoted by $\mathcal{V}(\boldsymbol{p}_i)$, such that every $\boldsymbol{x} \in \mathcal{V}(\boldsymbol{p}_i)$ lies closer to $\boldsymbol{p}_i$ than to any other point $\boldsymbol{p}_j \in S, j \neq i$. The disjoint regions $\mathcal{V}(\boldsymbol{p}_i)$ are called *Voronoi cells*. The set of Voronoi cells partitions the embedding space into n regions, usually referred to as the *Voronoi diagram* $\mathcal{V}(S)$ for the set S of n points [149, 150].

As shown in Figure 4.2a, the hyperplane defined by the pair $\{\boldsymbol{p}_i, \boldsymbol{n}_i\}$ splits the corresponding Voronoi cell $\mathcal{V}(\boldsymbol{p}_i)$ into two halves, $\Omega^-_{\mathcal{V}_i}$ and $\Omega^+_{\mathcal{V}_i}$:

$$\begin{aligned} \Omega^-_{\mathcal{V}_i} &= \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x} \in \mathcal{V}(\boldsymbol{p}_i) \wedge (\boldsymbol{p}_i - \boldsymbol{x}) \cdot \boldsymbol{n}_i < 0\} \\ \Omega^+_{\mathcal{V}_i} &= \{\mathbf{x} \in \mathbb{R}^d | \mathbf{x} \in \mathcal{V}(\boldsymbol{p}_i) \wedge (\boldsymbol{p}_i - \boldsymbol{x}) \cdot \boldsymbol{n}_i \geq 0\}. \end{aligned} \tag{4.2}$$

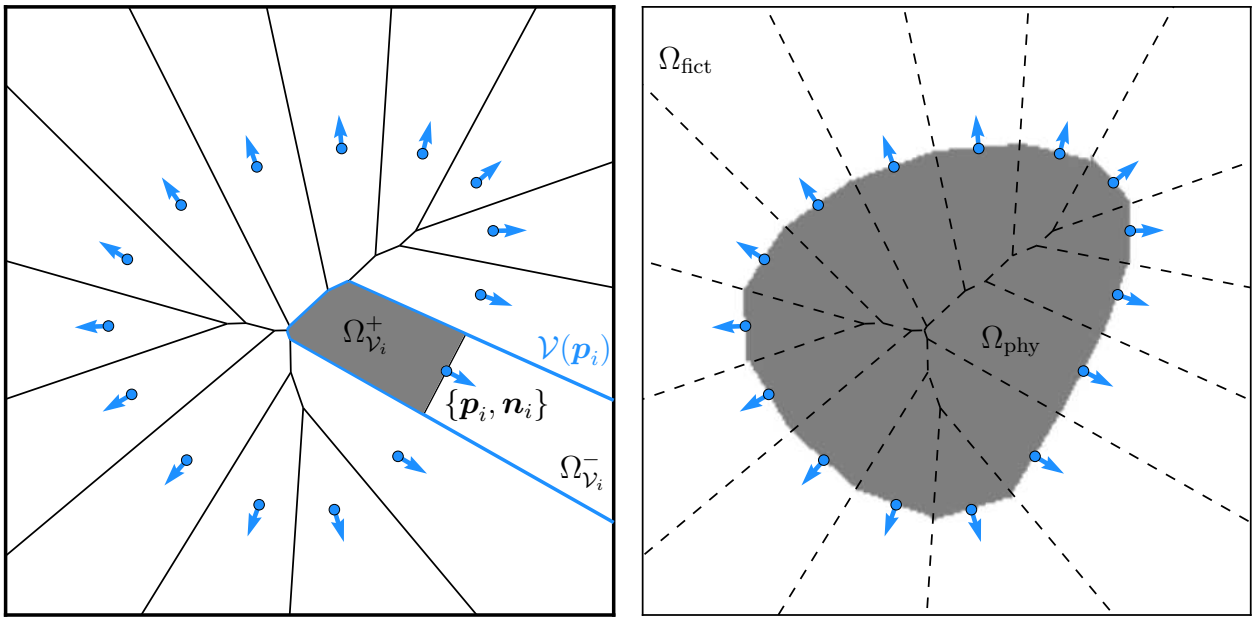
If a query point $\boldsymbol{q} \in \mathbb{R}^n$ is located in a Voronoi cell $\mathcal{V}(\boldsymbol{p}_i)$, its inside-outside state is determined by the point $\boldsymbol{p}_i \in S$ and the associated normal vector $\boldsymbol{n}_i$. As the Voronoi diagram partitions $\mathbb{R}^d$ into disjoint regions, there is a single Voronoi cell that any $\boldsymbol{q}$ can be associated to. It follows that the physical domain Ω_{phy} recovered by Algorithm 1 is the union of the inside part of all the Voronoi cells in $\mathcal{V}(S)$:

$$\Omega_{\text{phy}} = \bigcup_{i=0}^{n} \Omega^+_{\mathcal{V}_i}, \tag{4.3}$$

where n denotes the number of points in the point cloud. This concept is illustrated in Figure 4.2b.

4.3 Treatment of outliers

Sometimes, the cloud acquired by the shape measurement process carries outliers. As defined in [151], outliers are *"observations that deviate so much from other observations as to arouse suspicion that it was generated by a different mechanism."* The detection and treatment of outliers is a well-studied problem in the literature and lies beyond the scope of this thesis, see e.g. the methods in [151–157]. These algorithms can be applied in a pre-processing step



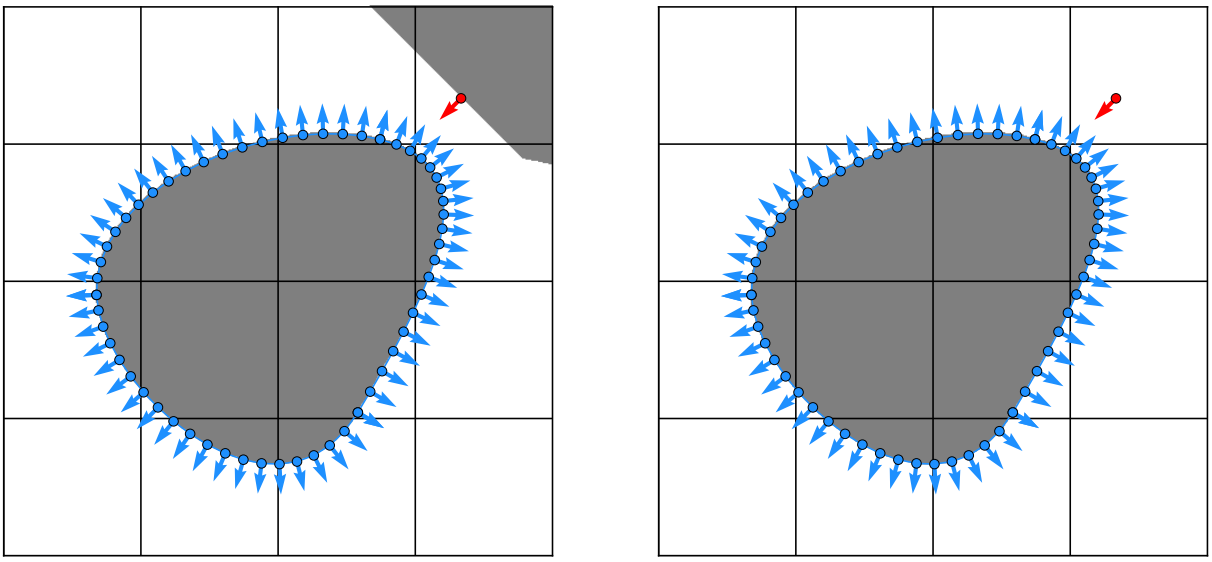
(a) Voronoi diagram showing one cell $\mathcal{V}(\boldsymbol{p}_i)$ split in halves.

(b) Inside-outside state computed by Algorithm 1

Figure 4.2: *The connection between Algorithm 1 and Voronoi diagrams. Every Voronoi cell* $\mathcal{V}(\boldsymbol{p}_i)$ *is split into two halves according to the hyperplane defined by* $\{\boldsymbol{p}_i, \boldsymbol{n}_i\}$*. The union of the inside part of all the Voronoi cells is* Ω_{phy}*, that Algorithm 1 recovers.*

in order to recover a clean cloud where the majority of the outliers have been removed. The cleaning procedure is an essential step also in the traditional measurement-to-analysis pipeline, as most of the geometry recovery algorithms rely on a clean input cloud.

While the cleaning process is able to remove the bulk of the outliers, if isolated outliers remain in the cloud, they may have an effect on the point-membership classification process of Algorithm 1. To demonstrate this, Figure 4.3a shows the inside-outside state recovered from a point cloud consisting of 90 points and an outlier in the upper right quadrant of the domain. As seen in the figure, the presence of the outlier causes a region in the domain to be falsely detected as "inside". To deal with isolated outliers, the point membership test of Algorithm 1 is modified by testing in the n-neighborhood of the query point. In this process, instead of checking against a single closest point, the n nearest points of $\boldsymbol{q}$ are found and the point membership with respect to each of them is computed. If $\boldsymbol{q}$ lies inside with respect to the majority of the points in the n neighborhood, its membership is determined as inside, otherwise outside. Following this idea, Figure 4.3b depicts the inside-outside state recovered using 3 nearest neighbor queries. The choice of n for the number of nearest neighbor queries is a parameter that needs to be determined prior to the simulation. In the examples of Section 4.6, the parameter is chosen in the range $n = 5...50$.



(a) Inside-outside testing using a single nearest neighbor

(b) Inside-outside testing using 3 nearest neighbors

Figure 4.3: *The effect of a single outlier (red point) on point-membership tests. The black and white regions represent parts detected as inside and outside, respectively [4].*

4.4 Treatment of missing parts

4.4.1 The effect of holes on point-membership classification

In some cases, the employed shape measurement technique may not be able to recover the surfaces of the object of interest completely. If this situation arises, the resulting point cloud has some missing parts. Such holes are frequently encountered if there is no direct line of sight from the recording device to some regions of the object; when a part of the surface is occluded, it will not be represented in the point cloud [106]. Missing parts may also appear for surfaces with certain physical properties, such as highly transparent or reflective objects. Further, for image-based multi-view stereo reconstructions, the lack of adequate surface texture may result in severe undersampling of the surfaces, which can also be regarded as holes.

The presence of holes poses a challenge for the point-membership classification algorithm of Section 4.2. To model this scenario, Figures 4.4c and 4.4b depict two reduced versions S^* of the cloud S in Figure 4.4a, such that $S = S^* \cup S^h$, where S^h represents the set of missing points that the (hypothetical) shape measurement process was unable to recover. In the absence of S^h, the shape of the recovered indicator function is determined by the positions and orientations of the remaining points S^*. More precisely, the Voronoi diagram $\mathcal{V}(S)$ changes such that the Voronoi cells associated to S^h are replaced by cells that are generated on the remaining points. Therefore, the shape of the interface in the region of the missing points will be determined by the orientation of the points associated to the new Voronoi cells.

If the shape of the missing region is almost planar, the recovered indicator function does not change substantially. This is because the normal vector of low-curvature regions is almost constant, hence the plane determined by the normals at the remaining points in the vicinity of

the hole can closely approximate the missing part, as also depicted in Figure 4.4b. However, if the missing part stems from a region of high curvature, the recovered indicator function shows high errors with respect to the original geometry. In this case, the recovered planar interface is not able to represent the region of high curvature, but rather creates a pattern which makes the impression that the indicator function was "flowing out", see Figure 4.4c.

4.4.2 Remedy strategies for holes

The literature discusses various approaches to deal with missing parts in the scanned data. There are methods that aim at augmenting the cloud with new points at the holes. As demonstrated in [158], this can be done in a two-step procedure. First, holes and their boundaries are identified, then an algebraic surface is fitted onto the identified regions, extending the shape of the point cloud in accordance with the shape of the vicinity of the boundary. If prior knowledge about the geometry of the object of interest is available (e.g. in the form of a triangulation), a similar two-step approach can be applied to first identify the holes, then smoothly deform the shape prior towards the scanned point cloud data by minimizing the deformation energy, as explained in [159].

Other methods integrate hole filling in the geometry recovery step: a gap that represents a hole is conceptually the same as the space between adjacent points in the cloud. Thus, these methods essentially fill holes during the reconstruction. Approaches that belong to this category are based on alpha-shapes [160, 161], balls [162] or crusts [163, 164]. If the object to be scanned is known to be watertight, those reconstruction methods that are based on fitting an implicit function are able to bridge the gaps that represent the hole [134, 165, 10, 11, 166, 167, 12].

Other approaches aim at filling the holes on the triangulation recovered by the surface fitting procedure. In this process, similar to the methods that operate directly on point clouds, holes are identified then subsequently patched, as explained in [168, 169].

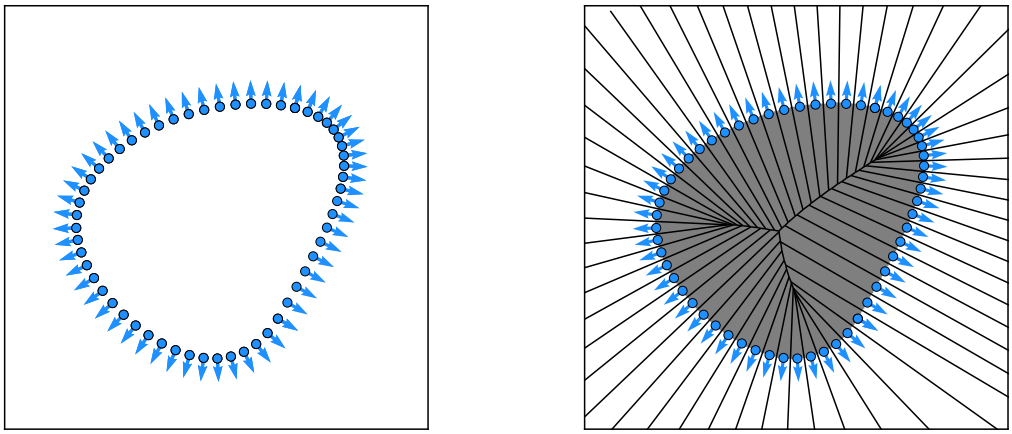
When the amount of missing data is significantly high, holes may be filled by extracting higher-level information from the point cloud, such as skeletons [170–172] or shape primitives [113, 117].

4.4.3 A two-step indicator function recovery for the FCM

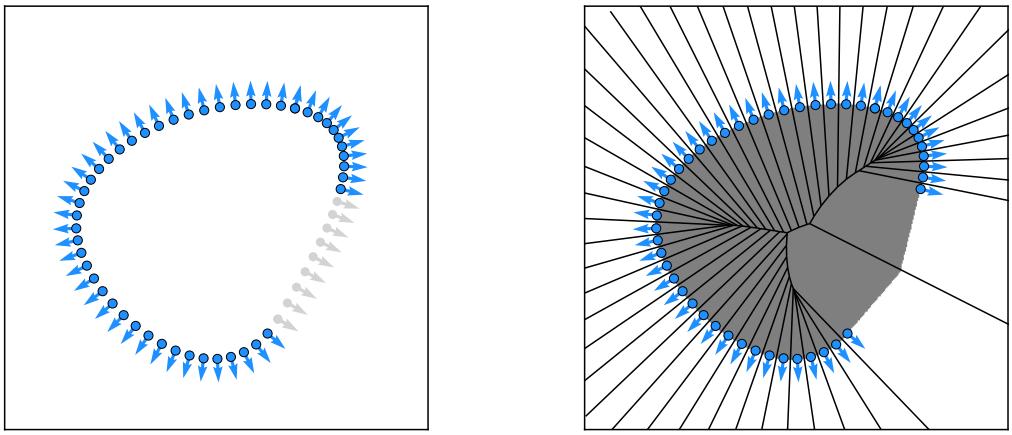
The approaches outlined in the previous section provide ready-to-use solutions to the challenge of missing parts in the measured point cloud. Obviously, the choice of what approach to use depends on the requirements posed by the application. From the perspective of the FCM-based measurement-to-analysis pipeline proposed within this chapter, the resolution strategy addressing the problem of holes should:

1. require as little manual interaction as possible,
2. work robustly for small- and large-sized holes,
3. fill holes, but preserve fine geometric details at the same time.

