Volumetric Subdivision for Efficient Integrated Modeling and Simulation

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation im Fachbereich Informatik von Christian Altenhofen aus Bad Kreuznach
Tag der Einreichung: 31.07.2020, Prüfung: 5.11.2020, Darmstadt

1. Gutachten: Prof. Dr. techn. Dr.-Ing. eh. Dieter W. Fellner
2. Gutachten: Prof. Dr.-Ing. André Stork
3. Gutachten: Assoc. Prof. M.Sc. PhD Ursula Augsdörfer

Technische Universität Darmstadt – D 17

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Volumetric Subdivision for Efficient Integrated Modeling and Simulation

Doctoral thesis in Computer Science by Christian Altenhofen

1. Review: Prof. Dr. techn. Dr.-Ing. eh. Dieter W. Fellner
2. Review: Prof. Dr.-Ing. André Stork
3. Review: Assoc. Prof. M.Sc. PhD Ursula Augsdörfer

Date of submission: 31.07.2020
Thesis defense: 5.11.2020, Darmstadt

Technische Universität Darmstadt – D 17

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, den 31.07.2020

[illegible]

Christian Altenhofen

Abstract

Continuous surface representations, such as *B-spline* and *Non-Uniform Rational B-spline* (NURBS) surfaces are the de facto standard for modeling 3D objects – thin shells and solid objects alike – in the field of *Computer-Aided Design* (CAD). For performing physically based simulation, *Finite Element Analysis* (FEA) has been the industry standard for many years. In order to analyze physical properties such as stability, aerodynamics, or heat dissipation, the continuous models are discretized into finite element (FE) meshes.

A tight integration of and a smooth transition between geometric design and physically based simulation are key factors for an efficient design and engineering workflow. Converting a CAD model from its continuous boundary representation (B-Rep) into a discrete volumetric representation for simulation is a time-consuming process that introduces approximation errors and often requires manual interaction by the engineer. Deriving design changes directly from the simulation results is especially difficult as the meshing process is irreversible.

Isogeometric Analysis (IGA) tries to overcome this *meshing hurdle* by using the same representation for describing the geometry and for performing the simulation. Most commonly, IGA is performed on bivariate and trivariate spline representations (B-spline or NURBS surfaces and volumes) [HCB05]. While existing CAD B-Rep models can be used directly for simulating thin-shell objects, simulating solid objects requires a conversion from spline surfaces to spline volumes. As spline volumes need a trivariate tensor-product topology, complex 3D objects must be represented via trimming or by connecting multiple spline volumes, limiting the continuity to C^0 [ME16; DSB19].

As an alternative to NURBS or B-splines, subdivision models allow for representing complex topologies with as a single entity, removing the need for trimming or tiling and potentially providing higher continuity. While subdivision surfaces have shown promising results for designing and simulating shells [WHP11; Pan+15; RAF16], IGA on subdivision volumes remained mostly unexplored apart from the work of Burkhart et al. [BHU10b; Bur11].

In this dissertation, I investigate how volumetric subdivision representations are beneficial for a tighter integration of geometric modeling and physically based simulation. Focusing on Catmull-Clark (CC) solids, I present novel techniques in the areas of efficient limit evaluation, volumetric modeling, numerical integration, and mesh quality analysis. I present an efficient link to FEA, as well as my IGA approach on CC solids that improves upon Burkhart et al.'s proof of concept [BHU10b] with constant-time limit evaluation, more accurate integration, and higher mesh quality.

Efficient limit evaluation is a key requirement when working with subdivision models in geometric design, visualization, simulation, and 3D printing. In this dissertation, I present the first method for constant-time volumetric limit evaluation of CC solids. It is faster than the subdivision-based approach by Burkhart et al. [BHU10b] for every topological constellation and parameter point that would require more than two local subdivision steps.

Adapting the concepts of well-known surface modeling tools, I present a volumetric modeling environment for CC-solid control meshes. Consistent volumetric modeling operations built from a set of novel volumetric *Euler operators* allow for creating and modifying topologically consistent volumetric meshes. Furthermore, I show how to manipulate groups of control points via parameters, how to avoid intersections with inner control points while modeling the outer surface, and how to use CC solids in the context of multi-material additive manufacturing.

For coupling of volumetric subdivision models with established FE frameworks, I present an efficient and consistent tetrahedral mesh generation technique for CC solids. The technique exploits the inherent volumetric structure of CC-solid models and is at least $26\times$ faster than the tetrahedral meshing algorithm provided by CGAL [Jam+15]. This allows to re-create or update the tetrahedral mesh almost instantly when changing the CC-solid model. However, the mesh quality strongly depends on the quality of the control mesh.

In the context of structural analysis, I present my IGA approach on CC solids. The IGA approach yields converging stimulation results for models with fewer elements and fewer degrees of freedom than FE simulations on tetrahedral meshes with linear and higher-order basis functions. The solver also requires fewer iterations to solve the linear system due to the higher continuity throughout the simulation model provided by the subdivision basis functions. Extending Burkhart et al.'s method [BHU10b], my hierarchical quadrature scheme for irregular CC-solid cells increases the accuracy of the integrals for computing surface areas and element stiffnesses. Furthermore, I introduce a quality metric that quantifies the parametrization quality of the limit volume, revealing distortions, inversions, and singularities. The metric shows that cells with multiple adjacent boundary faces induce singularities in the limit, even for geometrically well-shaped control meshes. Finally, I present a set of topological operations for splitting such boundary cells – resolving the singularities. These improvements further reduce the amount of elements required to obtain converging results as well as the time required for solving the linear system.

Zusammenfassung

Kontinuierliche Repräsentationsformen, wie z.B. *B-Spline-* und *NURBS*-Oberflächen (*Non-Uniform Rational B-Splines*), sind der De-facto-Standard für die Beschreibung von 3D-Modellen im Bereich des *Computer-Aided Design* (CAD). Der Industriestandard für die Durchführung physikalisch-basierter Simulationen ist seit vielen Jahren die *Finite-Elemente-Analyse* (FEA). Um physikalische Eigenschaften wie Stabilität, Aerodynamik oder Wärmeleitung zu simulieren, werden die kontinuierlichen Oberflächenmodelle in diskrete Finite-Elemente-Netze (FE-Netze) konvertiert.

Eine enge Integration von Design und Simulation ist eine wichtige Voraussetzung für einen effizienten Design- und Engineering-Workflow. Dazu gehört auch ein reibungsloser Übergang zwischen beiden Disziplinen. Die Konvertierung eines CAD-Modells von seiner kontinuierlichen Oberflächenbeschreibung (*Boundary Representation*, B-Rep) in eine diskrete volumetrische Darstellung für die Simulation ist ein zeitaufwendiger Prozess, der oft manuelle Eingriffe durch den Ingenieur erfordert. Durch die Diskretisierung werden unvermeidlich Approximationsfehler eingeführt. Designanpassungen direkt aus den Simulationsergebnissen abzuleiten ist schwierig, da der Vernetzungsprozess irreversibel ist.

In der *Isogeometrischen Analyse* (IGA) wird versucht, diese Hürde zu umgehen, indem eine einheitliche Repräsentation für die Beschreibung der Geometrie und die Durchführung der Simulation verwendet wird. Am häufigsten wird IGA mit bivariaten und trivariaten Splines (B-Spline- oder NURBS-Flächen, bzw. -Volumen) durchgeführt [HCB05]. Während vorhandene B-Rep-Modelle direkt für die Simulation schalenartiger Objekte, wie z.B. Blechteile, verwendet werden können, erfordert die Simulation volumetrischer Bauteile eine Umwandlung von Spline-Flächen in Spline-Volumen. Da trivariate Splines eine Tensor-Produkt-Topologie benötigen, müssen komplexe 3D-Objekte mit Hilfe von *Trimming* oder durch Verbinden mehrerer Spline-Körper (sog. *Tiling*) repräsentiert werden. Diese Vorgehensweise beschränkt die Stetigkeit des Modells jedoch auf C^0 [ME16; DSB19].

Anders als NURBS oder B-Splines erlauben es Subdivisionsmodelle, komplexe Topologien als eine einzige topologische Entität darzustellen. Dies gewährleistet eine potenziell höhere Stetigkeit im Modell, und die Notwendigkeit für Trimming oder Tiling entfällt. Während mit Subdivisionsflächen vielversprechende Ergebnisse bei der Modellierung und Simulation von schalenartigen Bauteilen erreicht wurden [WHP11; Pan+15; RAF16], wurde IGA auf Subdivisionsvolumen – abgesehen von den Arbeiten von Burkhart et al. [BHU10b; Bur11] – bisher kaum untersucht.