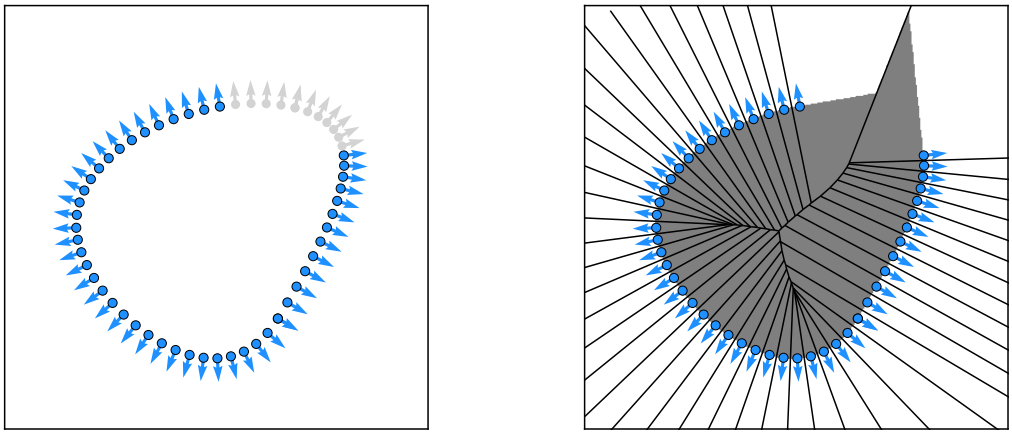
A natural candidate that satisfies the above requirements is the widely-known Poisson surface reconstruction method. In the following, the main ideas of the approach are summarized. For a complete discussion, refer to [10, 11].



(a) Complete point cloud



(b) Missing points in a region with low curvature



(c) Missing points in a region with high curvature

Figure 4.4: *The effect of holes on nearest neighbor-based point membership classification. Left column: point clouds, with gray points representing missing parts. Right column: Voronoi diagrams and the recovered indicator function. The Voronoi cells of the removed points are replaced by larger cells whose shape is determined by the remaining points at the vicinity of the hole. If points are missing in areas of low curvature, the indicator function is not influenced significantly. When points are missing in high curvature regions, the indicator functions "flows out".*

The key insight of the method is that the oriented points in the cloud can be thought of as discrete samples of the gradient of the regularized equivalent of the indicator function, i.e. the function that is one inside the physical domain and zero outside, with a smooth transition in a narrow region between these two states. Therefore, the Poisson surface reconstruction finds a function $\Phi(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, such that its gradient at the location of the sample points matches the normal vectors associated to the samples. Following these considerations, the function Φ can be found by minimizing:

$$\min_{\Phi} \int_{\Omega} ||\nabla\Phi - \boldsymbol{n}||^2 \mathrm{d}\Omega. \tag{4.4}$$

Applying the Euler-Lagrange equation [173], the functional in the equation above is stationary if the following equation holds:

$$\Delta\Phi = \nabla \cdot \boldsymbol{n}. \tag{4.5}$$

The continuous function $\Phi(\boldsymbol{x}) : \mathcal{R}^3 \to \mathcal{R}$ that solves this Poisson equation is a smooth approximation of the indicator function, while its zero level-set is an approximation of the boundary of the geometric domain represented by the input points. To extract this level-set, the marching cubes technique [55] is employed.

In the original publication of the Poisson reconstruction method, homogeneous Dirichlet boundary conditions were specified at all boundaries of the computational domain, with an extension to Neumann boundary conditions presented in [11].

The right hand side of Equation 4.5 requires a smooth vector field in order to apply the divergence operator. Therefore, the discrete normal vectors need to be extended to a smooth representation. To this end, the following procedure is applied:

1. Starting from the initial point cloud S, a quadtree (in 3D, octree) $\mathcal{O}$ with maximum depth D is generated, such that every point sample falls into a leaf node at depth D (Figure 4.5a).
2. To every node $o \in \mathcal{O}$ in the tree, a basis function $F_o(\boldsymbol{x})$ is assigned, centered at o and scaled by the size of o:

   $$F_o(\boldsymbol{x}) = F\left(\frac{\boldsymbol{x} - \mathbf{c}_o}{w_o}\right), \tag{4.6}$$

   where $\boldsymbol{c}_o$ and w_o denote the center and the width of the octree node, respectively. In [10], the function F of Equation 4.6 is a unit integral, bivariate (in 3D, trivariate) quadratic B-Spline (Figure 4.5b).
3. The vector field $\mathbf{n}(\mathbf{x})$ is created by taking a linear combination of the nodal bases:

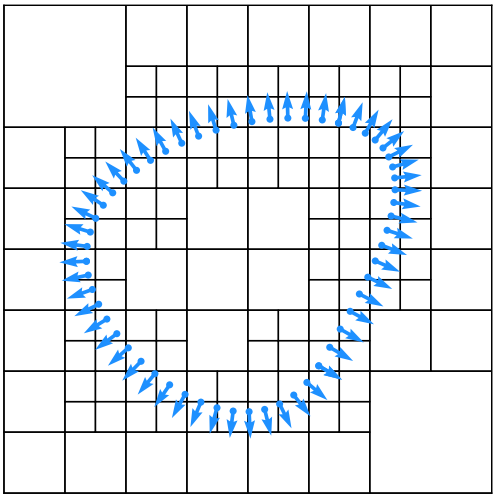
   $$\boldsymbol{n}(\boldsymbol{x}) = \sum_{o \in \mathcal{O}^D} F_o(\boldsymbol{x})\mathbf{n}_o, \tag{4.7}$$

   where $\mathcal{O}^D$ are all the quadtree (in 3D, octree) nodes at depth D, and $\mathbf{n}_o$ is defined as:

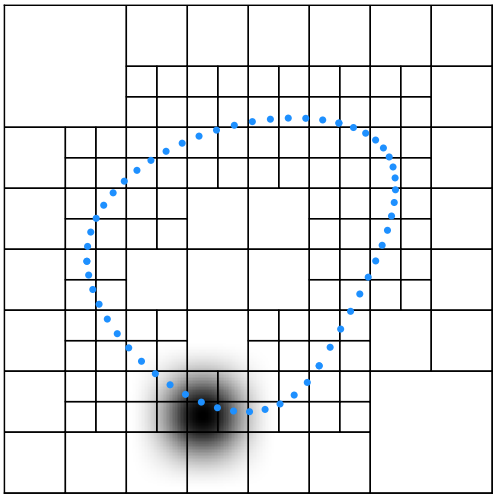
   $$\boldsymbol{n}_o = \sum_{(\mathbf{p}_i, \mathbf{n}_i) \in S} F_o(\boldsymbol{p}_i)\mathbf{n}_i, \tag{4.8}$$

   which is necessary to avoid clamping the samples to the centers of the octree nodes (Figure 4.5c).

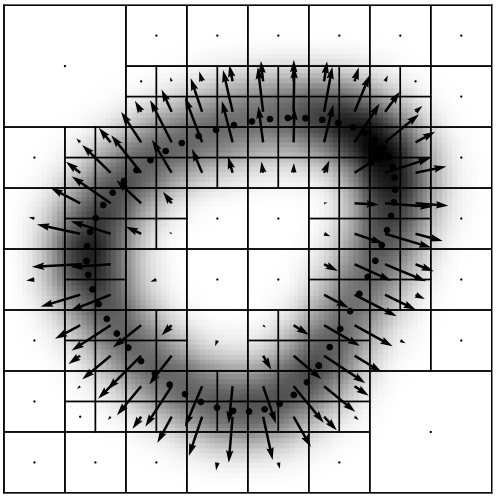
4. Having obtained the smooth vector field $\boldsymbol{n}_i$, its divergence is computed and used as the right hand side in Equation 4.5 (Figure 4.5d).



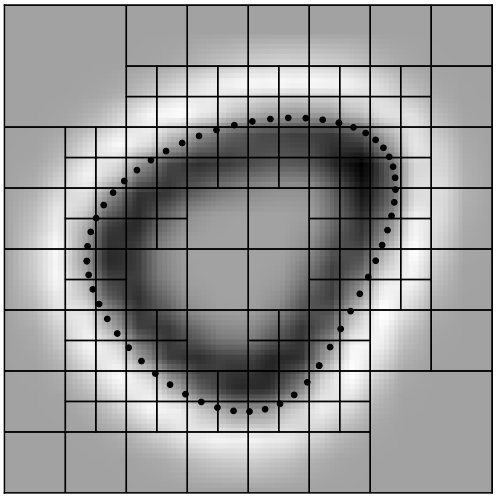
(a) Input point cloud with normal vectors and a quadtree $\mathcal{O}$ of depth 4.



(b) Single cubic B-Spline basis assigned to one quadtree node $o \in \mathcal{O}$.



(c) Smooth vector field $\boldsymbol{n}$.



(d) Divergence field $\nabla \cdot \boldsymbol{n}$.

Figure 4.5: *Poisson surface reconstruction following [10]: the process of generating a smooth divergence field from the discrete sample data*

Having formulated $\nabla \cdot \boldsymbol{n}$, the Poisson problem of Equation 4.5 can be solved by means of the Finite Element Method. In this setting, various choices of finite element basis functions are available to discretize the indicator function Φ. As an example, Figures 4.6a-4.6c depict Φ computed on the point clouds of Figure 4.4, using integrated Legendre polynomials of order $p = 4$.

Another choice for a basis can be found in the original publication of the Poisson surface reconstruction, where the same nodal B-Spline basis functions are used for discretizing the solution as for manufacturing the smooth vector field. The multiresolution structure of this

basis allows for applying a multigrid-like solution strategy to obtain an efficient solution of the resulting linear system of equations [10].

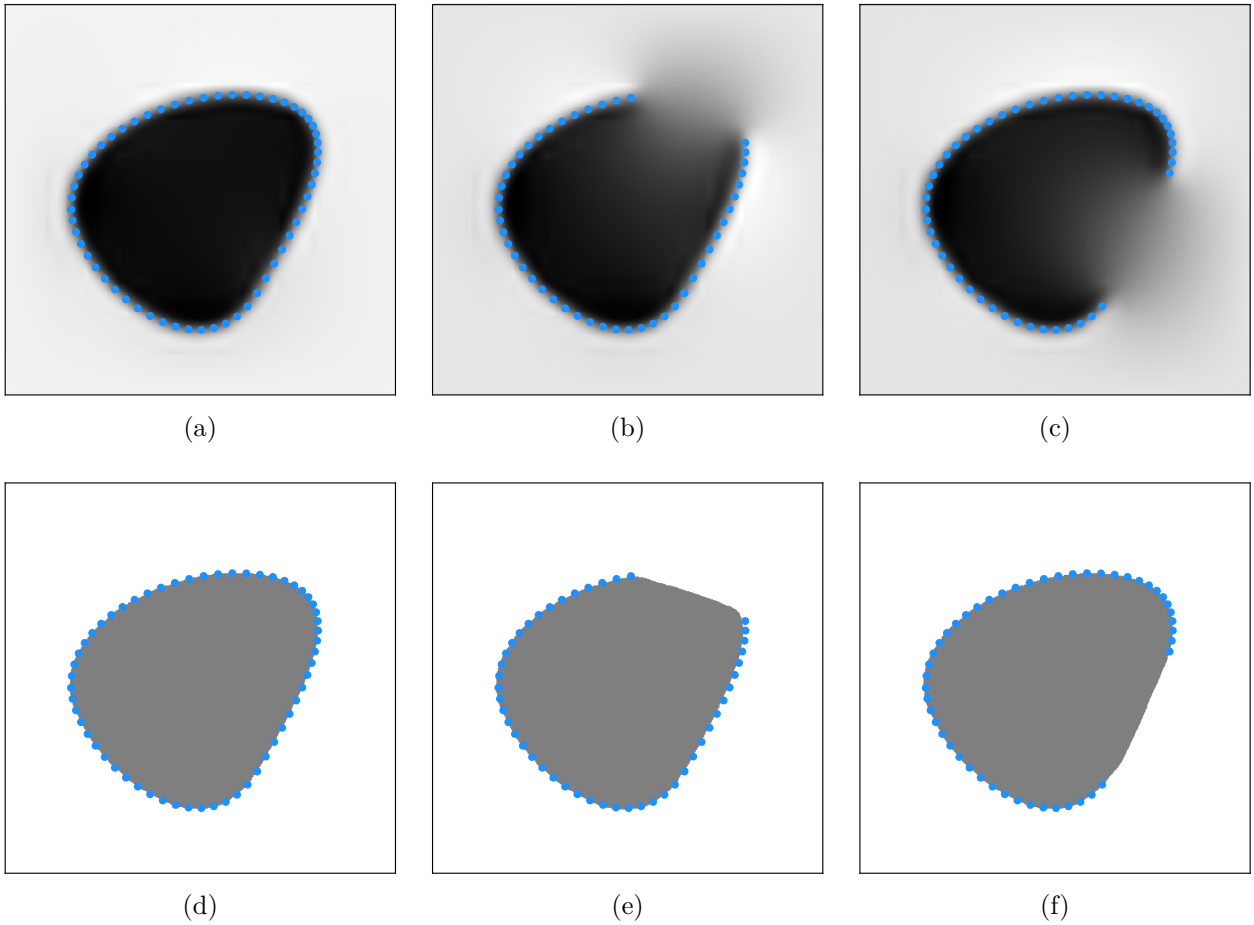


Figure 4.6: *Poisson surface reconstruction: (a)-(c) recovered indicator field* Φ*, (d)-(f) the corresponding field of inside-outside state computed on the point clouds of Figure 4.4.*

The iso-value for recovering the level-set surface of the indicator function can be found by taking the average value of Φ evaluated at the sample positions:

$$\partial\mathcal{M} = \{x \in \mathbb{R}^3 | \Phi(\boldsymbol{x}) = \Phi_c\}, \tag{4.9}$$

with

$$\Phi_c = \frac{1}{|S|} \sum_{\boldsymbol{p} \in S} \Phi(\boldsymbol{p}). \tag{4.10}$$

To convert $\partial\mathcal{M}$ to a triangulation, the marching squares (in 3D, marching cubes) algorithm can be used, which usually saves the resulting surface triangulation in an STL file. While this final step of geometry recovery is necessary if the geometric model is to be further processed (e.g. by a finite element mesh generator), it is not needed in the context of the finite cell method. The key insight here is that in order to determine whether a point lies inside or

outside the geometric model, it suffices to evaluate Φ and compare its value to the level set value Φ_c of Equation 4.10. Therefore, the scaling factor $\alpha(\boldsymbol{x})$ of Equation 2.46 can be determined as follows:

$$\alpha(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \Phi(\boldsymbol{x}) \leq \Phi_c, \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Computing the inside-outside state this way allows for conducting finite cell computations without recovering a surface triangulation of the geometric model. This method possesses the following three advantages:

- The inherent problems of data exchange between different implementations can be avoided.
- Querying the scalar field is a much faster operation than computing inside-outside tests on surface triangulations.
- The problems associated to point membership tests on non-watertight triangulations can be avoided.

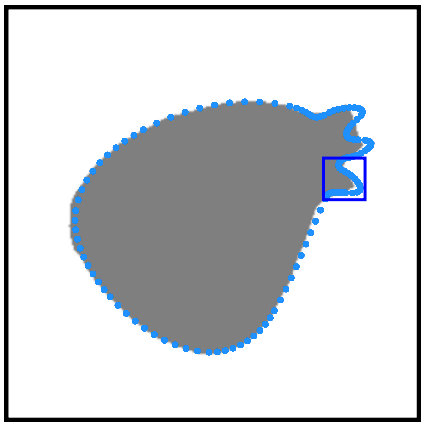
As observed in Figures 4.6d-4.6f, the Poisson method reconstructs an indicator function that does not show the problematic "flow-out" effects seen in Figure 4.4.

At this point, the impression is that using Poisson surface reconstruction is always preferable to the nearest-neighbor based inside outside tests of Section 4.2. While the Poisson approach is one of the most popular methods of choice in computer graphics, where the main focus is on recovering visually pleasing surface models, one needs to pay attention to other aspects when performing structural analysis with the finite element method and its derivatives, such as the finite cell method. In this context, it is important to have an accurate representation of the geometry in those regions where rapid variations in the stress field are expected. On the other hand, to avoid high analysis costs, it is very often desirable to disregard mechanically irrelevant parts in a finite element model. Indeed, a geometric defeaturing step prior to finite element mesh generation is a common procedure in everyday engineering practice [174, 175].

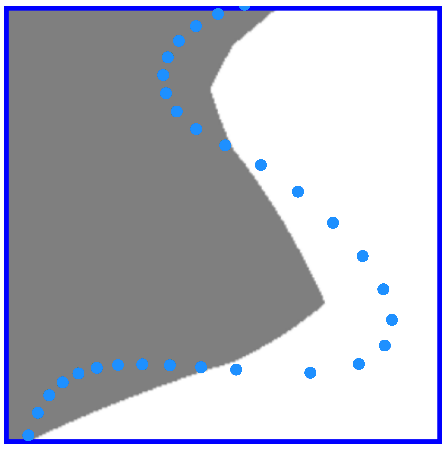
Because the Poisson method is based on a least-squares fit formulation, the resulting interface does not necessarily pass through the given points in the cloud. Instead, certain details may get lost, as shown in Figure 4.7b. To increase the amount of details kept by the method, either a higher octree depth D for the construction of the vector field $\boldsymbol{n}$ can be chosen, or a refinement of the finite element basis that approximates the solution of Equation 4.5 is needed, on the expense of higher computational costs. In contrast, when the nearest neighbor-based point membership-classification is employed, the interface is forced by definition to pass through the given points in the point cloud, retaining even the smallest details, see Figure 4.7c.