In dieser Dissertation erforsche ich, wie Subdivisionsvolumen genutzt werden können, um die Integration von geometrischer Modellierung und physikalisch-basierter Simulation zu stärken und den (iterativen) Konstruktionsprozess aus Design und Simulation effizienter zu gestalten. Basierend auf *Catmull-Clark-Subdivisionsvolumen* (CC Solids) stelle ich neue Techniken für die effiziente Limit-Auswertung, die volumetrische Modellierung, die numerische Integration und zur Messung und Verbesserung der Netzqualität vor. Ich präsentiere eine effiziente Art, CC Solids mit klassischer FEA zu koppeln und stelle meinen IGA-Ansatz vor, der auf dem Proof-of-Concept von Burkhart et al. aufbaut [BHU10b] und diesen durch performantere Limit-Auswertung, präzisere Integration und höhere Netzqualität erweitert.

Eine effiziente Limit-Auswertung ist ein wichtiger Baustein bei der Verwendung von Subdivisionsmodellen für geometrische Modellierung, Visualisierung, Simulation und 3D-Druck. In dieser Dissertation stelle ich die erste Methode zur volumetrischen Limit-Auswertung für CC Solids vor, die es erlaubt, jeden Limit-Punkt in konstanter Zeit auszuwerten. Die Methode ist schneller als der auf Subdivision basierende Ansatz von Burkhart et al. [BHU10b] für jede topologische Konstellation und jeden Parameterpunkt, die/der mehr als zwei lokale Subdivisionsschritte erfordern würde.

Aufbauend auf den Konzepten bekannter Software-Tools für Oberflächenmodellierung, präsentiere ich eine Modellierumgebung für volumetrische Kontrollnetze. Konsistente volumetrische Modellieroperationen, die aus neuartigen volumetrischen *Euler-Operatoren* aufgebaut sind, ermöglichen die Erstellung und Modifikation topologisch konsistenter volumetrischer Netze. Darüber hinaus zeige ich Techniken, um Gruppen von Kontrollpunkten über Parameter zu manipulieren und Überschneidungen mit inneren Kontrollpunkten beim Modellieren der äußeren Oberfläche zu vermeiden. Außerdem stelle ich Anwendungen für CC Solids im Kontext des Multimaterial-3D-Drucks vor.

Um die Vorteile von Subdivisionsvolumen auch in Kombination mit etablierten Finite-Elemente-Frameworks nutzen zu können, stelle ich eine effiziente und konsistente Methode zur Erzeugung von Tetraedernetzen aus CC-Solid-Modellen vor. Die Technik nutzt die inhärente volumetrische Struktur von CC-Solid-Modellen und ist mindestens $26\times$

schneller als der von CGAL bereitgestellte Vernetzungsalgorithmus [Jam+15], der auf Oberflächenmodellen arbeitet. Dadurch kann das Tetraedernetz nach Modifikation des CC-Solid-Modells in wenigen Millisekunden aktualisiert oder neu erstellt werden. Anders als bei CGAL [Jam+15], hängt die Netzqualität jedoch direkt von der Qualität des Kontrollnetzes ab.

Am Beispiel der Strukturanalyse stelle ich meinen IGA-Ansatz für Catmull-Clark Subdivisionsvolumen vor. Die Simulationsergebnisse des IGA-Ansatzes konvergieren bereits bei weniger Elementen und weniger Freiheitsgraden als FE-Simulationen mit Tetraedernetzen mit linearen und quadratischen Basisfunktionen. Aufgrund der durch die Subdivisionsbasisfunktionen erzeugten höheren Stetigkeit im Simulationsgebiet, benötigt der Gleichungssystemlöser zudem weniger Iterationen zur Lösung des linearen Gleichungssystems. In Erweiterung von Burkhart et al.s Ansatz [BHU10b] präsentiere ich ein hierarchisches Quadraturschema für irreguläre CC-Solid-Zellen, das die Genauigkeit der Integrale bei der Berechnung von Limit-Flächeninhalten und Elementsteifigkeiten erhöht. Ich führe eine Metrik zur Quantifizierung der Qualität von CC-Solid-Modellen ein, die die Parametrisierungsqualität des Limit-Volumens misst und Verzerrungen, Invertierungen und Singularitäten aufdeckt. Die Metrik zeigt, dass Zellen mit mehreren zusammenhängenden Randflächen Singularitäten im Limit-Volumen erzeugen, selbst wenn das Kontrollnetz eine hohe geometrische Netzqualität aufweist. Um diese Singularitäten aufzulösen und die Parametrisierungsqualität zu verbessern, stelle ich topologische Operationen zum Zerteilen solcher Randzellen vor. Dies reduziert die für die Konvergenz der Simulationsergebnisse benötigte Anzahl an Elementen weiter und beschleunigt zusätzlich das Lösen des Gleichungssystems.

Contents

1. Introduction **1**

1.1. Research Questions . . . 4

1.2. Contributions . . . 6

1.3. Structure . . . 9

2. Background and Related Work **11**

2.1. Surface Representations and Geometric Modeling . . . 11

2.1.1. Spline Surfaces . . . 12

2.1.2. Subdivision Surfaces . . . 15

2.2. Volumetric Representations . . . 27

2.2.1. Tetrahedral and Hexahedral Meshes . . . 27

2.2.2. Spline Volumes . . . 29

2.2.3. Subdivision Volumes . . . 31

2.3. Design Analysis . . . 33

2.3.1. Finite Element Analysis . . . 33

2.3.2. Spline-Based Isogeometric Analysis . . . 40

2.3.3. Subdivision-Based Isogeometric Analysis . . . 42

2.4. Summary . . . 44

3. Subdivision Solids and Volumetric Modeling **45**

3.1. Volumetric Catmull-Clark Subdivision . . . 46

3.1.1. The Subdivision Rules for Catmull-Clark Solids . . . 49

3.1.2. Precomputable Subdivision Matrices . . . 51

3.1.3. Local Parametrization . . . 53

3.2. Constant-Time Limit Evaluation for Catmull-Clark Solids . . . 54

3.2.1. The Regular Case . . . 55

3.2.2. Irregular Cases . . . 56

3.2.3. Boundary Cells . . . 62

3.2.4. Implementation and Performance . . . 65

3.3. Volumetric Modeling Operations 69
3.3.1. The Half-Face Data Structure 70
3.3.2. Volumetric Euler Operators 74
3.3.3. Local Refinement 79
3.3.4. Parametrizing Volumetric Control Meshes 81
3.3.5. Avoiding Inversions in the Volumetric Model under Interactive Modeling 85
3.4. Summary 87

4. Efficient Tetrahedral Mesh Generation with Catmull-Clark Solids 89
4.1. Tetrahedral Mesh Generation 91
4.1.1. Partitioning a Hexahedral Cell into Tetrahedra 91
4.1.2. Method A: Explicit Subdivision and Linear Interpolation 93
4.1.3. Method B: Explicit Subdivision and Interpolation with Subdivision Rules 95
4.1.4. Method C: Direct Limit Evaluation 96
4.2. Performance Evaluation 98
4.3. Consistency and Mesh Quality 102
4.3.1. Inserting Tetrahedral Nodes into Concave CC-Solid Cells 104
4.3.2. Laplacian Smoothing of Inner Points 105
4.3.3. Quantifying Mesh Quality 106
4.4. Summary 112

5. Isogeometric Analysis on Catmull-Clark Solids 115
5.1. Numerical Integration on Catmull-Clark Solids 116
5.1.1. Calculating the Area of Limit Patches and the Volume of Limit Cells 117
5.1.2. Computing Weights for External Forces 117
5.1.3. Computing Element Stiffnesses 119
5.1.4. The Hierarchical Quadrature Scheme 121
5.1.5. Accuracy Evaluation 124
5.1.6. Performance 128
5.2. Parametrization Quality of the Limit Volume 129
5.2.1. Quantifying Parametrization Quality 129
5.2.2. Critical Topological Configurations 131
5.2.3. Splitting Cells to Improve the Parametrization Quality 133
5.3. Structural Analysis with Catmull-Clark Solids 138
5.3.1. Initial Subdivision Steps 138
5.3.2. Applying External Forces 138

5.3.3. Assembling the Stiffness Matrix 140
5.3.4. Solving the System 140
5.3.5. Simulation Results 141
5.4. Summary 148

6. Applications 151
6.1. Integrating Modeling and Simulation with CC Solids and Tetrahedral Meshes 153
6.1.1. Interactive CC-Solid Modeling 153
6.1.2. Finite Element Simulation 156
6.1.3. Interactive Visual Feedback 157
6.2. Interactive Design Space Exploration for Mass-Customized 3D-Printed Objects 159
6.2.1. Customizing Parametrized CC-Solid Models 161
6.2.2. Finite Element Simulation 161
6.2.3. Guided Exploration of the Design Space 162
6.3. Modeling of Smoothly Graded Volumetric Material Properties for Multi-Material Additive Manufacturing 164
6.3.1. Material Definition on Volumetric Subdivision Models 165
6.3.2. Slicing and 3D Printing of Volumetric Multi-Material Models 165
6.4. CC-Solid-Based Internal Structures for AM Parts 167
6.4.1. Interactive Modeling of Inner Structures 169
6.4.2. Procedural Inner Structures 172
6.4.3. Creating the Combined Model 174
6.4.4. Simulating Parts with Internal Structures 176

7. Conclusion and Future Work 179

References 187

A. Publications 201
A.1. Publications as Main Author 201
A.2. Co-Authored Publications 202

B. Supervisory Activities 205
B.1. Master’s Theses 205
B.2. Bachelor’s Theses 205
B.3. GRIS Practical Courses 206

C. Acronyms 207

1. Introduction

For decades, computing power has been utilized by designers and engineers to develop new parts and improve existing designs. In almost every sector of mechanical engineering, parts are designed virtually using *Computer-Aided Design* (CAD). In addition to defining the shape of a part, its later use cases are also virtually described and evaluated by means of physically based simulation – most often using *Finite Element Analysis* (FEA). Increasing computational power allowed for more complex designs as well as more complex simulation scenarios over the years. Evaluating the physical properties of a part, such as its aerodynamic features, its stiffness under certain loads, or its heat dissipation behavior in a physically based simulation enables verifying a given design against its requirements without manufacturing a physical prototype. However, as such simulations can run for minutes, hours, days, or even weeks depending on the detailedness of the underlying physical model, a proper simulation setup has to be chosen based on the given situation. Especially in the early stages of the product development process, where design changes are common and frequent, choosing a *suitable* simulation setup can reduce the computation time and manufacturing costs significantly. It is in these early stages, where the accuracy of the simulation can be reduced to some extent in order to reduce the overall waiting time for the engineer. However, apart from an efficient simulation system, a smooth transition between the two stages – design and simulation – is crucial. This requires tight integration of all components, with as few conversion steps as possible. Although CAD and FEA have become industry standards over the years, geometric modeling and physically based simulation have evolved as individual disciplines. Their methods use the representation schemes that first and foremost suit the individual requirements of their corresponding area.

CAD geometry is often described by a so-called *Boundary Representation* (B-Rep). A B-Rep represents a solid 3D object by describing its outer surface based on mathematically defined smooth surface representations, such as Bézier, B-spline, or Non-Uniform Rational B-spline (NURBS) surfaces. In addition to the geometric description itself, CAD models may contain a history of the operations they are created with, including 2D sketches, extrusions, revolutions, Boolean operations, and trimming, as well as sets of parameters, defining radii, diameters, lengths, and distances. Although closed surface models, represented e.g. in B-Rep, implicitly describe their enclosed volume, CAD models usually don't include any volumetric parametrization. As B-Reps only contain information about the surface of an object, they cannot represent volumetric properties such as varying elasticity, porosity, or thermal conductivity at specific locations inside the object. Additionally, surface representations such as B-Reps do not provide the volumetric degrees of freedom necessary to resolve the physical effects in simulations.

Physical phenomena are generally described by *partial differential equations* (PDEs) defined over continuous domains. Performing a physically based simulation consists of solving one or multiple PDEs for a specific set of boundary conditions. Which PDEs to solve depends on the chosen physical domain (e.g. fluid dynamics or structural mechanics) as well as on the chosen simulation scenario. For a static (time independent) simulation in *computational structural mechanics* (CSM) for example, the goal is to find an equilibrium state between external forces and internal stresses for a given engineering part. However, analytical solutions to such simulation problems are almost always too complex or even impossible to compute. Therefore, numerical simulation methods such as the *Finite Element Method* (FEM) and the *Finite Volume Method* (FVM) have been developed. They discretize the given domain into *finite* sections and use those to calculate approximate numerical solutions to the corresponding PDEs. This boils down to solving a system of linear equations with several thousand to several billion unknowns.

For FEA, 3D objects are described by discrete volumetric representations – most often via tetrahedral or hexahedral meshes. Every tetrahedron or hexahedron represents one *finite element* with a set of linear or higher-order *basis functions* defined on it. Although very thin 3D objects with little volume (e.g. sheet metal parts) can be approximated with a discrete surface mesh, in general, volumetric meshes are required to correctly resolve the physical effects that take place inside an object. As CAD models are commonly represented by continuous surfaces, a conversion step into a discrete volumetric mesh is needed – the so-called *meshing* process. Meshing a B-Rep CAD model into e.g. a tetrahedral mesh can be a highly complex and time-consuming task. Besides the computational effort required for computing the topology and the geometry of the mesh itself, common problems in meshing include handling gaps and self-intersections between neighboring surface patches, especially when working with trimmed surfaces. Any construction history, parameters, or semantic information stored alongside the CAD geometry is also lost during the meshing process.

Conversion problems scale up in optimization loops, where the switch between the geometric domain and the simulation domain has to be performed in every iteration and where changes in the geometry (ideally) have to be derived directly from the simulation results. Automatic optimization processes, such as shape optimization, partly automate the early design phase by combining virtual design and numerical simulation in an iterative loop in order to find an *optimal* part design for a specific use case. Such optimization loops often require hundreds to thousands of iterations of performing the simulation and modifying the part's geometry based on the results. Even more than for manual part design, the efficiency of the simulation system as well as the transition between geometry modification and simulation greatly affect the overall required time and costs.

Aiming at closing this gap between geometric modeling and physically based simulation, the concept of *Isogeometric Analysis* (IGA) has been developed. IGA tries to overcome the incompatibility of representations by using one representation for both domains – geometric modeling and physically based simulation. This means using the same set of basis functions for describing the geometry as well as for solving the PDEs resembling the given simulation problem. The most common forms of IGA use bivariate and trivariate NURBS as presented by Hughes et al. [HCB05]. However, only for simulating *thin-shell* objects such as sheet metal parts, the CAD B-Rep model can be used directly. Solid 3D objects require a volumetric representation. For performing volumetric IGA on existing B-Rep models, the models first have to be converted into the trivariate representation. This can be seen as hexahedral meshing, which is also a highly complex process [Eri13; WL16]. Once the 3D objects exist in a trivariate NURBS representation, geometric changes as well as simulations can be performed without additional conversions.

As trivariate NURBS cells require a tensor product topology throughout the cell, 3D objects with a *branching* topology, such as a cube with two cylindrical holes, cannot be represented by a single cell. Such objects have to be modeled either by trimming the trivariate cell with a trimming surface or by using multiple cells connected with each other. However, the interfaces between cells have to be compatible and, in general, using multiple connected cells results in C^0 continuity at cell borders as described by Massarwi and Elber [ME16]. When solving simulation problems based on elliptic PDEs, e.g. heat dissipation or CSM, higher continuity between elements (C^1 or even C^2) increases the accuracy of the simulation results and reduces the required number of degrees of freedom compared to a simulation domain with C^0 continuity, as shown by Cottrell et al. [CHR07].