To combine the advantageous properties of the Poisson method and the nearest neighbor-based point membership classification, it is feasible for FCM simulations to define the indicator function in a two-step procedure:

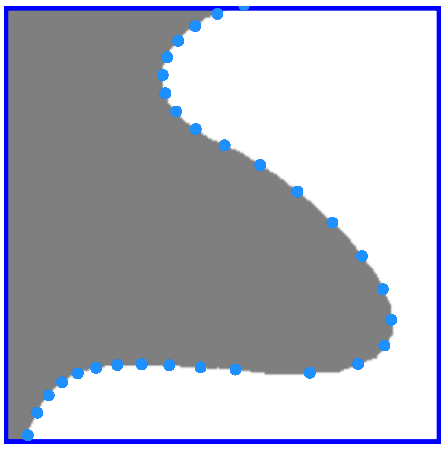
1. *Bulk field recovery.* In this step, the indicator field of the potentially incomplete point cloud is computed using Poisson surface reconstruction. This makes sure that substantial missing parts are closed, providing a watertight indicator field.
2. *Fine detail definition.* At the regions around the fine details the nearest neighbor-based point membership classification is used. This step makes sure that the details that may get lost during the first step are preserved.



(a) Point cloud and indicator function.



(b) Indicator function recovered by the Poisson method, $D = 4$.



(c) Indicator function recovered by the nearest neighbor-based method.

Figure 4.7: *Difference of the indicator functions recovered by the Poisson method and the nearest neighbor-based point membership classification. While the overall shape of the indicator functions look similar, the amount of details recovered by the two methods are different.*

This way, the computational costs associated to the Poisson reconstruction can be kept at a minimum, as the method only needs to be employed to recover the "bulk" part of the indicator field. However, fine details are retained in the cloud due to the application of the nearest-neighbor based inside-outside testing.

In practice, the two-step definition of the indicator function can be implemented by combining the result of the Poisson method and the nearest neighbor-based approach into a CSG tree. To this end, in a pre-processing step, the mechanically interesting parts of the input point cloud S are selected. Let $S_1 \subset S$ be one such part and B_1 its axis aligned bounding box.

The indicator function $\alpha_1(\boldsymbol{x})$ can be computed on S_1 with Algorithm 1. Further, using the Poisson method, $\alpha_P(\boldsymbol{x})$ can be defined on the whole cloud, following Equation 4.11. To

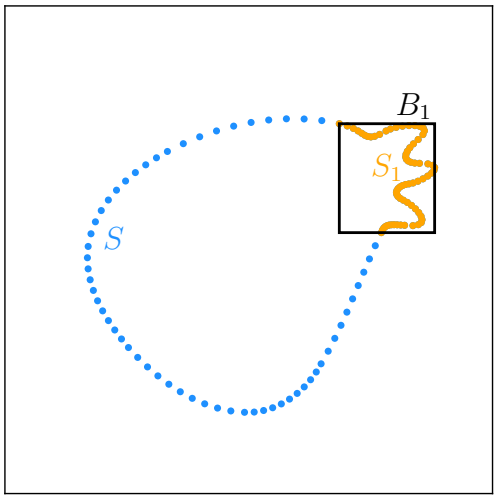


Figure 4.8: *Detail selection for the two-step indicator function recovery procedure. The orange part S_1 is selected from the overall point cloud S, depicted in blue color. In the bounding box B_1 of S_1, the nearest neighbor-based point membership classification procedure is used. In other parts of the domain, the Poisson reconstruction computes the indicator function.*

combine $\alpha_1(\boldsymbol{x})$ and $\alpha_P(\boldsymbol{x})$ together, the following auxiliary domains are defined:

$$\begin{aligned} A_1 &= \alpha_1 \cap B_1 \\ A_2 &= \alpha_P - B_1, \end{aligned} \tag{4.12}$$

where $\cap$ and $-$ denote the Boolean intersection and difference operations, respectively. Finally, the indicator function of the complete domain, including the fine details are defined as:

$$\Omega = A_1 \cup A_2, \tag{4.13}$$

where $\cup$ is the Boolean union operator. The CSG tree defined in this manner is demonstrated in Figure 4.9.

4.5 Neumann boundary conditions[d]

To apply boundary conditions in the weak sense, the contour integral in Equation 2.47 needs to be evaluated. For surface models, this is a relatively easy procedure, as they usually possess or can be converted into tessellations. Then, the integral over Γ_N is computed as the sum of the integrals over the individual simplices in the tessellation.

However, for point-based geometries, no such tessellations exist. Although there are methods that are able to recover a triangulation from point cloud descriptions, their application would require to perform the same steps as the standard steps of the measurement-to-analysis pipeline in Figure 1.1.

An alternative formulation which allows for applying boundary conditions directly on point cloud-based surface representations is needed. One possible solution to this challenge is to convert the contour integral into a domain integral by using the sifting property of the Dirac

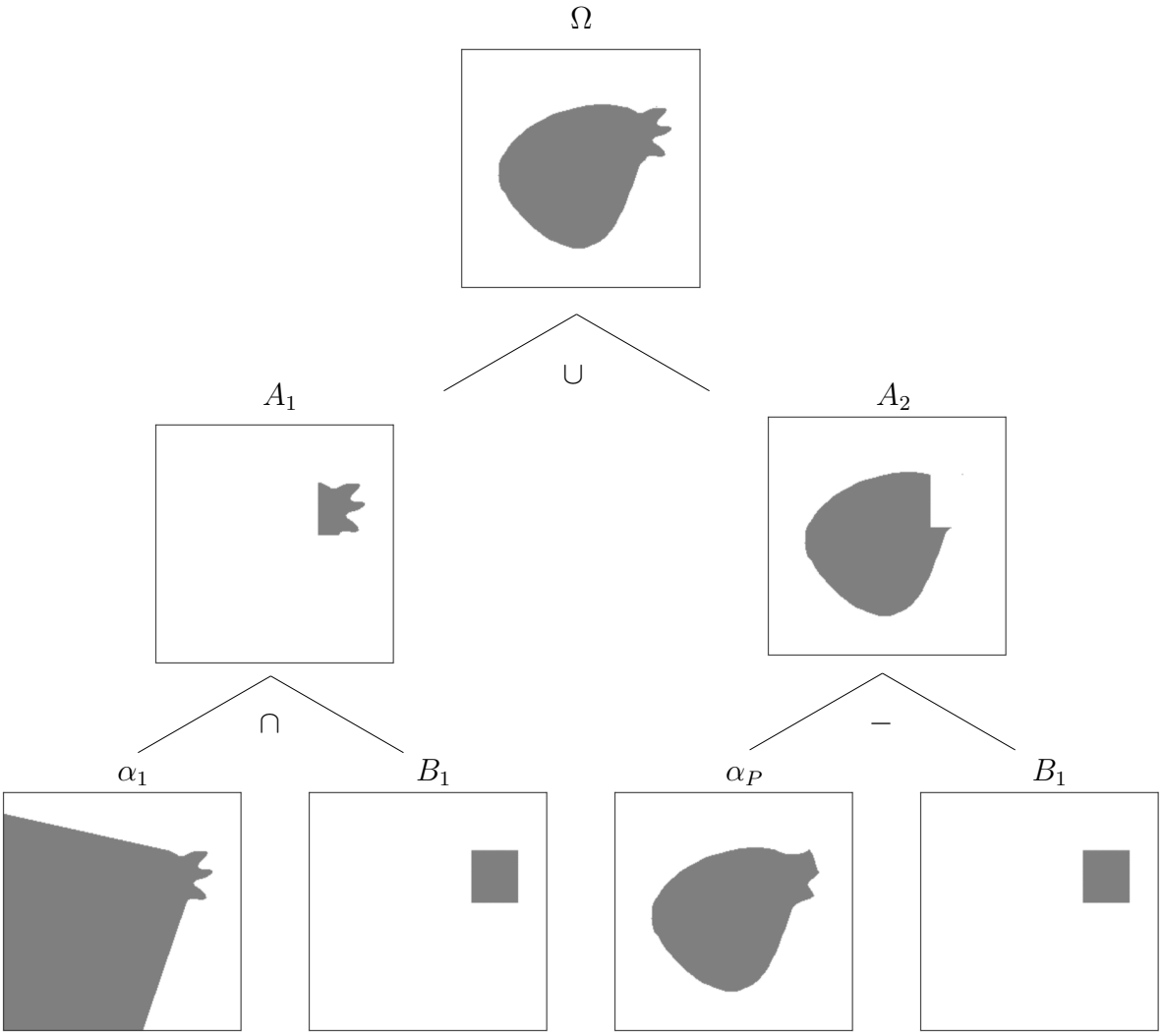


Figure 4.9: *CSG tree built by combining the indicator functions defined by the Poisson method and the nearest neighbor-based algorithm, on the cloud S of Figure 4.8. The bulk part of the domain is recovered by the former, while the fine details by the latter.*

delta distribution [176]:

$$\int_{\Gamma} f(\boldsymbol{x}) d\Gamma = \int_{\Omega} f(\boldsymbol{x}) \delta(\boldsymbol{x}) d\Omega, \tag{4.14}$$

with

$$\delta(\boldsymbol{x}) = \begin{cases} \infty & \forall \boldsymbol{x} \in \Gamma \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

In numerical applications, the regularized variant of the Dirac delta distribution is employed. There are different choices available for the regularization, see e.g. [177, 178]. In our examples,

we employ the following 1D formulation:

$$\delta(x) \approx \delta_\epsilon(x) = \begin{cases} \frac{1}{2\epsilon}\left(1 + \cos\left(\frac{\pi x}{\epsilon}\right)\right) & \text{if } |x| \leq \epsilon, \\ 0 & \text{otherwise,} \end{cases} \tag{4.16}$$

where ϵ is a length scale parameter controlling the width of the regularization. Figure 4.10 depicts δ_ϵ for different choices of ϵ.

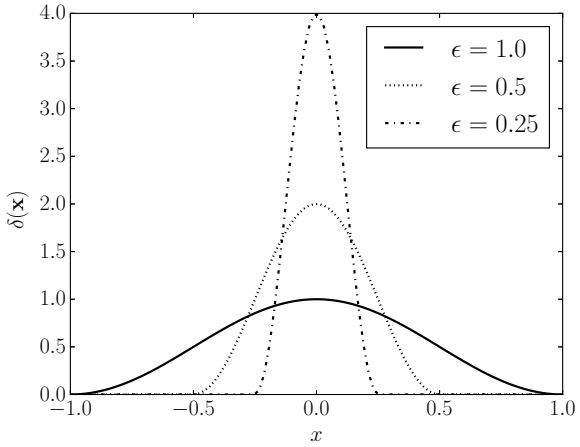


Figure 4.10: *Regularized Dirac delta functions for different length scales [4].*

To extend the delta function to more dimensions, the distance function $d_\Gamma(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is needed, that, for a given $\boldsymbol{x} \in \mathbb{R}^n$ returns the distance of $\boldsymbol{x}$ to the contour Γ. Then, the multi-dimensional regularized delta function, associated to the contour Γ can be written as:

$$\delta_\Gamma(\boldsymbol{x}) = \delta_\epsilon(d_\Gamma(\boldsymbol{x})). \tag{4.17}$$

In the point-cloud setting, Γ is represented by the point set $S_\Gamma \subset S$. Instead of computing the distance to the closest point $\boldsymbol{p}_i$, we compute the planar approximation of the n-neighborhood of $\boldsymbol{p}_i$ by principal component analysis and evaluate the distance toward this approximant, as depicted in Figure 4.11.

Finally, the contour integral for the Neumann boundary condition in Equation 2.47 is formulated as:

$$\int_{\Gamma_N} \mathbf{v} \cdot \tilde{\mathbf{t}} \mathrm{d}A \approx \int_{\Omega_\cup} \delta_\epsilon\left(d_{S_\Gamma}(\boldsymbol{x})\right)\left(\mathbf{v} \cdot \tilde{\mathbf{t}}\right) \mathrm{d}\Omega \tag{4.18}$$

It is noted here that the transformation of boundary condition integrals into a domain integral using the delta function is an already existing concept in the context of FCM. Similar to point-cloud descriptions, phase-field models do not possess tessellations as well. In this case, the approach outlined in this section can be applied to compute Neumann boundary conditions as well as Dirichlet boundary conditions formulated in the weak sense. For more details, refer to [179, 180].

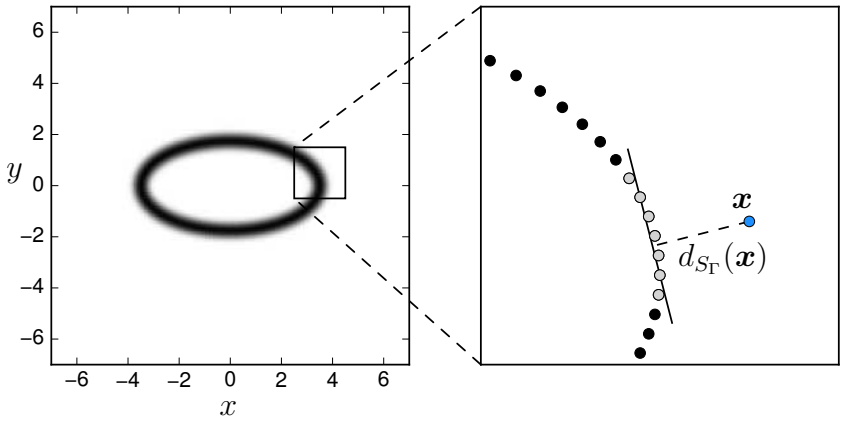


Figure 4.11: *Computing the distance $d_{S_\Gamma}(\boldsymbol{x})$ towards the point set S_Γ. For a given query point $\boldsymbol{x}$, the nearest point $\boldsymbol{p}^*$ and its n-neighborhood $\{\boldsymbol{p}_i\}, i = 1...n$ is found, depicted as gray dots. Here, $n = 6$. Then, the distance towards the planar approximant on $\{\boldsymbol{p}_i\}$ is computed. The resulting delta field is shown on the left side [4].*

4.6 Examples[d]

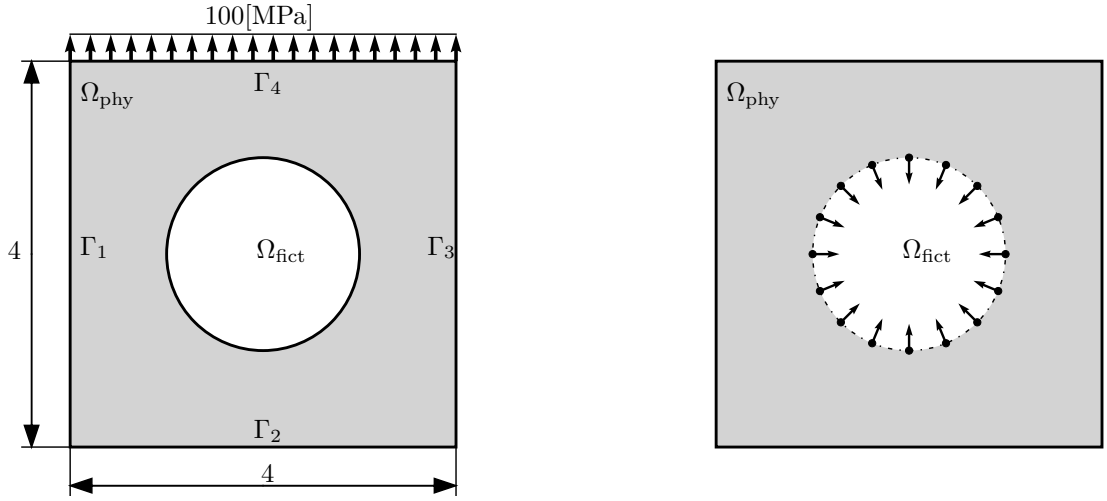
This section demonstrates the proposed point cloud-based FCM approach with numerical examples in two and three dimensions. First, we study an example with a known reference solution, where all the boundary conditions are aligned with the finite cell boundaries. This way, the modeling errors due to the approximate application of Neumann BC-s (Equation 4.18) can be ruled out. Thereafter, an example where the performance of applying Neumann boundary conditions on non-conforming interfaces (as explained in Section 4.5) is tested. Finally, the point cloud-based FCM method is demonstrated on real-life, three dimensional examples of historical structures.

4.6.1 2D studies

4.6.1.1 Perforated plate with circular hole

Figure 4.12a depicts a rectangular plate with a circular hole in the center. The plate is subjected to a constant traction along Γ_4, while symmetry boundary conditions are applied on Γ_1 and Γ_2. The Young's modulus and the Poisson's ratio of the material are $E = 2.069 \cdot 10^5$[MPa] and $\nu = 0.29$, respectively. Considering plane stress physics, the reference strain energy obtained by an overkill FEM analysis is $U_{\text{ref}} = 0.7021812127$ [29]. For the FCM, the embedding domain is discretized into 2×2 elements, and a value of $\alpha = 10^{-12}$ is applied to scale the material parameters in the fictitious domain. The polynomial order of the shape functions is $p = 12$. The continuous curve representing the circular hole is replaced by an oriented point cloud consisting of n points, where n is gradually increased in the range of $n = 4...4096$. Refer to Figure 4.12b for an example of such cloud. The accuracy of the analysis is measured using the following error norm:

$$e = \frac{|U_{\text{ref}} - U_{\text{num}}|}{U_{\text{ref}}}, \tag{4.19}$$



(a) Geometry and boundary conditions.

(b) The inner boundary replaced by an oriented point cloud of 16 points.

Figure 4.12: *Rectangular plate with a circular hole [4].*

where U_{num} is the strain energy computed on the discrete point cloud representation of the circular hole.

The discretization of the circular interface into an oriented point cloud can be thought of as a replacement of the boundary by an n-sided polygon. This introduces a modeling error when integrating the term over Ω_{phy} in Equation 2.47, as the integration is not performed over an exact circle but rather over its polygonal approximation. Therefore, the error is expected to converge at the same rate as the area of an n-sided polygon converges towards the area of its inscribed circle, i.e. quadratically.

This expectation is confirmed by the error plots on Figure 4.13. The figure depicts the evolution of the error for increasing cloud densities, for different maximum levels of quadtree subdivision $k = \{4, 6, 8\}$. Initially, the polygonal approximation dominates the error and the curves show quadratic convergence. Eventually, depending on the value of k, the convergence curves level off into a plateau. In the plateau region, the error due to the quadtree-based integration overtakes the polygonal discretization error: even though the interface is modeled by higher resolutions, the integration scheme is not able to resolve this increase in geometric accuracy.

4.6.1.2 Perforated plate with elliptical hole under internal pressure

In this example, the circular interface from the previous section is replaced by an elliptical curve (Figure 4.14). To investigate the effects of applying Neumann boundary conditions as described in Section 4.5, the elliptical hole is discretized into an oriented point cloud along which a constant internal pressure of 1[MPa] is applied. The reference value for the strain energy computed by p-FEM is $U = 44.28375067893$. The domain is discretized into a regular

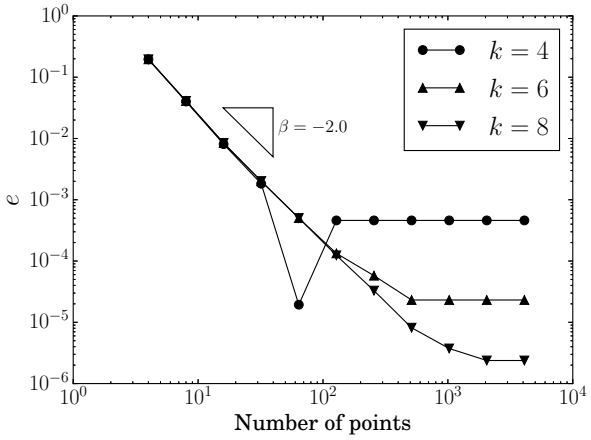


Figure 4.13: *Rectangular plate with circular hole: convergence of error for increasing cloud densities and different maximum levels of quadtree subdivision k [4].*

grid of 6×6 finite cells, where the polynomial order of the shape functions varies in the range $p = 1...10$. The regularization parameter of the Dirac delta function in Equation 4.16 is chosen as $\epsilon = 0.0625$. The diffuse region obtained this way (see Figure 4.11) is integrated using composed Gaussian quadrature of order 10 combined with a quadtree-based subdivision of maximum level $k = 8$. While this integration depth seems to be excessively large for practical applications, it should be noted here that in the usual case the size of the surfaces where weak boundary conditions need to be applied is significantly smaller than the overall sizes of the geometries of interest. Therefore, applying the boundary conditions in this manner does not lead to a significant performance penalty.

To rule out the errors associated to the discontinuity in the indicator function $\alpha(\boldsymbol{x})$, the stiffness matrix is integrated using the exact integration technique described in [16]. This way, the error due to the approximate application of the Neumann boundary condition is examined exclusively.

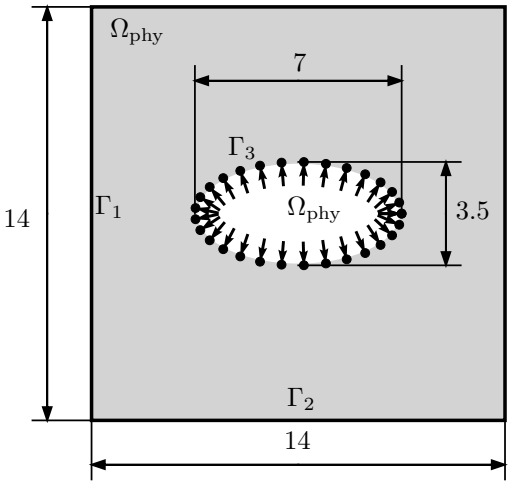


Figure 4.14: *Rectangular plate with elliptical hole under internal pressure [4].*

Figure 4.15 depicts the convergence of the error in the energy norm, for different point cloud

densities. As the figure shows, the expected exponential rate of convergence can be attained in the pre-asymptotic range. However, similar to the study conducted in Section 4.6.1.1, the convergence curves level off into a plateau, depending on the density of the point cloud. As higher cloud densities are able to represent the underlying elliptical contour with higher accuracy, the level-off location shifts towards lower errors for increasing number of points in the cloud. It is noted here, however, that even a relatively low density (1000 points) is able to produce an error in the range of 1%, which is sufficient for most engineering applications.

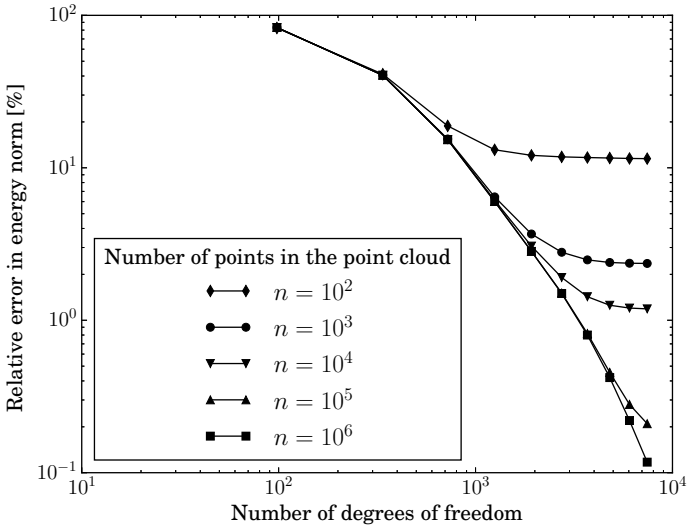


Figure 4.15: *Convergence of the error in energy norm when boundary conditions are applied using the regularized delta function. The polynomial order of shape functions is varied in the range* $p = 1..10$. *The regularization parameter of the delta function is* $\epsilon = 0.0625$ *[4].*

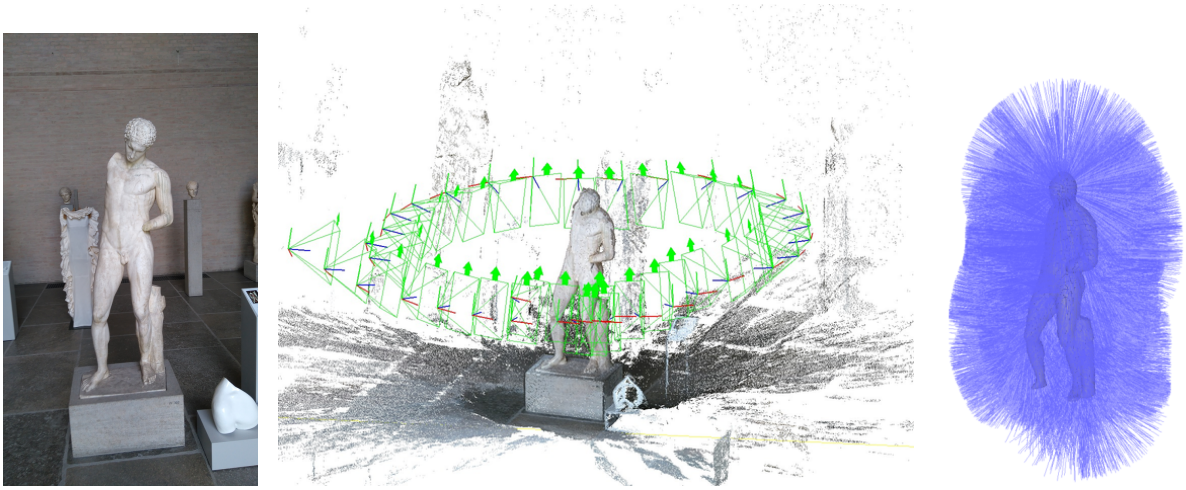
4.6.2 3D examples

In the following, the proposed point cloud-based FCM approach is demonstrated on three-dimensional structures represented by oriented point clouds.

4.6.2.1 Athlete

Figure 4.16a shows a statue from the museum *Glyptothek* located in Munich, Germany. The object was recorded using a cell phone camera from 36 different views. These input images were processed by the popular structure-from-motion toolbox VisualSFM [181], and the multi-view reconstruction algorithm of [103], as demonstrated in Figure 4.16b. The resulting point cloud is depicted in Figure 4.16c. The point cloud was embedded in a regular mesh of 325 finite cells with polynomial order $p = 5$, as shown in Figure 4.17a. Linear elastic material behavior is assumed, and the structure is loaded under its self-weight. The scaling factor α for the FCM was chosen as 10^{-6}. Homogeneous Dirichlet boundary conditions were applied on the bottom faces of the finite cell mesh, in order to rigidly fix the statue to the ground.

The resulting stress field is depicted in Figure 4.17b, while a detailed view on a cross-section of the left foot is shown in Figure 4.17c. As seen in the Figure, the peak stress occurs at

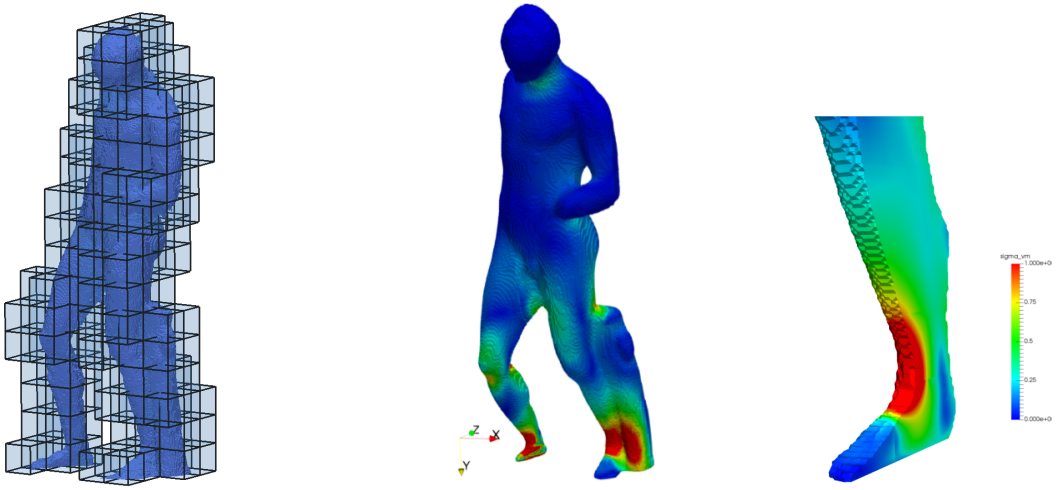


(a) Example input picture

(b) Recovered point cloud and camera positions

(c) Point cloud and the associated normal vectors

Figure 4.16: *Statue example: input pictures and the resulting cloud [4].*



(a) Point cloud embedded into a regular mesh of finite cells

(b) von-Mises stresses throughout the structure

(c) A detailed view on a section of the left foot

Figure 4.17: *Statue example: discretization and stresses [4].*

the "ankle" region, which possesses the smallest cross-section over the entire structure. This phenomenon is in good accordance with other observations from the study of the structural behavior of stone statues: numerical computations conducted on "David" from Michelangelo showed a similar stress distribution, with the peak occurring in the ankle area [6]. Interestingly, other areas of increased stresses also appear such as the area around the neck as well as areas in the upper thigh. Other intuitive candidates for high stresses, such as the left arms and shoulders can be disregarded.

While this example illustrates the main steps of the proposed point cloud based FCM pipeline, without further knowledge about the material parameters, the computed results merely allow for a qualitative assessment of the stress distribution in the statue. The next example addresses this question on a structure where the material properties are known.

4.6.2.2 The cistern of the Hagia Thekla Basilica in Turkey

The archaeological site at Hagia Thekla (Meryemlik) was a major pilgrimage site in late antiquity [182]. There are numerous structures of different types in the site, which can be identified above ground by sight.

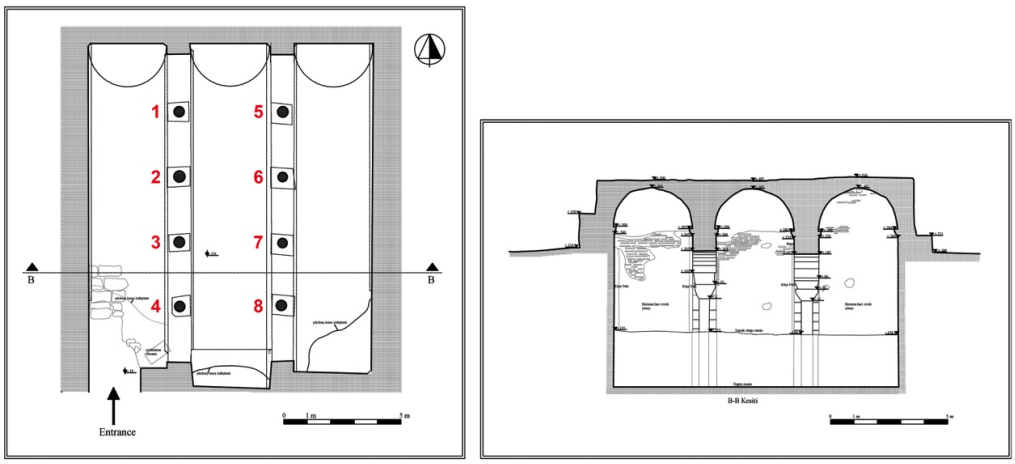
The cistern of the Thekla Basilica is part of the water storage and distribution system of the main church of the site and its sacred area enclosed by walls. It has a rectangular plan measuring approximately 12×14.6 meters in the interior. The interior space is divided into three aisles by two rows of columns (Figures 4.18b and 4.18a[†]). The columns in each row are connected by arches. Three barrel vaults cover the interior running in the north-south direction. The columns supporting the upper structure are made of a pink calcareous stone and originally had a diameter of approximately 45 cm. The columns have double capitals made of limestone. It is not possible to make observations about the condition of the column bases and the floor, due to the thick layer of earth accumulated inside the cistern over centuries. The outer walls are built with a multi-leaf masonry construction system. The outer facing of the walls are made of big limestone blocks, while the inner faces are constructed with brick and mortar. As seen in Figure 4.18a, the cross-sections of the columns have decreased remarkably. The exterior surfaces are flaking due to physicochemical effects; the erosion continues. In addition to surface erosion with a non-uniform pattern, there are deep cavities on the columns. One of the columns (Column 3) has already collapsed and was replaced by a concrete column in the 1960's.

The shape of the decayed column surfaces and cavities are difficult to record using manual measurement procedures. Therefore, the structure was measured using a high definition surveying scanner as demonstrated in [8]. During the field campaign, the instrument was set up at a number of positions around each column at a distance of a few meters. Thus, a maximum point density of approx. 5 mm was ensured to represent the highly decayed columns. More details on the measurement process can be found in [8]. Figure 4.19a depicts the measured point cloud, consisting of 10^7 points.