In 2010, Burkhart et al. presented a first isogeometric simulation approach for volumetric subdivision models, i.e. for Catmull-Clark (CC) solids [BHU10b]. CC solids as presented by Joy and MacCracken [JM99] can be seen as the volumetric extension of Catmull-Clark subdivision surfaces [CC78] and support polyhedral control meshes of almost arbitrary topology. Such control meshes may contain any type of polyhedron that subdivides into hexahedra after a finite number of subdivision steps. In contrast to trivariate NURBS models, CC solids are not restricted to tensor product topologies and therefore support complex geometry without the need for trimming. This allows for representing complex 3D objects as a single subdivision volume, inherently providing C^2 continuity between the individual CC-solid cells away from extraordinary configurations. Trimming cells or stitching multiple subdivision volumes is not needed. However, analytical shapes such as cylinders, spheres, or cones are only approximated by the Catmull-Clark subdivision scheme. Although Burkhart et al.'s research on IGA with CC solids showed promising results, they only provided a proof of concept with a basic volumetric limit evaluation method and a straight-forward numerical integration. The influence of sub-optimal mesh quality on the simulation also remains to be investigated.

In the last years, additive manufacturing (AM) introduced new design freedom to designers and engineers. AM allows for manufacturing objects that cannot be built with other manufacturing methods such as milling, turning or casting. Additively manufactured parts can have complex inner structures, including cavities and lattices, or can consist of multiple materials with different material properties, without the need of assembling multiple individually manufactured components. These new possibilities also made AM leverage topology optimization. Originally presented in the 1980s by Bendsøe [Ben89], a wide range of topology optimized designs are now manufacturable using modern AM processes. In order to support and exploit this widened design space, new software tools, but more fundamentally, new digital representation schemes as well as new 3D modeling concepts are required.

1.1. Research Questions

In my thesis, I investigate if and how subdivision volumes, i.e. Catmull-Clark solids, can be a suitable representation scheme for integrating geometric modeling and physically based simulation more tightly.

As with subdivision surface methods, efficient limit evaluation is a core requirement when working with subdivision volumes such as CC solids – for geometric modeling as well as for simulation. However, limit evaluation for CC solids is much more complex than for CC surfaces as a lot more topological combinations of irregularities exist. Liu et al. present a theoretical description of all possible configurations [Liu+18]. Up to now, only cells in a regular neighborhood can be evaluated directly. Using the algorithm by Burkhart et al. [BHU10b], direct evaluation only works for a small set of topologies and a limited set of parameter values (u, v, w). For all other cases, iterative subdivision has to be used or Stam's constant-time surface evaluation approach [Sta98a] has to be applied multiple times. This is the motivation for the first research question:

RQ1: How can limit evaluation for CC solids be performed more efficiently?

Another core aspect of integrating geometric modeling and physically based simulation with CC solids is the ability to define and modify volumetric control meshes. Although many 3D modeling tools support the creation and modification of subdivision surfaces, to the best of my knowledge, no 3D modeling environment for Catmull-Clark solids or for volumetric subdivision models in general has been presented prior to this dissertation. This leads to the second research question:

RQ2: How can volumetric control meshes be created and modified?

Even though subdivision-based IGA is promising, the FEA community offers a wide range of mature and well-evaluated methods and functionality, including different kinds of boundary conditions, efficient massively parallel FEM solvers, as well as extensive post-processing features. Still, the process of mesh generation is a bottleneck – especially in optimization loops, where the geometry changes in every iteration. Given their volumetric nature, CC solids can be utilized to simplify and speed up the mesh generation process for tetrahedral meshes. This allows for combining the subdivision-based representation of the geometry with fully matured and commercially available FEA frameworks, while at the same time increasing the performance and stability of mesh generation. Therefore, the third research question is:

RQ3: How can the volumetric nature of CC solids be exploited when generating tetrahedral meshes?

Burkhart et al. have presented a first approach for IGA on CC solids [BHU10b]. However, the following two important aspects have not been addressed in their work: Firstly, numerical integration of Catmull-Clark solids has only been discussed briefly. As pointed out by Nguyen et al. [NKP14], standard integration schemes lead to erroneous results for irregular CC-surface patches. The same applies for irregular CC-solid elements, resulting in erroneous entries in the final system of linear equations and therefore reducing the accuracy of the simulation results. Secondly, effects related to the quality of the (u, v, w)

parametrization – a good parametrization is required for properly capturing the physical phenomena in the basis functions – have not been investigated. My experiments have shown that Catmull-Clark solids suffer from distortions and singularities especially at the boundary. As investigating these two points would considerably improve the maturity of IGA based on CC solids, the final two research question are:

RQ4: How can numerical integration be improved for irregular CC-solid cells?

RQ5: How can parameterization problems be identified, quantified, and avoided if possible?

1.2. Contributions

Motivated by the five research questions stated above, my research provides the following contributions:

C1: Constant-Time Volumetric Limit Evaluation

I investigated efficient limit evaluation for Catmull-Clark solids. While Burkhart et al. presented a modified edge rule [BHU10b] for Joy's and MacCracken's original subdivision scheme [JM99], I developed a modified vertex rule in order to achieve a tensor product property of the basis functions for extraordinary edges (EEs) and extraordinary vertices (EVs). Based on these improved subdivision rules, I present the first constant-time limit evaluation method for CC solids. It comprises:

- a new direct evaluation technique for irregular layered structures with one EE,
- a new direct evaluation technique for regular and layered boundary cells, as well as
- a consistent numbering scheme for evaluating the limit of irregular cells containing multiple EEs.

This results in constant-time limit evaluation for irregular structures with one or multiple EEs of arbitrary valence. The approach is faster than the subdivision-based evaluation approach by Burkhart et al. [BHU10b] for every topology and every parameter point (u, v, w) that – up to now – required more than two local subdivision steps.

C2: Interactive Volumetric Modeling

Based on a half-face data structure similar to *OpenVolumeMesh* [KBK13], I developed an interactive 3D modeling environment for volumetric subdivision control meshes, i.e. for CC-solid models. This volumetric modeling environment includes:

- extensions to the standard Euler operators for creating and modifying volumetric control meshes,
- parametrized modifications of control points,
- handling of inner control points when modifying the outer surface, as well as
- the ability to encode additional attributes in the subdivision volume, defined by the same basis functions as the geometry.

All CC-solid models that are used throughout my dissertation were created using this environment.

C3: Efficient and Consistent Generation of Tetrahedral Meshes

I present an efficient and consistent tetrahedral mesh generation approach based on CC solids, bridging the gap to the FEA community. The method:

- utilizes the inherent volumetric structure of CC solids,
- creates precomputable mesh generation matrices derived from modified subdivision rules,
- allows for efficiently updating the tetrahedral mesh when the control mesh is modified, and
- provides a relation between the geometric model and the simulation results.

When changing the topology of the control mesh, my mesh generation approach is up to $30\times$ faster than creating the tetrahedral mesh based on the outer surface (i.e. using the *CGAL* library [Jam+15]). When modifying the geometry of the control mesh, the precomputability of the mesh generation matrices can be exploited, allowing for an almost instant update of the tetrahedral mesh. However, regarding common quality measures for finite element meshes such as the ones presented by Shewchuk [She02] and Field [Fie00], the resulting tetrahedral meshes have lower mesh quality than those created with tetrahedral meshing algorithms that start from a surface model.

C4: Improved Numerical Integration for Irregular CC-Solid Cells

I present an improved numerical integration approach for Catmull-Clark solids, i.e. for irregular cells, including:

- a hierarchical 2D quadrature for integrating over irregular surface patches at the boundary,
- a 3D tensor-product quadrature for integrating over CC-solid cells with one extraordinary edge, as well as
- a hierarchical 3D quadrature for integrating over irregular cells with multiple EEs.

Using this integration technique instead of standard Gauss-Legendre quadrature as proposed by Burkhart et al. [BHU10b], reduces the integration error by up to three orders of magnitude for computing limit surface areas and by up to two orders of magnitude for calculating element stiffnesses.

C5: Quality Metric and Improvements for the Parametrization of the Limit Volume

I introduce a parametrization quality metric for assessing local distortions in the parametrization of a CC-solid model. The metric:

- measures distortions in the (u, v, w) parametrization of CC-solid cells in terms of non-uniform scaling and shearing, and
- reveals singularities and vanishing derivatives in the Catmull-Clark limit volume.

In order to improve the parametrization quality at the boundaries of CC-solid models, I present a set of local split operations for different kinds of boundary cells. Applying these splits improves the average parameterization quality value in boundary cells by $25.5\times$ to $360\times$, and most importantly, resolves the singularities that are inherently present at the boundary of CC-solid models.