The most vulnerable elements of the structure are the columns. As stress concentrations are expected at the cavities on the surfaces of the columns, a reduction of the discretization error by a refinement of the computational grid is needed. For reasons of efficiency, it is important to refine the grid only around the columns, where the stress field is expected to

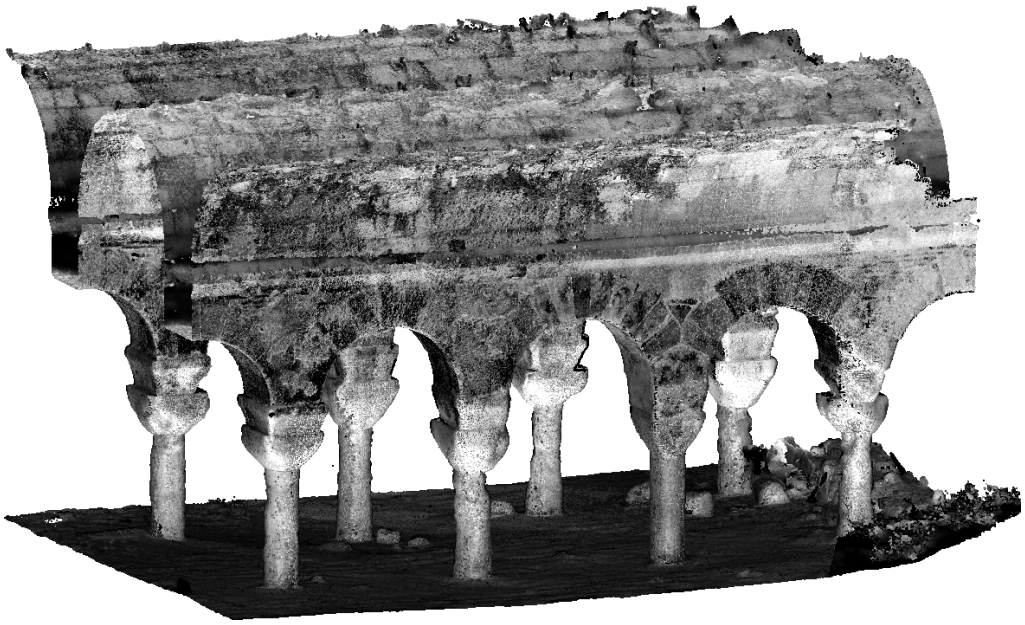
[†]The illustrations of the figure and the point cloud of the temple were provided by Umut Almac, this contribution is greatly appreciated.

(a) Interior view.

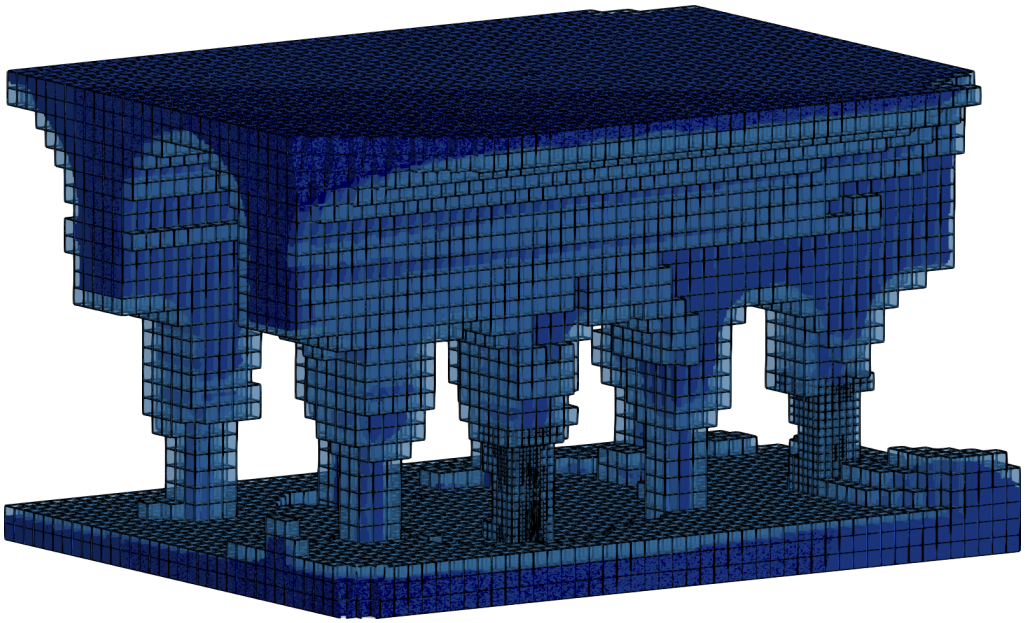


(b) Plan and cross section.

Figure 4.18: *The cistern of Hagia Thekla Basilica [4].*

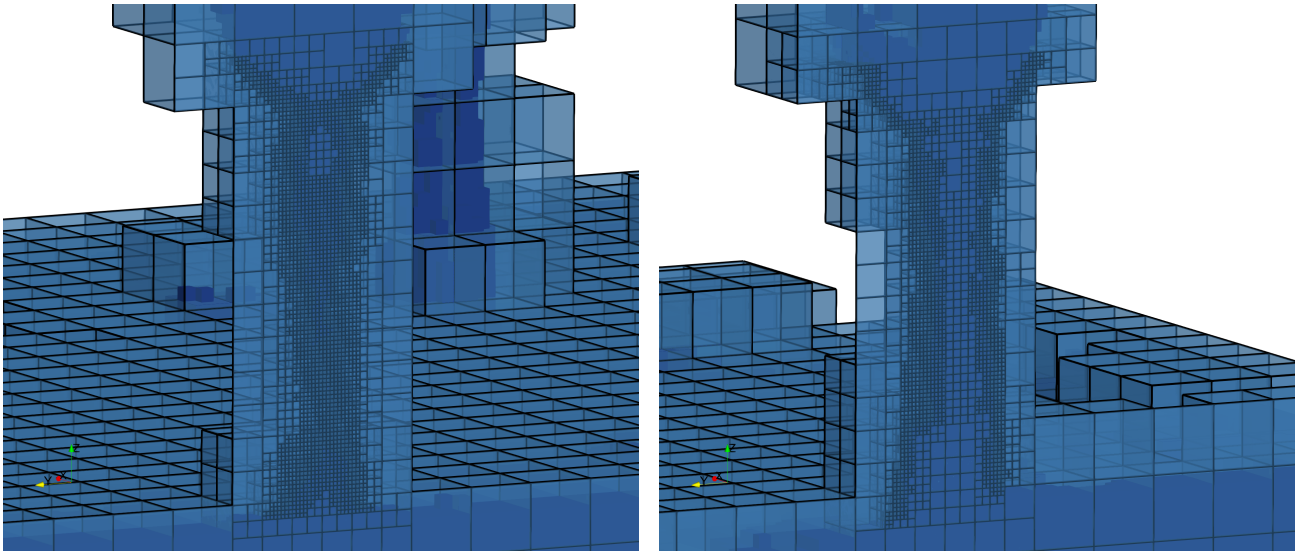


(a) Point cloud



(b) The geometry embedded into a finite cell mesh

Figure 4.19: *Cistern example [4].*



(a) Column 2 (b) Column 4

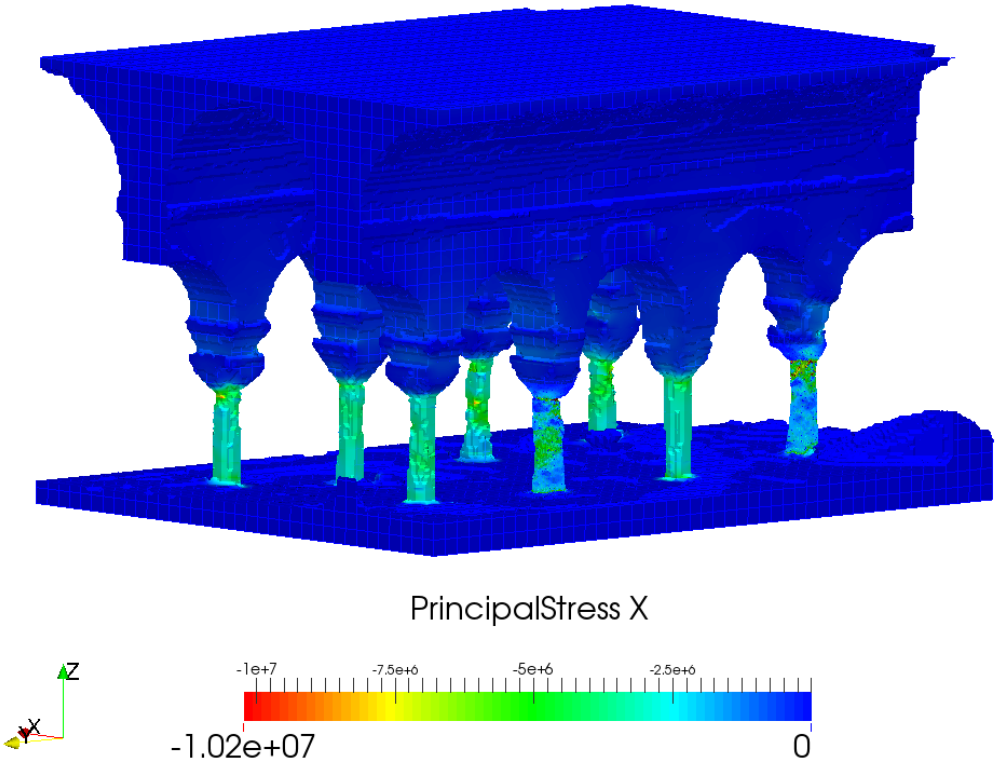
Figure 4.20: *Refined computational grid around the two columns [4].*

change rapidly. For the FEM and the FCM, such *local refinement* techniques have been well-studied recently. In our applications, we employ the *multi-level hp-adaptivity* technique of [38]. In the refinement procedure, those cells that are intersected by the points representing column 2 and 4 are recursively subdivided into eight equal subcells, until a subdivision depth of 4 is reached. A cross sectional view of the refined mesh is depicted in Figure 4.20.

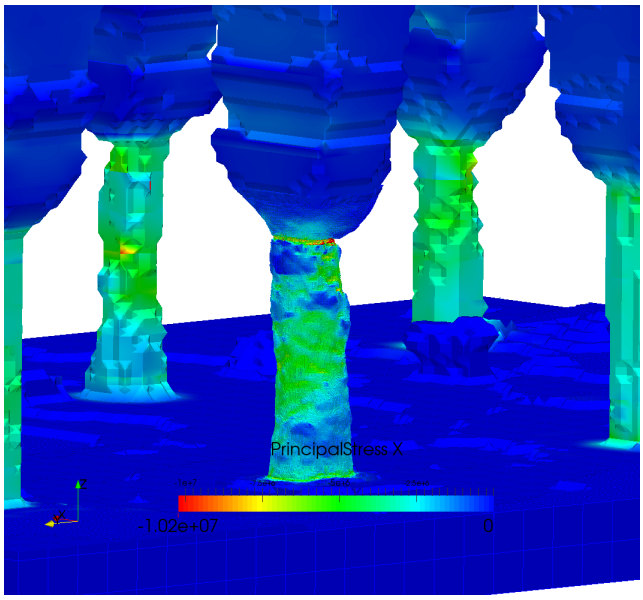
The material properties were defined to be linear elastic and isotropic, with an elastic modulus and a Poisson's ratio of $E = 2 \cdot 10^4$MPa and $\nu = 0.2$, respectively. The specific gravity of the material was set to 27 kN/m^3. These parameters were obtained from measurements conducted on stone specimens collected from the vaults, walls and columns of the structure. In the fictitious domain, the material was given a stiffness of 2 MPa. The foundation of the structure was rigidly fixed to the ground. The maximum principal stress distribution computed by the FCM is depicted in Figure 4.21.

As expected, the highest compressive stresses occur in the columns. The stress values are in the range of 2...6 MPa, while the peak value occurs at the connection between the columns and their capitals. This is in good agreement with the values computed in [8], following the traditional measurement-to-analysis procedure. A comparison of principal stresses along column 2 is given on Figure 4.22.

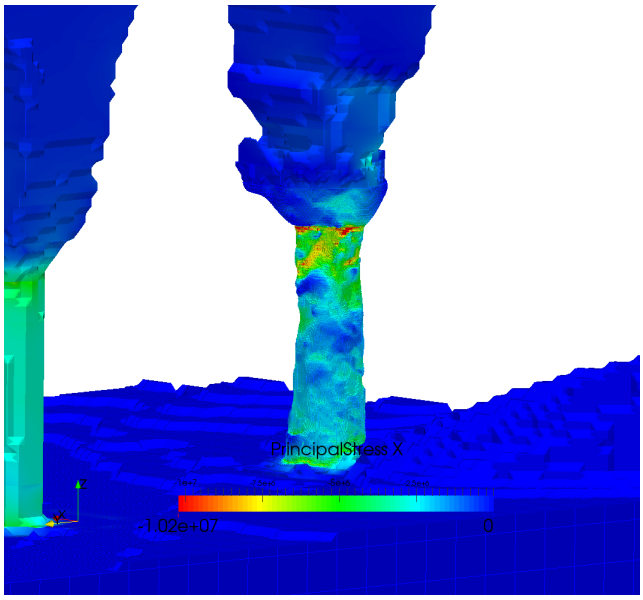
Because the only requirement toward the geometric model is that it needs to be able to provide point-membership information, the finite cell method opens up a convenient way to investigate the effects of geometric changes, such as the removal of a specific column. Within the FCM, it is easily possible to integrate different geometric models following the idea of Constructive Solid Geometry (CSG) modeling [43]. In CSG, 3D objects can be created by combining geometric primitives into a tree structure, using boolean operations. To determine if a point lies within the model or not, the tree structure is traversed from the root towards the leaves, combining the inside-outside state according to the boolean operations at each level. Following this idea, we investigate the effect of removing Column 3 from Ω_{phy}: using



(a) Complete structure



(b) Principal stresses in Column 2



(c) Principal stresses in Column 4

Figure 4.21: *Cistern example: principal stress distribution [4].*

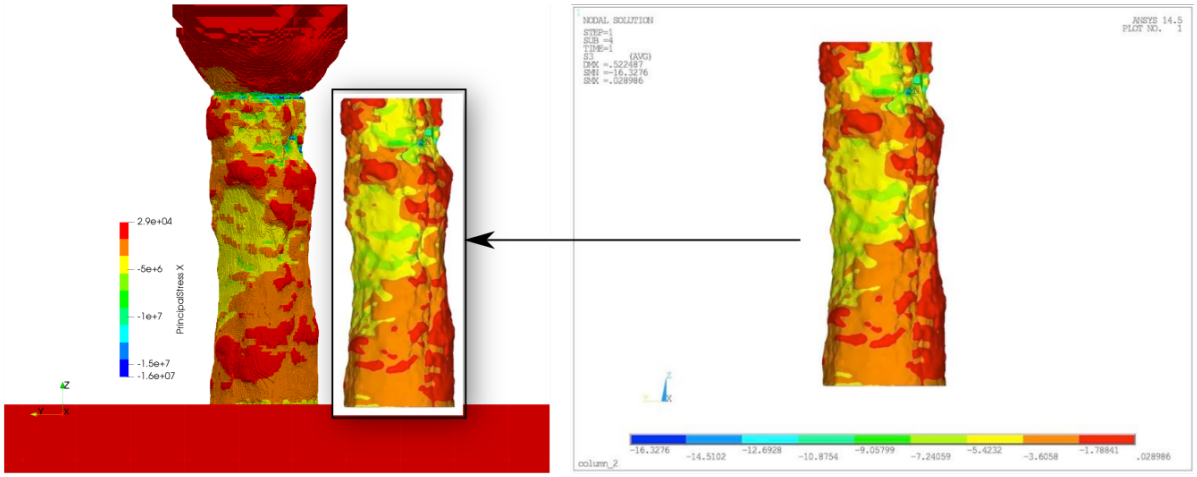


Figure 4.22: *Cistern example: comparison of the maximum principal stresses computed by the FCM (left) and a commercial FEM software (right). The picture on the right is taken from [8] [4].*

boolean difference, the bounding box of the points representing the column is subtracted from the point cloud, as Figure 4.23 depicts.

The principal stress distribution in this scenario is depicted in Figure 4.24. As shown on the figure, the removal of the column causes a redistribution of the loads onto the neighboring supports, leading to an increase in the principal stresses in columns 2 and 4. The redistribution phenomenon can be demonstrated by comparing the principal stress trajectories in the configurations with and without the column, as depicted in Figure 4.25. Due to the removal of the column, a new "arc" forms between the two neighboring columns. This arc is in compression and carries the redistributed load. The newly formed stress state answers why did no structural failure occur when column 3 collapsed: as stone is able to carry substantially higher loads in compression than in tension and the arc of principal stresses is predominantly in compression, the structure was able to endure the collapse.

4.6.2.3 Tower measured by an UAV

Image-based shape measurement algorithms are not limited to pictures stemming from handheld devices. Recent developments in the technology of unmanned air vehicles (UAV) have made it possible to record high-quality pictures of objects of virtually any size for a low cost. For an overview of UAV-based remote sensing methods, refer to [183].