Summary

Having a 3D modeling environment with the ability to create, modify, and define parameters on CC-solid models, together with the physically based simulation on CC solids, finally enables exploiting the potential of IGA based on CC solids. My research delivers an environment based on CC solids for generating 3D models suited for design and simulation alike, as well as a tighter integration of geometric modeling and physically based simulation with shorter iteration times in the product development process and more insightful feedback from simulation results.

To achieve such integration, I present two ways of bridging the gap between geometric modeling and simulation. The first is to combine the volumetric representation of CC solids with the well-developed world of tetrahedral finite element methods. The second is to follow the IGA concept and perform the simulation directly on the CC-solid representation using its subdivision basis functions. As shown in this dissertation, both ways have their advantages over the traditional approach of using CAD and FEM and prove valid for enhancing the product development process.

In addition to the theoretical advances, I demonstrate the practical value of my research results in the following four applications:

1. Interactive integrated modeling and simulation of 3D solid objects
2. Interactive design space exploration with simulation-based user guidance
3. Modeling of smoothly graded volumetric material properties for multi-material additive manufacturing
4. Modeling of cavities and inner structures for additive manufacturing parts

1.3. Structure

The remaining part of this dissertation is structured as follows: Chapter 2 provides background knowledge on concepts and methods used in this dissertation and gives an overview about existing research in the individual areas covered by and relevant for my work. Chapter 3 describes the foundation of my research. It presents Catmull-Clark solids as the volumetric representation my work is based on, the novel constant-time limit evaluation method (**C1**), as well as the volumetric modeling environment used for interactively creating and modifying volumetric subdivision control meshes (**C2**). Chapter 4 presents the efficient tetrahedral mesh generation approach based on CC solids (**C3**), which resembles a bridge between my work and existing methods in the field of FEA. Chapter 5 describes my work on Isogeometric Analysis with Catmull-Clark solids and presents the hierarchical numerical integration approach (**C4**) as well as the parametrization quality metric and the related splitting operations (**C5**). Chapter 6 presents four practical applications from different areas, each application containing one or multiple of my concepts and methods. Chapter 7 concludes this dissertation. I discuss the contributions and benefits as well as the limitations of my work and present open points for future investigation. A list of my published research papers as well as a list of my supervising activities can be found in Appendix A and in Appendix B, respectively.

2. Background and Related Work

This chapter explains the concepts and techniques, my research is based on. The chapter covers the basics of continuous and discrete representations for surfaces and volumetric models and explains the concepts of Finite Element Analysis (FEA) and Isogeometric Analysis (IGA). It also presents similar as well as alternative or complementary work by other researchers. As my research is built around subdivision algorithms, subdivision surfaces as well as subdivision solids are presented in more detail.

Section 2.1 presents the surface representations related to this dissertation: B-splines, Non-Uniform Rational B-splines (NURBS), and subdivision surfaces. Section 2.2 presents discrete and continuous volumetric representations, i.e. tetrahedral and hexahedral meshes, spline volumes, and subdivision solids. Section 2.3 presents the concepts of physically based simulation, focusing on FEA on tetrahedral meshes, as well as on IGA based on splines and subdivision models. Section 2.4 summarizes the shortcomings of existing approaches for integrating geometric modeling and physically based simulation and highlights the need for action.

2.1. Surface Representations and Geometric Modeling

For representing three-dimensional objects, the main surface representations are:

- Discrete surface meshes, such as triangular or quadrilateral meshes
- Spline surfaces
- Subdivision surfaces

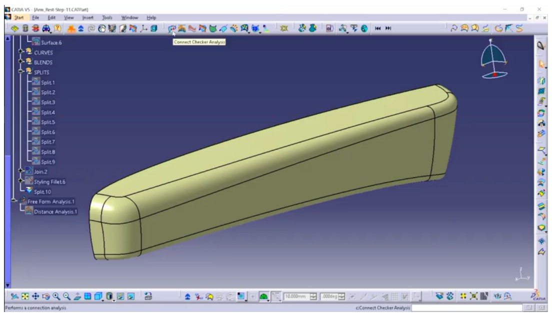
In the field of 3D visualization (rendering), as well as for FEA on thin-shell objects and for additive manufacturing, 3D objects are most often represented by discrete triangular or quadrilateral meshes. Spline surfaces and subdivision surfaces can be discretized into triangular meshes by applying *tessellation*, *triangular meshing*, or *quad meshing*. Multiple methods exist for performing such conversions. However, discrete surface meshes are out of scope for this dissertation.

Continuous spline surfaces are used in the field of *Computer-Aided Design* (CAD) and can be modeled with common CAD tools, such as SolidWorks [Das20a], Rhino [MA18] or Fusion 360 [Aut20]. Subdivision surfaces are heavily used in the computer graphics and computer animation community and are available in polygonal modeling tools such as Maya [Aut18] or Blender [Fou20].

2.1.1. Spline Surfaces

Spline surfaces such as B-spline or NURBS surfaces define smooth surface geometry by a set of control points and are well suited for engineering applications since analytic shapes such as cylinders and cones can be described exactly. Spline surfaces are most commonly used in CAD software for representing thin-shell and solid objects alike. Having a set of closed surfaces, a given CAD model is described with a so-called *Boundary Representation* (B-Rep). CAD tools provide a large variety of modeling operations for spline geometry, such as 2D and 3D sketching, sweeping, and lofting. Figure 2.1 shows examples of spline-based surface models in the CAD systems CATIA [Das20b] and SolidWorks [Das20a].

However, B-spline or NURBS surface patches require a tensor product parametrization, which only allows for simple shapes to be represented by a single spline surface. Complex


(a) NURBS surface modeling in CATIA. Image taken from [Mar19].

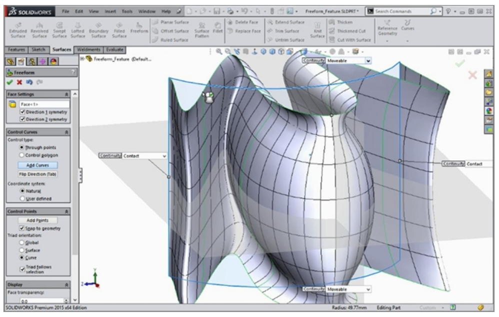

(b) NURBS surface modeling in SolidWorks. Image taken from [Gav15].

Figure 2.1.: Examples of NURBS surface models in CATIA (a) and SolidWorks (b).

topologies require either trimming the tensor product model or stitching of multiple surfaces to handle irregularities. Unless spending additional computational effort, connecting multiple spline surfaces results in C^0 continuity between them. Trimming neighboring patches with a shared trimming curve will result in gaps, as the trimmed boundary cannot be represented identically by both patches. Detailed information, as well as the mathematical backgrounds and important algorithms can be found in the textbook by Piegl and Tiller [PT12].

Although B-splines and NURBS are the most common spline variants, other types of splines have been presented over the years, such as *T-splines* [Sed+03] and *Locally Refinable (LR) B-splines* [DLP13]. They weaken the tensor-product requirement of the control mesh and allow for local refinement. T-splines are available in commercial software, e.g. in the 3D modeling tool Rhino [MA18].

In the following, I present the concept of basis functions and how they define the shape of a spline curve or surface using the example of uniform cubic B-splines. The concept can be transferred to other types of splines and also to subdivision surfaces such as Catmull-Clark surfaces (see Section 2.1.2).

B-Spline Basis Functions

The shape of a B-spline curve of degree d is defined by its n control points p_i multiplied with the corresponding basis functions N_i^d. The basis functions N_i^d are polynomial functions, the degree of which is equal to the degree d of the curve. Every segment of the curve is defined by $d+1$ consecutive control points, $d+1$ basis functions N_i^d and the local parameter $u \in [0,1]$. Neighboring segments share d control points and each control point influences the shape of $d+1$ consecutive segments. For cubic B-splines, the basis functions are defined as follows:

$$\begin{aligned} N_0^3(u) &= \frac{1}{6}(1-u)^3 \\ N_1^3(u) &= \frac{1}{6}\left(3u^3 - 6u^2 + 4\right) \\ N_2^3(u) &= \frac{1}{6}\left(-3u^3 + 3u^2 + 3u + 1\right) \\ N_3^3(u) &= \frac{1}{6}u^3 \end{aligned} \tag{2.1}$$

The basis functions are visualized in Figure 2.2.

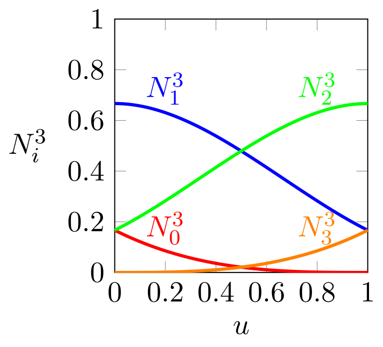


Figure 2.2.: Cubic B-spline basis functions N_i^3. Each of the four basis functions describes the influence of one of the four control points.

A point $c(u)$ in a segment of the curve is defined as follows:

$$c(u) = \sum_{i=0}^{d} p_i N_i^d(u) \tag{2.2}$$

Analogously, a single B-spline surface patch of degree d is defined by a grid of $(d+1)\times(d+1)$ control points and $(d+1)^2$ bivariate basis functions. These basis functions are calculated by multiplying two univariate basis functions, parametrized by two parameters u and v, both $\in [0,1]$. Bicubic B-spline patches feature the following 16 basis functions:

$$N_i^3(u,v) = N_{\lfloor i/4 \rfloor}^3(u)\; N_{i \bmod 4}^3(v) \tag{2.3}$$

Given the basis functions, any point $s(u,v)$ on the patch can be calculated by summing all control points weighted by the factor of the corresponding basis function:

$$s(u,v) = \sum_{i=0}^{n-1} p_i N_i^d(u,v) \tag{2.4}$$

This can also be expressed as a matrix-vector multiplication where $\mathbf{C}$ contains the control points p_i and $\mathbf{N}$ contains the basis functions N_i^3:

$$\begin{gathered}\mathbf{C} = \begin{pmatrix} p_0 & \cdots & p_{n-1} \end{pmatrix}^T, \mathbf{N} = \begin{pmatrix} N_0^3 & \cdots & N_{n-1}^3 \end{pmatrix}^T \\ s(u,v) = \mathbf{C}^T \mathbf{N}(u,v) \end{gathered} \tag{2.5}$$

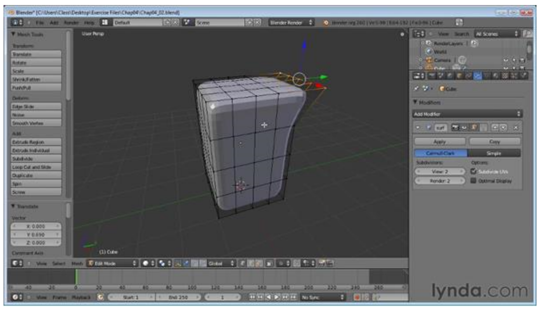
These concepts are also used by Catmull-Clark subdivision surfaces and by Catmull-Clark solids, which are the foundation of this dissertation.

2.1.2. Subdivision Surfaces

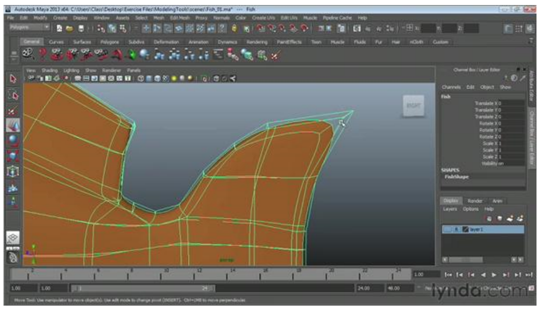
Subdivision surface representations have established as a standard in computer graphics and computer animation and are gaining increasing acceptance in industrial surface design. An example for heavily using subdivision techniques in different areas of 3D modeling and computer animation is the company Pixar [Stu20].

Similar to B-splines or NURBS, subdivision algorithms represent smooth geometry based on a control mesh with a relatively low amount of degrees of freedom. The principal concept of subdivision algorithms is the generation of a smooth manifold by iteratively refining a coarse base mesh. With every subdivision step, the mesh becomes finer and the surface smoother, as it converges against the so-called *limit surface*. For most schemes, a direct mathematical description exists for that limit. Subdivision schemes can be distinguished into approximating schemes – i.e. schemes where the control points are not part of the limit – and interpolating ones (the control points are part of the limit).

Polygonal modeling tools such as Blender [Fou20] and Maya [Aut18] offer subdivision surface algorithms to design organic 3D models. Subdivision models can also be generated by sketch-based approaches as shown e.g. by Bein et al. [Bei+09] and by Ji et al. [JLW10]. Figure 2.3 shows the modeling of subdivision surfaces in Blender [Fou20] and in Maya [Aut18].



(a) Surface modeling in Blender. Image taken from [Mae17].



(b) Surface modeling in Maya. Image taken from [Mae12].

Figure 2.3.: Two examples for polygonal modeling of subdivision surfaces in the 3D modeling tools Blender (a) and Maya (b).

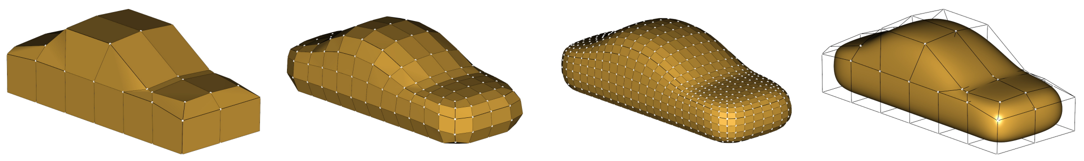

(a) Control mesh (b) 1 level of subdivision (c) 2 levels of subdivision (d) Limit surface

Figure 2.4.: Refinement process of a Catmull-Clark subdivision surface from the control mesh (a) to the limit surface (d).

In 1974, Chaikin presented an algorithm for creating smooth curves out of arbitrary polylines by iterative refinement [Cha74]. The limit curve of Chaikin's subdivision scheme is a quadratic B-spline curve. In 1978, Doo and Sabin as well as Catmull and Clark adopted this approach to create smooth surfaces [DS78; CC78] – Catmull-Clark (CC) subdivision surfaces being a generalization of cubic B-spline surfaces. While some subdivision schemes require a control mesh with a certain topology, e.g. purely triangular meshes for the scheme presented by Loop [Loo87], others can operate on control meshes with arbitrary topology (such as the schemes by Doo and Sabin [DS78] and Catmull and Clark [CC78]). Figure 2.4 shows the application of Catmull-Clark surface subdivision on an abstract model of a car body.

In contrast to tensor product surfaces, subdivision surfaces, such as the methods presented by Catmull-Clark [CC78] or Loop [Loo87], are able to represent complex topologies including irregular, non-tensor-product configurations, as a single topological entity – a single subdivision surface. Depending on the subdivision scheme and the topology of the control mesh, the limit surface has different continuity C^i. However, for most schemes, the continuity is reduced around irregularities.

In order to overcome problems with extraordinary vertices, several subdivision surface schemes have been improved and extended, e.g. with the approaches by Zorin et al. [ZSS96], Huang and Schröder [HS10], and Li et al. [LFS16]. NURBS-compatible schemes such as the one by Cashman et al. [Cas+09] aim at bridging the gap to spline-based representations.

In the following, Catmull-Clark surface subdivision will be presented in detail. Since the techniques and approaches presented in this dissertation are based on Catmull-Clark solids, it is important to first understand the concepts of Catmull-Clark surfaces.

Catmull-Clark Subdivision Surfaces

Catmull and Clark observed that bicubic B-spline knot insertion is equivalent to subdividing each surface patch into four sub-patches. They generalized this subdivision process for irregular meshes, resulting in subdivision rules that adapt to the topology of the control mesh. For formulating the rules, the control mesh is regarded as a set of vertices, edges and faces, where each edge is defined by its two vertices and each face is defined by its loop of edges. The resulting points of the refined mesh are calculated as a weighted sum of the respective surrounding control points according to the subdivision rules. The Catmull-Clark surface subdivision rules [CC78] are:

1. For each face, add a *face point* F as the average of its corner vertices.
2. For each edge, add an *edge point* E as the average of its two vertices and the two neighboring face points.
3. Update the position of each original vertex as a weighted sum of its original position V and the averages of the adjacent face midpoints F_{avg} and edge midpoints E_{avg}, respectively.
$$V_{new} = \frac{F_{avg} + 2E_{avg} + (\mathrm{N} - 3)V}{\mathrm{N}} \tag{2.6}$$
N denotes the valence of the vertex.
4. Connect each face point to its surrounding edge points.