In the context of cultural heritage preservation, aerial images have proven to be especially useful in bridging the disparate scales that are present when scanning archaeological sites [184]. As shown in [185, 186], a carefully conducted acquisition process can provide point clouds that are accurate up to a few centimeters, even if the overall scale of the measured site is in the order of hundreds of meters. The combination of UAV-based remote sensing and traditional terrestrial measurement approaches (e.g. laser scanning) even allows for reconstructing multiresolution models of scales ranging from the geographical level to the level resolving the

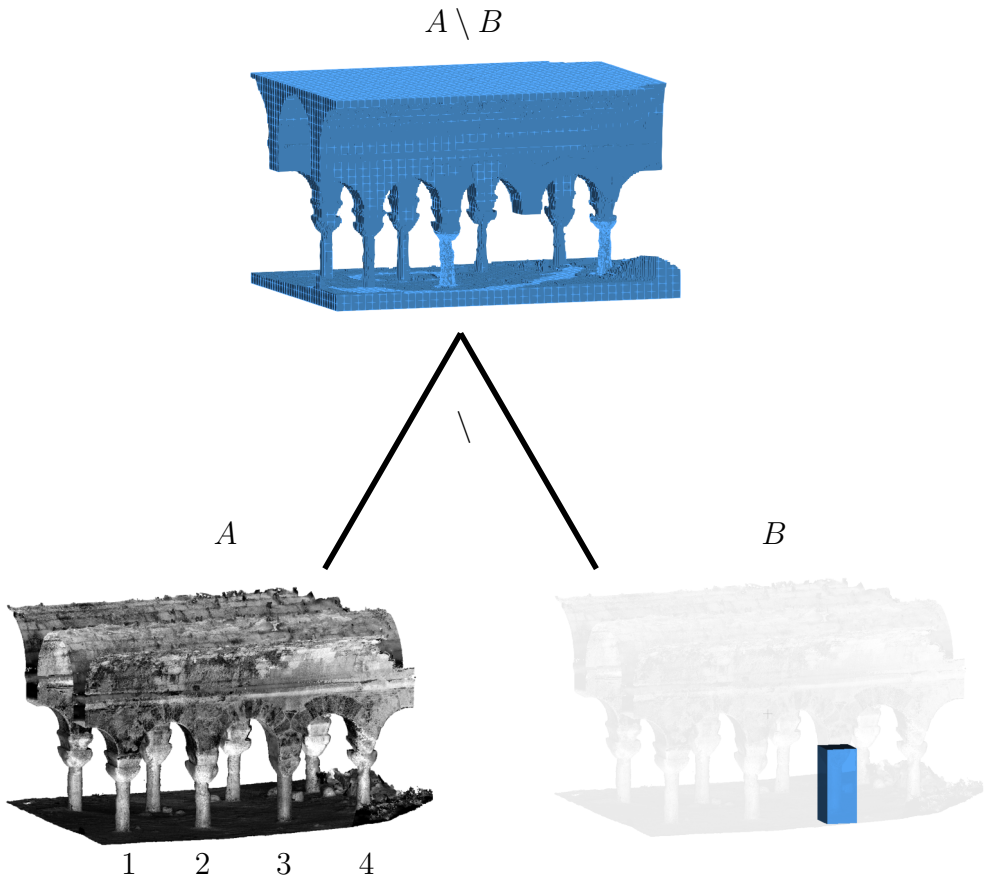
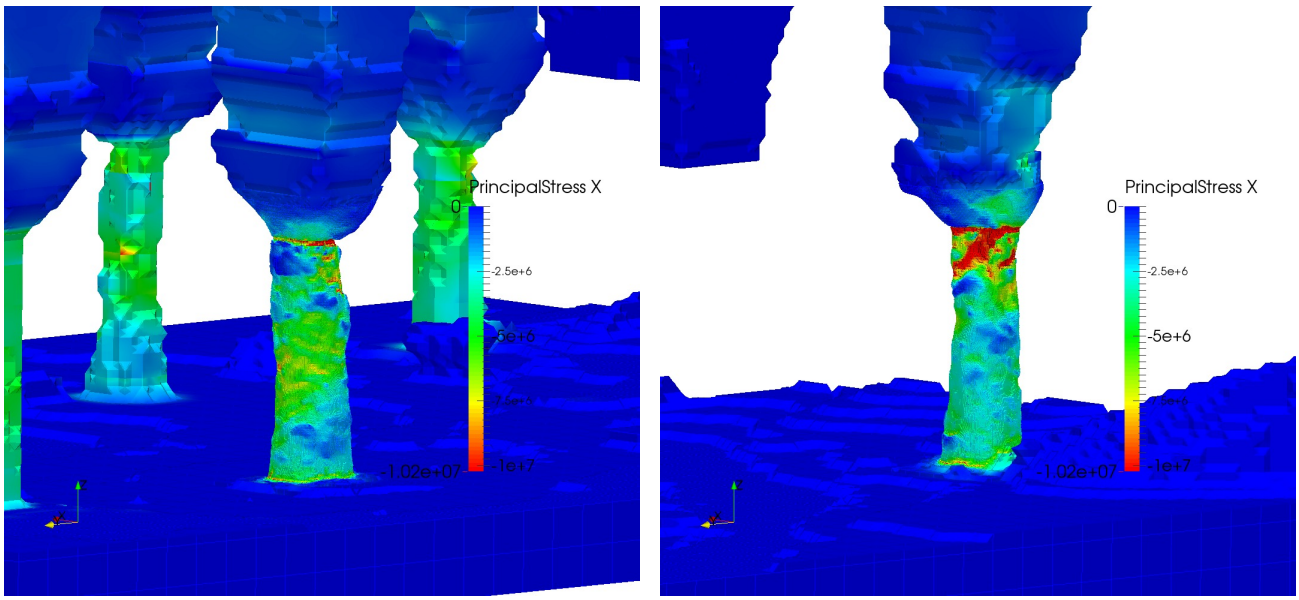


Figure 4.23: *Column 3 removed using a CSG tree with a boolean difference operation. The column numbers are displayed under the point cloud [4].*



(a) Principal stresses in Column 2

(b) Principal stresses in Column 4

Figure 4.24: *Cistern example: principal stress distribution without Column 3 [4].*

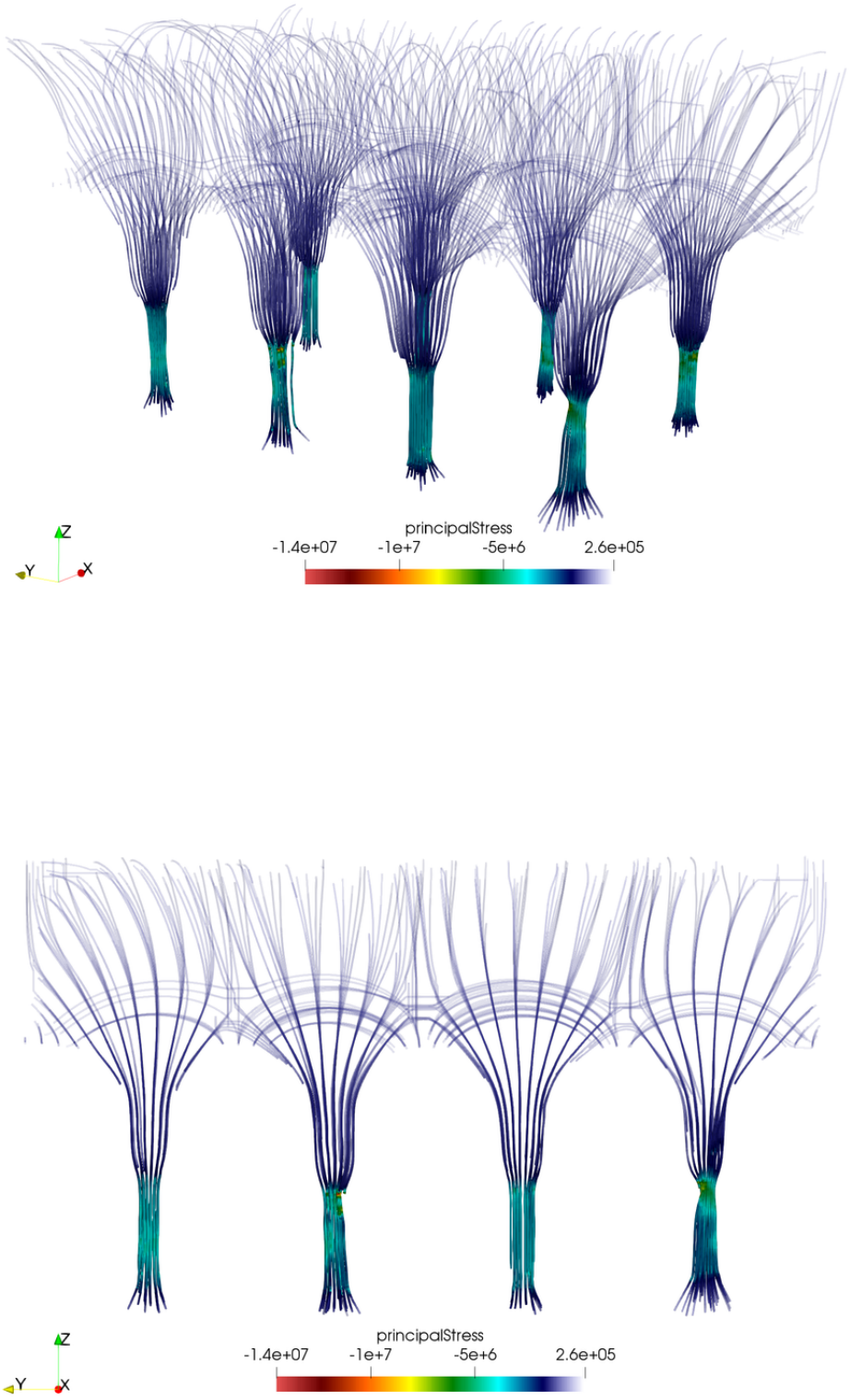


Figure 4.25: *Cistern example: principal stress trajectories on the intact structure [4].*

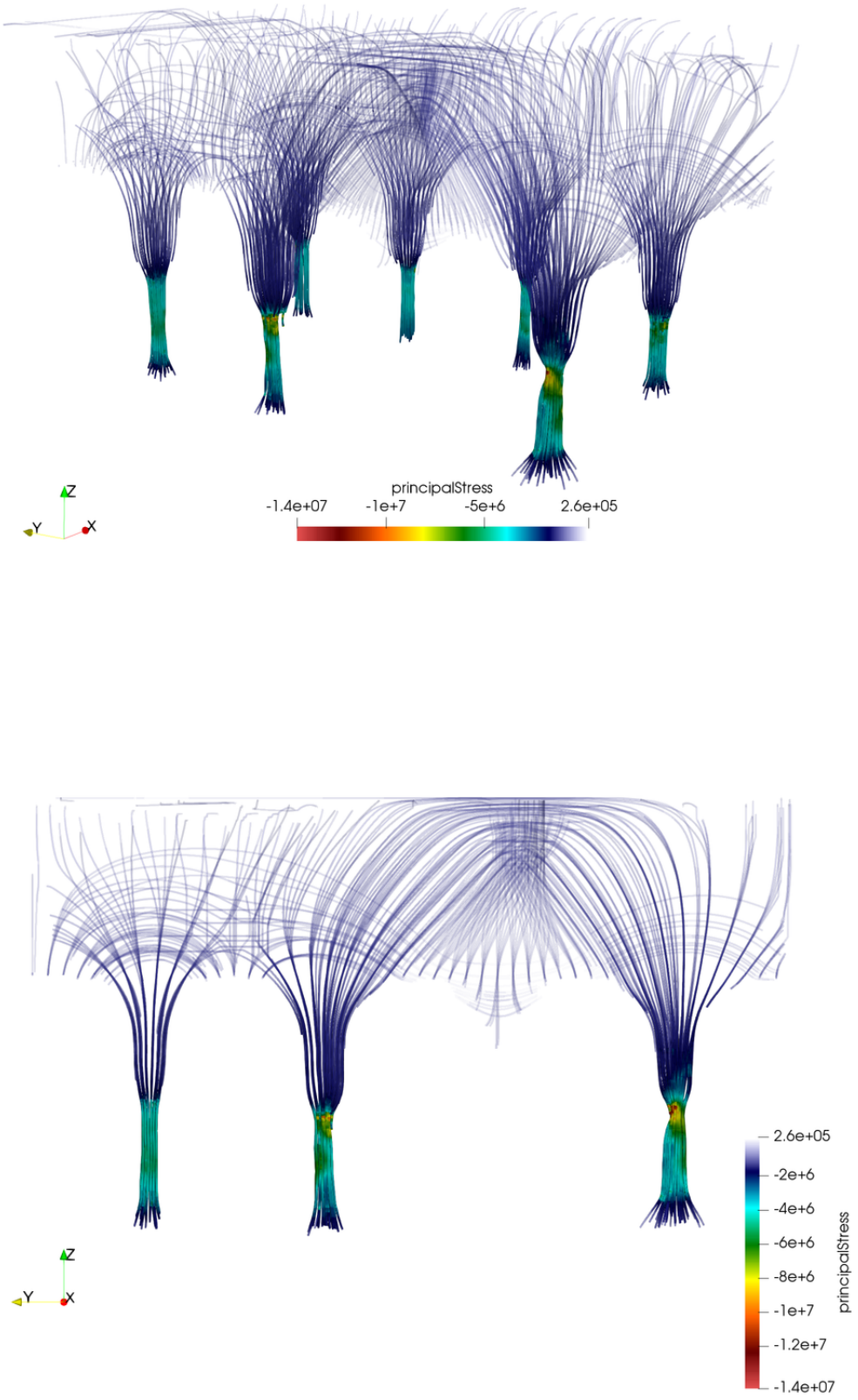


Figure 4.26: *Cistern example: principal stress trajectories on the structure without column 3 [4].*

Figure 4.27: *Hocheppan Castle and the tower located on the north side of the site [4].*

finest architectural details [187].

While the geometric models recovered in this manner are predominantly used for documentation, conservation planning or entertainment such as virtual reality exhibitions (see e.g. [105]), the measured point clouds are also perfectly suited for numerical analysis by the Finite Cell Method. In the following, this idea is demonstrated on the ruin of a medieval tower of the *Hocheppan Castle*, located in South Tyrol, Italy. The castle and its surroundings as well as the tower are depicted in Figure 4.27.

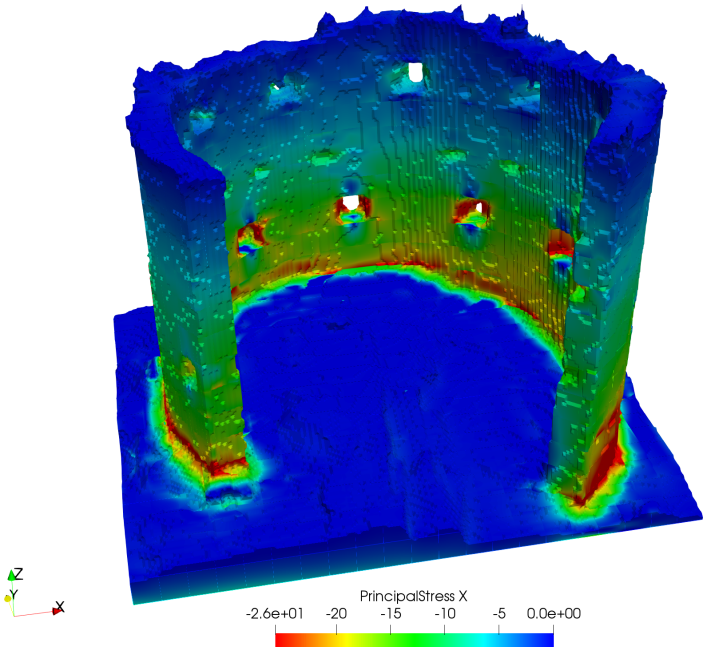
To recover the shape of the tower, a DJI Phantom 4 drone was flown around the structure. During the 30 minutes long flight, 120 pictures were recorded from different heights and angles. As seen on the right side of Figure 4.27, the side of the ruin opposite to the collapsed parts is surrounded by trees. In order to get a better view from these sides as well, 15-20 additional photos were made by a hand-held camera. The recorded images were processed by the open source SfM software COLMAP [188, 189], followed by a multi-view stereo triangulation using Furukawa's PMVS [103] algorithm. The resulting point cloud consisting of 3 million points is depicted in Figure 4.28a.

To compute the stress state that is present in the structure under its self weight, the point cloud was immersed in a regular mesh of $12 \times 12 \times 12$ finite cells of order $p = 3$. The material was assumed to be linear elastic, with an elastic modulus and Poisson's ratio of $E = 2 \cdot 10^{10}\ Pa$ and $\nu = 0.2$, respectively. The scaling parameter in the fictitious domain was chosen as $\alpha = 1 \cdot 10^{-8}$. The foundation of the tower was fixed to the ground rigidly and the structure was loaded under its self-weight, with a specific gravity of $27\frac{kN}{m^3}$. Conducting a finite element analysis for this setup results in the stress state depicted in Figure 4.28b.

Without precise knowledge about the characteristics of the materials used for the construction of the tower, it is only possible to assess the validity of the results qualitatively. As expected, the gravitational loads lead to an overall steady increase in the principal stresses towards the bottom of the structure. The openings along the walls act as "stress concentrators", causing local peaks in the field of principal stresses.

In order to get a better understanding about the structural health of the tower ruin, more detailed analyses could be conducted in the future, addressing several aspects that were not in the scope of this thesis. These points are summarized in the next section.