Limit Evaluation

Most subdivision algorithms converge to a smooth *limit* for a conceptually infinite number of subdivision steps. This limit can also be evaluated mathematically. Since performing a high number of subdivision steps to approximate the limit geometry is computationally expensive, algorithms for directly evaluating the limit provide a substantial performance benefit.

Stam presented a ground-breaking constant-time limit evaluation technique for Catmull-Clark, as well as for Loop subdivision surfaces [Sta98a; Sta98b]. Zorin and Kristjansson developed an extended version of Stam's evaluation by incorporating piecewise smooth surfaces and boundaries [ZK02] using Biermann's crease rules [BLZ00]. As an alternative, Bolz and Schröder presented a data-driven evaluation approach for subdivision surfaces that relies on large lookup tables, containing precomputed basis functions for regular, irregular and crease configurations [BS02]. Settgast et al. showed how to efficiently generate accurate subdivision geometry for rendering, using adaptive tessellation [Set+04].

However, subdivision surfaces (as well as subdivision solids) struggle with so-called *extraordinary vertices* (EVs). The valence of a regular vertex is defined by the subdivision scheme itself, e.g. a valence of 4 for Catmull-Clark surfaces [CC78] or 6 for the Butterfly scheme [DLG90]. All vertices with a different valence are defined as EVs. Although most subdivision schemes define a limit surface/volume for these vertices, the continuity is reduced around EVs (e.g. C^1 or even C^0 instead of C^2) and the mathematical evaluation procedure is more complex.

Being able to efficiently evaluate both, the limit geometry and also the derivatives at any given point of a subdivision model is a crucial building block for all subdivision-based simulation approaches. However, integration near extraordinary vertices is problematic for many subdivision schemes. Independently, Wawrzinek and Polthier [WP16], as well as Barendrecht et al. [BBK18] worked on improving integration on Catmull-Clark surfaces for simulation.

In the following, Stam's limit evaluation technique for CC surfaces is presented in detail. Understanding Stam's limit evaluation approach is essential for understanding my constant-time volumetric limit evaluation in Section 3.2. Concepts such as the partitioning of the parameter space, Stam's numbering scheme, the *eigenstructure*, as well as the usage of picking matrices have been adapted and extended in my approach.

Stam's Constant-Time Limit Evaluation for CC Surfaces

The conceptual idea behind subdivision algorithms is to subdivide a base mesh a certain number of times until the resulting surface is perceived as smooth. However, many applications require the evaluation of specific points on the surface at arbitrary parameter values (u, v). Such applications include physically based simulation and efficient data representation, e.g., for scalar fields. Parametric models, such as B-spline surfaces, can be evaluated directly, as the surface is given as a polynomial function. This used to be a significant advantage of spline-based models over subdivision models until Stam devised his method [Sta98a] that allows for direct evaluation of Catmull-Clark subdivision surfaces in constant time for any parameter point (u, v).

Note that before Stam's method can be applied to a given CC-surface control mesh, each face is required to be quadrilateral and to contain at most one extraordinary vertex. However, both requirements are fulfilled after at most two subdivision steps.

Stam's method exploits the fact that Catmull-Clark surfaces are equivalent to cubic B-spline surfaces for regular topologies, which can be evaluated directly with uniform cubic

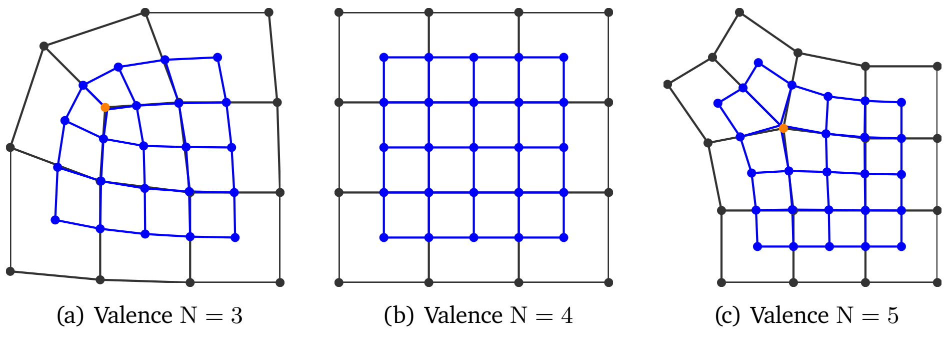

(a) Valence $\mathrm{N} = 3$
(b) Valence $\mathrm{N} = 4$
(c) Valence $\mathrm{N} = 5$

Figure 2.5.: Local control meshes (black) and their corresponding subdivided local control meshes (blue) for valences $\mathrm{N} = 3$ (a), $\mathrm{N} = 4$ (b) and $\mathrm{N} = 5$ (c). The EVs are marked in orange in (a) and (c). As a valence of 4 represents the regular case, there is no EV in (b).

B-spline basis functions. Regular topologies form grids of 4×4 control points as shown in Figure 2.5(b). Topologies that contain *extraordinary vertices* (EVs), i.e. vertices with a valence other than $\mathrm{N} = 4$, as in Figures 2.5(a) and (c), have to be treated differently. As the area affected by an EV shrinks with every subdivision step, the parameter point (u, v) to be evaluated lies inside a regular patch after a certain number of subdivisions.

Since only patches with EVs have to be subdivided for the evaluation, the required subdivision steps are performed locally on each individual patch. This means that only control points that define the current surface patch are considered. As can be seen in Figure 2.5, the topology of the local control mesh does not change during subdivision. This is a crucial requirement for the method presented by Stam. At the same time, the subdivided control mesh (shown in blue in Figure 2.5) describes the same surface patch as the original mesh (shown in black).

For every step of local subdivision, the patch is divided into four sub-patches. Three sub-patches – numbered with $k = 1, 2, 3$ – are defined by a regular 4×4 grid, as shown in dark blue in Figure 2.6. Being defined by a regular control grid, these three sub-patches can be evaluated directly using the uniform cubic B-spline basis functions. The fourth sub-patch $k = 4$ still contains the extraordinary vertex and thus remains irregular. If the parameter point (u, v) to be evaluated lies inside $k = 4$, the irregular sub-patch is subdivided further, again, partitioning it into four new sub-patches.

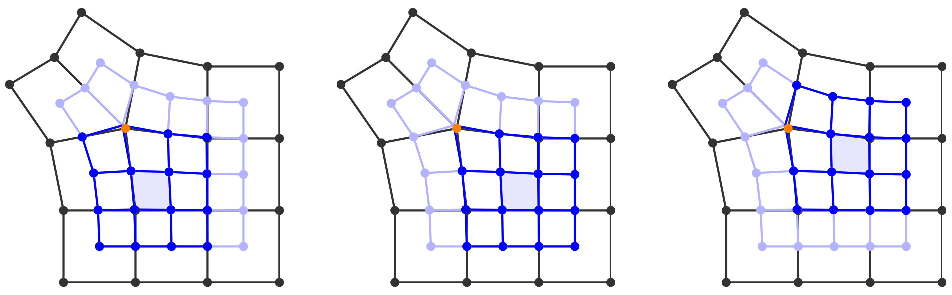

(a) Sub-patch $k = 1$ (b) Sub-patch $k = 2$ (c) Sub-patch $k = 3$

Figure 2.6.: Three regular sub-patches $k = 1$, $k = 2$ and $k = 3$ of an irregular local control mesh of valence $\mathrm{N} = 5$. The control mesh is shown in black. Each sub-patch is defined by a 4×4 grid of subdivided control points (dark blue). The rest of the subdivided mesh is shown in light blue. The extraordinary vertex is marked in orange.

Figure 2.7 illustrates this effect of local subdivisions. The original patch is parametrized by its local coordinates $u, v \in [0, 1]$, with the EV being located at $(0, 0)$. The three sub-patches with $u \geq 0.5$ or $v \geq 0.5$ correspond to the three regular patches, denoted with $k = 1$, $k = 2$ and $k = 3$.