(a) Point cloud



(b) Principal stresses

Figure 4.28: *Tower ruin at the Hocheppan Castle: point cloud and principal stresses computed by the FCM [4].*

Chapter 5

Summary and outlook

The Finite Element Method is one of the most important tools in everyday engineering practice, used to predict the behavior of physical phenomena governed by partial differential equations. One reason for the high popularity of the method is that it can deal with almost arbitrarily complex geometric models, as long as said models can be meshed, i.e. decomposed into an interconnected network of finite elements. The challenge that one faces very often when conducting a finite element simulation is that many geometric models are not directly suited for mesh generation. There can be various reasons for this difficulty. Firstly, because geometric models form the basis of many downstream applications in product life cycle management, they are not designed primarily with numerical analysis in mind. Therefore, models may contain topological defects, missing parts, or unnecessary details, which are acceptable from the design aspect, but not for numerical analysis. Secondly, product design and numerical analysis are usually not performed within a single software. Instead, different tools are used for individual simulation tasks. Transferring the data between representations of such tools may remove important details of the geometric model or introduce unwanted artefacts. Finally, some of the structures to be analyzed may not possess a CAD representation at all. This is the case for natural shapes (landscapes or rock formations), or objects of cultural heritage preservation, such as statues, old buildings and monuments.

Nevertheless, it is of high importance to assess the structural health of objects without a CAD representation. To construct a finite element model in these cases, the shape of interest needs to be measured by some shape measurement technique. The output of this process is usually a point cloud, a set of discrete points representing the structure's surface. In the standard case, this point cloud data is processed through several stages to derive a finite element model.

This *measurement-to-analysis* pipeline relies on the interplay of various algorithms from computational science and engineering. The problem lying in this dependency is two-fold: the process is hard to automate, and the analyst performing the finite element computation needs to be experienced with many softwares and be aware of the respective pitfalls associated to the individual steps.

This thesis explored and demonstrated the steps and shortcuts that may be taken when performing the steps of the measurement-to-analysis pipeline, in particular the following:

- The step of generating a finite element mesh can be avoided by employing a high-order immersed boundary technique, the Finite Cell Method. As this method computes on

a non-boundary-conforming background mesh, it is of high importance to evaluate accurately the discontinuous integrals that arise in those elements which are cut by the boundary. In Chapter 2, an efficient and accurate method for numerical integration in two- and three-dimensions was presented. It was shown that the method is capable of delivering results with higher accuracy than traditional approaches with less computational effort.

- The usual assumption when gathering the shapes from pictures is that no refraction occurs between the recording device and the object. Chapter 3 presented an approach that includes the effects of refraction in the corresponding bundle adjustment formulation. It was demonstrated how the method can be applied for reconstructing objects immersed in a water container.
- A particularly effective shortcut can be taken when combining point cloud-based geometry descriptions with the Finite Cell Method. As shown in Chapter 2, the only information that the FCM needs from a geometric model is inside-outside state: given a point in space, does this point lie in the physical domain or in the surroundings? Chapter 4 demonstrated that this information can be extracted from oriented point clouds, allowing for direct finite cell analysis of point cloud-based geometric models. This way, the potentially complicated step of geometry recovery and mesh generation can be completely avoided, establishing a seamless connection between shape measurements and structural analysis. The chapter also addressed possible solution strategies to the question of applying boundary conditions on point-based surfaces.

The results and observations of this work open the door for further research in various directions, such as:

- The method of smart octrees was primarily designed for efficient integration of BRep models. There are however other geometric representations where the application of the method could be investigated. One possible direction of further research could be to apply the method on constructive solid geometries, or implicit geometry representations, such as level set functions. The challenging aspect within this application area is that these models do not contain explicit information about sharp edges or vertices, the identification of which is indispensable when the goal is to perform accurate domain decomposition.
- Another direction concerning the method of smart octrees is to investigate its combination with moment fitting. Here, smart octrees could be employed to evaluate the domain integral on the right hand side of the equation for the weights, see Equation 2.49. This way, the smart octree method would ensure that domain integrands are evaluated accurately, while moment fitting would allow for a further reduction of the number of quadrature points.
- The examples in Chapter 4 relied on the assumption that the material of the objects is homogeneous and linear elastic. For historical structures, dealing with non-homogeneous material distributions should be investigated. As reconstruction from pictures or by laser scanning delivers only surface information, this extension would require the coupling of these measurement approaches to methods known from non-destructive testing.
- For more accurate results, cracks, if visible on the structure, should be incorporated in the finite cell model.

- The method presented for dealing with boundary conditions on point-based surfaces was explored only for Neumann boundary conditions. A natural extension of this would be to investigate the application of Dirichlet boundary conditions formulated in the weak sense, e.g. by the penalty method or Nitsche's formulation.

Bibliography

[1] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric analysis: Towards Integration of CAD and FEM*. New York: John Wiley & Sons, 2009.

[2] Y. Zhang, "Challenges and advances in image-based geometric modeling and mesh generation," in *Image-Based Geometric Modeling and Mesh Generation* (Y. J. Zhang, ed.), pp. 1–10, Dordrecht: Springer Netherlands, 2013.

[3] K. Kolev, T. Brox, and D. Cremers, "Fast joint estimation of silhouettes and dense 3D geometry from multiple images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 493–505, 2012.

[4] L. Kudela, S. Kollmannsberger, U. Almac, and E. Rank, "Direct structural analysis of domains defined by point clouds," *Computer Methods in Applied Mechanics and Engineering*, vol. 358, p. 112581, 2020.

[5] I. Kalisperakis, C. Stentoumis, L. Grammatikopoulos, M. E. Dasiou, and I. N. Psycharis, "Precise 3D recording for finite element analysis," in *Proceedings of the 2015 International Congress on Digital Heritage*, vol. 2, pp. 121–124, IEEE, 2015.

[6] A. Borri and A. Grazini, "Diagnostic analysis of the lesions and stability of Michelangelo's David," *Journal of Cultural Heritage*, vol. 7, no. 4, pp. 273–285, 2006.

[7] B. Riveiro, J. Caamaño, P. Arias, and E. Sanz, "Photogrammetric 3D modelling and mechanical analysis of masonry arches: An approach based on a discontinuous model of voussoirs," *Automation in Construction*, vol. 20, no. 4, pp. 380–388, 2011.

[8] U. Almac, I. P. Pekmezci, and M. Ahunbay, "Numerical analysis of historic structural elements using 3D point cloud data," *The Open Construction and Building Technology Journal*, vol. 10, no. 1, 2016.

[9] G. Castellazzi, A. M. D'Altri, G. Bitelli, I. Selvaggi, and A. Lambertini, "From laser scanning to finite element analysis of complex buildings by using a semi-automatic procedure," *Sensors*, vol. 15, no. 8, pp. 18360–18380, 2015.

[10] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, p. 61–70, Eurographics Association, 2006.

[11] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 29, 2013.

[12] F. Calakli and G. Taubin, "SSD: Smooth signed distance surface reconstruction," in *Computer Graphics Forum*, vol. 30, pp. 1993–2002, Wiley Online Library, 2011.

[13] M. W. Beall, J. Walsh, and M. S. Shephard, "Accessing CAD Geometry for Mesh Generation," in *12th International Meshing Roundtable, Sandia National Laboratories*, 2003.

[14] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, and E. Rank, "Integrating cad and numerical analysis: 'dirty geometry' handling using the finite cell method," *Computer Methods in Applied Mechanics and Engineering*, vol. 351, pp. 808–835, 2019.

[15] L. Kudela, "Highly accurate subcell integration in the context of the finite cell method," Master's thesis, Technical University of Munich, 2013.

[16] L. Kudela, N. Zander, T. Bog, S. Kollmannsberger, and E. Rank, "Efficient and accurate numerical quadrature for immersed boundary methods," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 2, no. 1, p. 10, 2015.

[17] L. Kudela, N. Zander, S. Kollmannsberger, and E. Rank, "Smart octrees: Accurately integrating discontinuous functions in 3D," *Computer Methods in Applied Mechanics and Engineering*, vol. 306, pp. 406–426, 2016.

[18] L. Kudela, F. Frischmann, O. E. Yossef, S. Kollmannsberger, Z. Yosibash, and E. Rank, "Image-based mesh generation of tubular geometries under circular motion in refractive environments," *Machine Vision and Applications*, vol. 29, no. 5, pp. 719–733, 2018.

[19] O. C. Zienkiewicz, R. L. Taylor, O. C. Zienkiewicz, and R. L. Taylor, *The finite element method*, vol. 36. McGraw-Hill London, 1977.

[20] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Mineola, New York: Dover Publications, 2000.

[21] B. Szabó and I. Babuška, *Finite element analysis*. John Wiley & Sons, 1991.

[22] A. Düster, E. Rank, and B. Szabó, "The p-version of the finite element and finite cell methods," *Encyclopedia of Computational Mechanics Second Edition*, pp. 1–35, 2017.

[23] W. J. Gordon and C. A. Hall, "Transfinite element methods: blending-function interpolation over arbitrary curved element domains," *Numerische Mathematik*, vol. 21, no. 2, pp. 109–129, 1973.

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

[25] K. Ho-Le, "Finite element mesh generation methods: a review and classification," *Computer-Aided Design*, vol. 20, no. 1, pp. 27–38, 1988.

[26] J. F. Thompson, B. K. Soni, and N. P. Weatherill, *Handbook of grid generation*. CRC press, 1998.

[27] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[28] C. Sorger, F. Frischmann, S. Kollmannsberger, and E. Rank, "TUM.GeoFrame: automated high-order hexahedral mesh generation for shell-like structures," *Engineering with Computers*, vol. 30, no. 1, pp. 41–56, 2014.

[29] J. Parvizian, A. Düster, and E. Rank, "Finite cell method," *Computational Mechanics*, vol. 41, no. 1, pp. 121–133, 2007.

[30] D. Schillinger and M. Ruess, "The finite cell method: a review in the context of higher-order structural analysis of CAD and image-based geometric models," *Archives of Computational Methods in Engineering*, vol. 22, no. 3, pp. 391–455, 2015.

[31] A. Düster, J. Parvizian, Z. Yang, and E. Rank, "The finite cell method for three-dimensional problems of solid mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 45-48, pp. 3768–3782, 2008.

[32] D. Schillinger, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, and E. Rank, "Small and large deformation analysis with the p- and B-spline versions of the finite cell method," *Computational Mechanics*, vol. 50, no. 4, pp. 445–478, 2012.

[33] M. Ruess, D. Schillinger, Y. Bazilevs, V. Varduhn, and E. Rank, "Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method," *International Journal for Numerical Methods in Engineering*, vol. 95, no. 10, pp. 811–846, 2013.

[34] I. Babuška, "The finite element method with penalty," *Mathematics of Computation*, vol. 27, no. 122, pp. 221–228, 1973.

[35] B. Flemisch and B. I. Wohlmuth, "Stable Lagrange multipliers for quadrilateral meshes of curved interfaces in 3D," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 8, pp. 1589–1602, 2007.

[36] A. Embar, J. Dolbow, and I. Harari, "Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements," *International Journal for Numerical Methods in Engineering*, vol. 83, no. 7, pp. 877–898, 2010.

[37] D. Schillinger and E. Rank, "An unfitted hp-adaptive finite element method based on hierarchical B-splines for interface problems of complex geometry," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 47-48, pp. 3358–3380, 2011.

[38] N. Zander, T. Bog, S. Kollmannsberger, D. Schillinger, and E. Rank, "Multi-level hp-adaptivity: high-order mesh adaptivity without the difficulties of constraining hanging nodes," *Computational Mechanics*, vol. 55, no. 3, pp. 499–517, 2015.

[39] D. D'Angella, S. Kollmannsberger, E. Rank, and A. Reali, "Multi-level Bézier extraction for hierarchical local refinement of isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 328, pp. 147–174, 2018.

[40] F. de Prenter, C. V. Verhoosel, G. J. van Zwieten, and E. H. van Brummelen, "Condition number analysis and preconditioning of the finite cell method," *Computer Methods in Applied Mechanics and Engineering*, vol. 316, pp. 297–327, 2017.

[41] A. Abedian, J. Parvizian, A. Düster, H. Khademyzadeh, and E. Rank, "Performance of different integration schemes in facing discontinuities in the finite cell method," *International Journal of Computational Methods*, vol. 10, no. 03, p. 1350002, 2013.

[42] M. Ruess, D. Tal, N. Trabelsi, Z. Yosibash, and E. Rank, "The finite cell method for bone simulations: verification and validation," *Biomechanics and Modeling in Mechanobiology*, vol. 11, no. 3-4, pp. 425–437, 2012.

[43] B. Wassermann, S. Kollmannsberger, T. Bog, and E. Rank, "From geometric design to numerical analysis: a direct approach using the finite cell method on constructive solid geometry," *Computers & Mathematics with Applications*, vol. 74, no. 7, pp. 1703–1726, 2017.

[44] M. Elhaddad, N. Zander, S. Kollmannsberger, A. Shadavakhsh, V. Nübel, and E. Rank, "Finite cell method: High-order structural dynamics for complex geometries," *International Journal of Structural Stability and Dynamics*, vol. 15, no. 07, p. 1540018, 2015.

[45] B. Müller, F. Kummer, and M. Oberlack, "Highly accurate surface and volume integration on implicit domains by means of moment-fitting," *International Journal for Numerical Methods in Engineering*, vol. 96, no. 8, pp. 512–528, 2013.

[46] M. Joulaian, S. Hubrich, and A. Düster, "Numerical integration of discontinuities on arbitrary domains based on moment fitting," *Computational Mechanics*, vol. 57, no. 6, pp. 979–999, 2016.

[47] S. Hubrich, P. Di Stolfo, L. Kudela, S. Kollmannsberger, E. Rank, A. Schröder, and A. Düster, "Numerical integration of discontinuous functions: moment fitting and smart octree," *Computational Mechanics*, vol. 60, no. 5, pp. 863–881, 2017.

[48] S. Hubrich and A. Düster, "Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting," *Computers & Mathematics with Applications*, vol. 77, no. 7, pp. 1983–1997, 2019.

[49] K. W. Cheng and T.-P. Fries, "Higher-order XFEM for curved strong and weak discontinuities," *International Journal for Numerical Methods in Engineering*, vol. 82, no. 5, pp. 564–590, 2010.

[50] G. Legrain, "A NURBS enhanced extended finite element approach for unfitted CAD analysis," *Computational Mechanics*, vol. 52, no. 4, pp. 913–929, 2013.

[51] R. Sevilla, S. Fernández-Méndez, and A. Huerta, "NURBS-enhanced finite element method (NEFEM)," *International Journal for Numerical Methods in Engineering*, vol. 76, no. 1, pp. 56–83, 2008.

[52] T.-P. Fries and S. Omerović, "Higher-order accurate integration of implicit geometries," *International Journal for Numerical Methods in Engineering*, vol. 106, no. 5, pp. 323–371, 2016.

[53] E. Nadal, J. Ródenas, J. Albelda, M. Tur, J. Tarancón, and F. Fuenmayor, "Efficient finite element methodology based on cartesian grids: application to structural shape optimization," in *Abstract and Applied Analysis*, vol. 2013, Hindawi, 2013.

[54] O. Marco, R. Sevilla, Y. Zhang, J. J. Ródenas, and M. Tur, "Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry," *International Journal for Numerical Methods in Engineering*, vol. 103, no. 6, pp. 445–468, 2015.

[55] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 21, pp. 163–169, Association for Computing Machinery, 1987.

[56] G. Királyfalvi and B. A. Szabó, "Quasi-regional mapping for the p-version of the finite element method," *Finite Elements in Analysis and Design*, vol. 27, no. 1, pp. 85–97, 1997.

[57] Q. Chen and I. Babuška, "Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle," *Computer Methods in Applied Mechanics and Engineering*, vol. 128, no. 3-4, pp. 405–417, 1995.

[58] C. Verhoosel, G. van Zwieten, B. van Rietbergen, and R. de Borst, "Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone," *Computer Methods in Applied Mechanics and Engineering*, vol. 284, pp. 138–164, 2015. Isogeometric Analysis Special Issue.

[59] I. Khan, "Motor cycle engine internal setup - connecting rod." `https://grabcad.com/library/motor-cycle-engine-internal-setup-connecting-rod-1`, 2015. [Online, accessed 19$^{\text{th}}$ October 2015].

[60] R. J. Hocken and P. H. Pereira, *Coordinate measuring machines and systems*. CRC Press, 2016.

[61] C. Butler, "An investigation into the performance of probes on coordinate measuring machines," *Industrial Metrology*, vol. 2, no. 1, pp. 59–70, 1991.

[62] T. Várady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models–an introduction," *Computer-Aided Design*, vol. 29, no. 4, pp. 255–268, 1997.

[63] M. Vermeulen, P. Rosielle, and P. Schellekens, "Design of a high-precision 3D-coordinate measuring machine," *CIRP Annals*, vol. 47, no. 1, pp. 447–450, 1998.

[64] Y. Li and P. Gu, "Free-form surface inspection techniques state of the art review," *Computer-Aided Design*, vol. 36, no. 13, pp. 1395–1417, 2004.

[65] P. Blondel and B. J. Murton, *Handbook of seafloor sonar imagery*, vol. 7. Wiley Chichester, UK, 1997.

[66] A. Fenster, D. B. Downey, and H. N. Cardinal, "Three-dimensional ultrasound imaging," *Physics in Medicine & Biology*, vol. 46, no. 5, p. R67, 2001.

[67] J. Blitz and G. Simpson, *Ultrasonic methods of non-destructive testing*, vol. 2. Springer Science & Business Media, 1995.

[68] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation," *Sensors*, vol. 9, pp. 568–601, Jan 2009.

[69] F. Blais, M. Rioux, and J.-A. Beraldin, "Practical considerations for a design of a high precision 3-D laser scanner system," in *Optomechanical and electro-optical design of industrial systems*, vol. 959, pp. 225–247, International Society for Optics and Photonics, 1988.

[70] B. Curless and M. Levoy, "Better optical triangulation through spacetime analysis," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 987–994, IEEE, 1995.

[71] K. L. Boyer and A. C. Kak, "Color-encoded structured light for rapid active ranging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 14–28, 1987.

[72] O. Hall-Holt and S. Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects," in *Proceedings of the Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 359–366, IEEE, 2001.

[73] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[74] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Heidelberg: Springer-Verlag, 1st ed., 2010.

[75] A. Heyden and M. Pollefeys, "Multiple view geometry," in *Emerging Topics in Computer Vision*, Prentice Hall PTR, 2005.

[76] R. I. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[77] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[78] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[79] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[80] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, pp. 2564–2571, IEEE, 2011.

[81] A. W. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *European Conference on Computer Vision (ECCV)*, pp. 311–326, Springer, 1998.

[82] P. Moulon, P. Monasse, and R. Marlet, "Adaptive structure from motion with a contrario model estimation," in *Asian Conference on Computer Vision (ACCV)*, pp. 257–270, Springer, 2012.

[83] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3248–3255, 2013.

[84] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal graphs for efficient structure from motion," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[85] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision Algorithms: Theory and Practice* (B. Triggs, A. Zisserman, and R. Szeliski, eds.), pp. 298–372, Springer, 1999.

[86] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European Conference on Computer Vision (ECCV)*, pp. 29–42, Springer, 2010.

[87] K. H. Wong and M. M.-Y. Chang, "3D model reconstruction by constrained bundle adjustment," in *Proceedings of the 17th International Conference on Pattern Recogniion*, vol. 3, pp. 902–905, IEEE, 2004.

[88] H.-G. Maas, "On the accuracy potential in underwater/multimedia photogrammetry," *Sensors*, vol. 15, no. 8, pp. 18140–18152, 2015.

[89] R. Kotowski, "Phototriangulation in multi-media photogrammetry," *International Archives of Photogrammetry and Remote Sensing*, vol. 27, no. B5, pp. 324–334, 1988.

[90] P. Drap, J. Seinturier, D. Scaradozzi, P. Gambogi, L. Long, and F. Gauch, "Photogrammetry for virtual exploration of underwater archeological sites," in *Proceedings of the 21st International Symposium, CIPA*, p. 1e6, 2007.

[91] M. Shortis and E. H. D. Abdo, "A review of underwater stereo-image measurement for marine biology and ecology applications," in *Oceanography and marine biology*, pp. 269–304, CRC Press, 2016.

[92] T. Kwasnitschka, T. H. Hansteen, C. W. Devey, and S. Kutterolf, "Doing fieldwork on the seafloor: photogrammetric techniques to yield 3D visual models from ROV video," *Computers & Geosciences*, vol. 52, pp. 218–226, 2013.

[93] H.-G. Maas and A. Gruen, "Digital photogrammetric techniques for high-resolution three-dimensional flow velocity measurements," *Optical Engineering*, vol. 34, no. 7, pp. 1970–1977, 1995.

[94] A. Jordt-Sedlazeck and R. Koch, "Refractive structure-from-motion on underwater images," in *Proceedings of the 2013 IEEE international conference on Computer Vision (ICCV)*, pp. 57–64, IEEE, 2013.

[95] A. Jordt, K. Köser, and R. Koch, "Refractive 3D reconstruction on underwater images," *Methods in Oceanography*, vol. 15, pp. 90–113, 2016.

[96] X. Chen and Y.-H. Yang, "Two-view camera housing parameters calibration for multi-layer flat refractive interface," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 524–531, 2014.

[97] K. N. Kutulakos and E. Steger, "A theory of refractive and specular 3D shape by light-path triangulation," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 13–29, 2008.

[98] Y. Qian, M. Gong, and Y.-H. Yang, "Stereo-based 3d reconstruction of dynamic fluid surfaces by global optimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1269–1278, 2017.

[99] A. S. Glassner, ed., *An Introduction to Ray Tracing*. London, UK, UK: Academic Press Ltd., 1989.

[100] C. Mulsow, "A flexible multi-media bundle approach," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, pp. 472–477, 2010.

[101] J. Belden, "Calibration of multi-camera systems with refractive interfaces," *Experiments in Fluids*, vol. 54, no. 2, p. 1463, 2013.

[102] M. Lhuillier and L. Quan, "Robust dense matching using local and global geometric constraints," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 1, pp. 968–972, IEEE, 2000.

[103] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.

[104] M. Gross and H. Pfister, *Point-based graphics*. Elsevier, 2011.

[105] T. P. Kersten, F. Tschirschwitz, and S. Deggim, "Development of a virtual museum including a 4D presentation of building history in virtual reality," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 361, 2017.

[106] 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*, pp. 131–144, ACM Press/Addison-Wesley Publishing Co., 2000.

[107] M. Schleinkofer, *Wissensbasierte Unterstützung zur Erstellung von Produktmodellen im Baubestand*. Dissertation, Technische Universität München, München, 2007.

[108] T. Várady, P. Benkő, and G. Kos, "Reverse engineering regular objects: simple segmentation and surface fitting procedures," *International Journal of Shape Modeling*, vol. 4, no. 03n04, pp. 127–141.

[109] P. Benkő, R. R. Martin, and T. Várady, "Algorithms for reverse engineering boundary representation models," *Computer-Aided Design*, vol. 33, no. 11, pp. 839–851, 2001.

[110] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer graphics forum*, vol. 26, pp. 214–226, Wiley Online Library, 2007.

[111] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[112] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 998–1005, IEEE, 2010.

[113] R. Schnabel, P. Degener, and R. Klein, "Completion and reconstruction with primitive shapes," in *Computer Graphics Forum*, vol. 28, pp. 503–512, Wiley Online Library, 2009.

[114] N. Silberman, D. Sontag, and R. Fergus, "Instance segmentation of indoor scenes using a coverage loss," in *European Conference on Computer Vision*, pp. 616–631, Springer, 2014.

[115] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen, "Non-local scan consolidation for 3d urban scenes," in *ACM Transactions on Graphics (TOG)*, vol. 29, p. 94, Association for Computing Machinery, 2010.

[116] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer, "A survey of urban reconstruction," in *Computer Graphics Forum*, vol. 32, pp. 146–177, Wiley Online Library, 2013.

[117] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra, "Globfit: Consistently fitting primitives by discovering global relations," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 52, 2011.

[118] S. Tuttas, A. Braun, A. Borrmann, and U. Stilla, "Acquisition and consecutive registration of photogrammetric point clouds for construction progress monitoring using a 4D BIM," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 3–15, 2017.

[119] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, "Semantic segmentation of aerial images with an ensemble of CNNs," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, p. 473, 2016.

[120] Y. Xu, Z. Ye, W. Yao, R. Huang, X. Tong, L. Hoegner, and U. Stilla, "Classification of LiDAR point clouds using supervoxel-based detrended feature and perception-weighted graphical model," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[121] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, vol. 36, pp. 301–329, Wiley Online Library, 2017.

[122] W. Wang, H. Pottmann, and Y. Liu, "Fitting B-spline curves to point clouds by curvature-based squared distance minimization," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 2, pp. 214–238, 2006.

[123] J. Hoschek, "Intrinsic parametrization for approximation," *Computer Aided Geometric Design*, vol. 5, no. 1, pp. 27–31, 1988.

[124] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 295–302, Association for Computing Machinery, 1994.

[125] S. Flöry and M. Hofer, "Surface fitting and registration of point clouds using approximations of the unsigned distance function," *Computer Aided Geometric Design*, vol. 27, no. 1, pp. 60–77, 2010.

[126] H. Pottmann and S. Leopoldseder, "A concept for parametric surface fitting which avoids the parametrization problem," *Computer Aided Geometric Design*, vol. 20, no. 6, pp. 343–362, 2003.

[127] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005.

[128] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, vol. 153. Springer Science & Business Media, 2006.

[129] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *SIGGRAPH Computer Graphics*, vol. 26, pp. 71–78, July 1992.

[130] D. Levin, "Mesh-independent surface interpolation," in *Geometric modeling for scientific visualization*, pp. 37–49, Springer, 2004.

[131] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[132] M. Alexa and A. Adamson, "On normals and projection operators for surfaces defined by point sets," in *Proceedings of the First Eurographics conference on Point-Based Graphics*, pp. 149–155, Eurographics Association, 2004.

[133] N. Amenta and Y. J. Kil, "Defining point-set surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 264–270, Association for Computing Machinery, 2004.

[134] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 67–76, ACM, 2001.

[135] H. Wendland, *Scattered data approximation*, vol. 17. Cambridge University Press, 2004.

[136] I. Macedo, J. P. Gois, and L. Velho, "Hermite radial basis functions implicits," in *Computer Graphics Forum*, vol. 30, pp. 27–42, Wiley Online Library, 2011.

[137] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proceedings of the third Eurographics symposium on Geometry processing*, p. 73, Eurographics Association, 2005.

[138] J. Wilhelms and A. Van Gelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 3, pp. 201–227, 1992.

[139] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill, "Octree-based decimation of marching cubes surfaces," in *Proceedings of the Seventh Annual IEEE Visualization'96*, pp. 335–342, IEEE, 1996.

[140] R. Westermann, L. Kobbelt, and T. Ertl, "Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces," *The Visual Computer*, vol. 15, no. 2, pp. 100–111, 1999.

[141] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 57–66, Association for Computing Machinery, 2001.

[142] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pp. 70–76, IEEE, 2004.

[143] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, 2006.

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

[145] O. E. Yossef, "Experimental investigation of arteries' mechanical properties: compressibility, passive and active responses," Master's thesis, Dept. of Mechanical Eng., Ben-Gurion University, Israel, 2017.

[146] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[147] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees." `https://github.com/jlblancoc/nanoflann`, 2014.

[148] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[149] F. Aurenhammer and R. Klein, "Voronoi diagrams," *Handbook of Computational Geometry*, vol. 5, no. 10, pp. 201–290.

[150] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry*. Springer, 1997.

[151] D. M. Hawkins, *Identification of Outliers*. Springer, 1980.

[152] C.-T. Lu, D. Chen, and Y. Kou, "Algorithms for spatial outlier detection," in *Third IEEE International Conference on Data Mining (ICDM)*, pp. 597–600, IEEE, 2003.

[153] S. Sotoodeh, "Outlier detection in laser scanner point clouds," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 297–302, 2006.

[154] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung, and A. Sorkine-Hornung, "Point cloud noise and outlier removal for image-based 3D reconstruction," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 118–127, IEEE, 2016.

[155] A. Nurunnabi, G. West, and D. Belton, "Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data," *Pattern Recognition*, vol. 48, no. 4, pp. 1404–1419, 2015.

[156] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 813–822, Springer, 2009.

[157] O. Schall, A. Belyaev, and H.-P. Seidel, "Robust filtering of noisy scattered point data," in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pp. 71–144, IEEE, 2005.

[158] P. Chalmoviansk̀y and B. Jüttler, "Filling holes in point clouds," in *Mathematics of Surfaces*, pp. 196–212, Springer Berlin Heidelberg, 2003.

[159] Y. Quinsat *et al.*, "Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics," *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 1-4, pp. 411–421, 2015.

[160] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43–72, 1994.

[161] C. L. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3D scans," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 109–118, Association for Computing Machinery, 1995.

[162] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[163] N. Amenta, M. W. Bern, M. Kamvysselis, *et al.*, "A new Voronoi-based surface reconstruction algorithm," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, vol. 98, pp. 415–421, Association for Computing Machinery, 1998.

[164] T. K. Dey, J. Giesen, and J. Hudson, "Delaunay based shape reconstruction from large data," in *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pp. 19–27, IEEE Press, 2001.

[165] J. Davis, S. R. Marschner, M. Garr, and M. Levoy, "Filling holes in complex surfaces using volumetric diffusion," in *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission*, pp. 428–441, IEEE, 2002.

[166] A. Hornung and L. Kobbelt, "Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, pp. 41–50, Eurographics Association, 2006.

[167] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, "Voronoi-based variational reconstruction of unoriented point sets," in *Proceedings of the fifth Eurographics symposium on Geometry processing*, pp. 39–48, Eurographics Association, 2007.

[168] P. Liepa, "Filling holes in meshes," in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 200–205, Eurographics Association, 2003.

[169] J. Wang and M. M. Oliveira, "Filling holes on locally smooth surfaces reconstructed from point clouds," *Image and Vision Computing*, vol. 25, no. 1, pp. 103–113, 2007.

[170] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 71, Association for Computing Machinery, 2009.

[171] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen, "L1-medial skeleton of point cloud," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 65–1, 2013.

[172] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via Laplacian based contraction," in *2010 Shape Modeling International Conference*, pp. 187–197, IEEE, 2010.

[173] C. Fox, *An introduction to the calculus of variations*. Courier Corporation, 1987.

[174] A. V. Mobley, M. P. Carroll, and S. A. Canann, "An object oriented approach to geometry defeaturing for finite element meshing," in *7th International Meshing Roundtable, Sandia National Labs*, pp. 547–563, 1998.

[175] W. R. Quadros and S. J. Owen, "Defeaturing CAD models using a geometry-based size field and facet-based reduction operators," in *Proceedings of the 18th international meshing roundtable*, pp. 301–318, Springer, 2009.

[176] C. Kublik, N. M. Tanushev, and R. Tsai, "An implicit interface boundary integral method for Poisson's equation on arbitrary domains," *Journal of Computational Physics*, vol. 247, pp. 279–311, 2013.

[177] B. Engquist, A.-K. Tornberg, and R. Tsai, "Discretization of Dirac delta functions in level set methods," *Journal of Computational Physics*, vol. 207, no. 1, pp. 28–51, 2005.

[178] H. G. Lee and J. Kim, "Regularized Dirac delta functions for phase field models," *International Journal for Numerical Methods in Engineering*, vol. 91, no. 3, pp. 269–288, 2012.

[179] S. K. Stoter, P. Müller, L. Cicalese, M. Tuveri, D. Schillinger, and T. J. Hughes, "A diffuse interface method for the Navier–Stokes/Darcy equations: Perfusion profile for a patient-specific human liver based on MRI scans," *Computer Methods in Applied Mechanics and Engineering*, vol. 321, pp. 70–102, 2017.

[180] L. H. Nguyen, S. K. Stoter, M. Ruess, M. A. Sanchez Uribe, and D. Schillinger, "The diffuse Nitsche method: Dirichlet constraints on phase-field boundaries," *International Journal for Numerical Methods in Engineering*, vol. 113, no. 4, pp. 601–633, 2018.

[181] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *Proceedings of the 2011 Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3057–3064, IEEE, 2011.

[182] S. Hill, *The early Byzantine churches of Cilicia and Isauria*. Variorum Aldershot, UK, 1996.

[183] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–330, 2015.

[184] M. Lo Brutto, A. Garraffa, and P. Meli, "UAV platforms for cultural heritage survey: first results," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 5, pp. 227–234, 2014.

[185] M. L. Brutto, A. Borruso, and A. D'argenio, "UAV systems for photogrammetric data acquisition of archaeological sites," *International Journal of Heritage in the Digital Era*, vol. 1, no. Supplement 1, pp. 7–13, 2012.

[186] E. Nocerino, F. Menna, F. Remondino, and R. Saleri, "Accuracy and block deformation analysis in automatic UAV and terrestrial photogrammetry–lesson learnt," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, pp. 203–208, 2013.

[187] G. Guidi, M. Russo, S. Ercoli, F. Remondino, A. Rizzi, and F. Menna, "A multi-resolution methodology for the 3D modeling of large and complex archeological areas," *International Journal of Architectural Computing*, vol. 7, no. 1, pp. 39–55, 2009.

[188] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[189] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *Proceedings of the 2016 European Conference on Computer Vision (ECCV)*, pp. 501–518, 2016.