With the help of this representation, the number of local subdivisions n required for a given point (u, v) to lie inside a regular patch can be determined using the fact that the parameter space is bisected with each subdivision:

$$n = \lceil \min \{ -\log_2 (u) \,, \; -\log_2 (v) \} \rceil \tag{2.7}$$

In order to evaluate a parameter point inside a given sub-patch, u and v need to be mapped onto the parameter space (unit square) of that particular sub-patch with the transformation $\phi_{k,n}$:

$$\begin{aligned} k = 1: \quad & \phi_{1,n}(u, v) = (2^n u - 1, \quad 2^n v) \\ k = 2: \quad & \phi_{2,n}(u, v) = (2^n u - 1, \quad 2^n v - 1) \\ k = 3: \quad & \phi_{3,n}(u, v) = (2^n u, \quad 2^n v - 1) \end{aligned} \tag{2.8}$$

Due to the rapid increase of n as u and v approach 0, explicit local subdivision becomes inefficient for small u and v. Hence, Stam's method subdivides each patch locally using the *eigenstructure* of the local *subdivision matrix*.

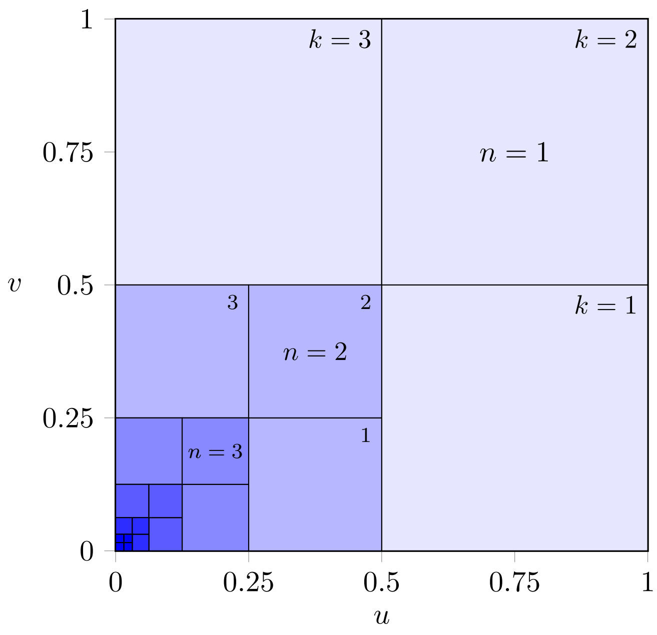


Figure 2.7.: Partitioning of the parameter space for different subdivision levels n. k represents the local numbering of the sub-patches created in each subdivision step. The extraordinary vertex corresponds to $(u, v) = (0, 0)$

Performing a local subdivision by means of matrix multiplication can be expressed as a multiplication of the initial control points C_0 and the subdivision matrix $\bar{\mathbf{A}}$:

$$C_1 = \bar{\mathbf{A}} C_0, \tag{2.9}$$

C_1 denotes the subdivided points. In order to re-use $\bar{\mathbf{A}}$ during multiple subdivision steps, the order of the points in $\bar{\mathbf{A}}$ and C_0 has to be defined consistently.

As shown in Figure 2.8, the vertices are numbered in a specific pattern, depending on the valence N, so that the indices can be used universally for every valence. Furthermore, with this particular ordering, multiple subdivision steps can be performed by extracting the top $K = 2\mathrm{N} + 8$ rows from $\bar{\mathbf{A}}$. This results in a $(K \times K)$ matrix, denoted in the following with $\mathbf{A}$. Since the topology of the control mesh stays consistent with each subdivision as mentioned earlier, $\mathbf{A}$ can be pre-multiplied an arbitrary number of times for performing multiple subdivision steps. Thus, n levels of subdivision can be calculated as follows:

$$C_n = \bar{\mathbf{A}}\mathbf{A} \cdots \mathbf{A} C_0 = \bar{\mathbf{A}}\mathbf{A}^{n-1} C_0 \tag{2.10}$$

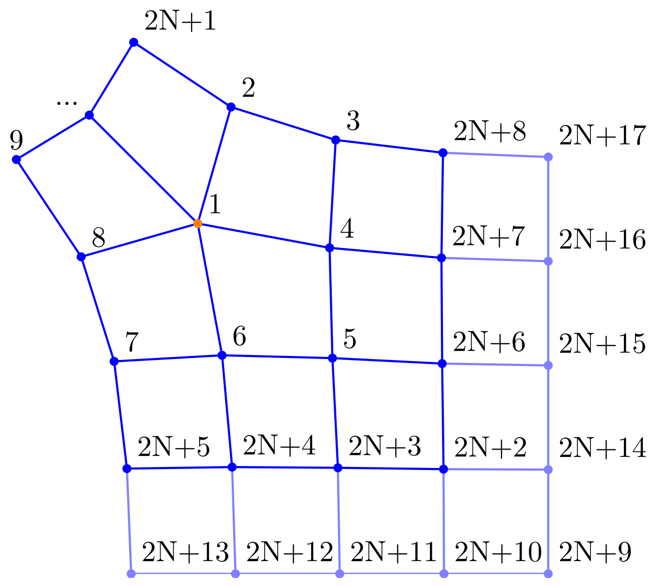


Figure 2.8.: Ordering of control points. The initial vertices (blue) are numbered $1, \ldots, 2\mathrm{N}+8$ and the vertices added by local subdivision (light blue) are numbered $2\mathrm{N}+9, \ldots, 2\mathrm{N}+17$. The EV is marked in orange.

In order to compute the power of the squared subdivision matrix $\mathbf{A}$ more efficiently, the eigenstructure of $\mathbf{A}$ is computed. The eigenstructure consists of the matrix $\mathbf{V}$ containing the eigenvectors of $\mathbf{A}$ as column vectors and the diagonal matrix Λ containing the eigenvalues of $\mathbf{A}$:

$$\mathbf{AV} = \mathbf{V}\Lambda \quad \Leftrightarrow \quad \mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}, \tag{2.11}$$

Inserting the eigenstructure into (2.10) leads to a formulation that only involves a limited number of matrix-matrix multiplications and the exponentiation of a diagonal matrix for performing n local subdivision steps:

$$C_n = \bar{\mathbf{A}}\left(\mathbf{V}\Lambda\mathbf{V}^{-1}\right)^{n-1} C_0 = \bar{\mathbf{A}}\mathbf{V}\Lambda^{n-1}\mathbf{V}^{-1}C_0, \tag{2.12}$$

This allows to perform multiple local subdivisions in constant time.

The analytical calculation of the eigenstructure is detailed in the work by Stam [Sta98a]. However, since later additions to this method require the calculation of subdivision matrices with no predefined structure, I compute the eigenstructure numerically using built-in functionalities of the *Eigen* library [JG08]. This guarantees the necessary versatility to support crease edges as well as multiple extraordinary edges for volumetric limit evaluation, as described in Section 3.2. Although the numerical computation of the eigenstructure introduces a deviation from the analytical calculation, no difference could be observed in my experiments when applying the eigenstructures.

After the required number of subdivisions n have been performed, the resulting regular sub-patches can be evaluated using the bivariate uniform cubic B-spline basis functions. The control points $C_{n,k}$ that define the corresponding sub-patch can be extracted from C_n using a picking matrix $\mathbf{P}_k$. Finally, the limit point $\mathbf{e}$ can be evaluated by multiplying the control points $C_{n,k}$ with the corresponding coefficients $\mathbf{N}\left(u,v\right)$ of the bivariate cubic B-spline basis functions. With Equation (2.12), the evaluation can be written as follows:

$$\begin{aligned}\mathbf{e}\left(u,v\right) &= C_0^T\mathbf{V}^{-T}\Lambda^{n-1}\mathbf{V}^T\bar{\mathbf{A}}^T\mathbf{P}_k^T\mathbf{N}\left(\phi_{k,n}\left(u,v\right)\right)\\ &=: C_0^T\mathbf{V}^{-T}\hat{\varphi}\left(u,v\right),\\ &=: C_0^T\varphi\left(u,v\right),\end{aligned} \tag{2.13}$$

$\varphi\left(u,v\right)$ denotes a vector containing the basis functions corresponding to the control points in C_0. I call these functions the *bivariate subdivision basis functions*. $\hat{\varphi}\left(u,v\right)$ are the *eigenbasis functions* as presented by Stam [Sta98a].

Sharp Features and Crease Edges

As stated above, standard subdivision rules (e.g. the rules by Catmull and Clark [CC78]) generate a smooth surface from a coarse base mesh. In real-world applications, geometric models almost always include sharp features. Therefore, it may be desired to define sharp edges or creases on parts of the model. One way for modeling such features is to define specific edges as *crease edges*.

In order to create sharp features on otherwise smooth subdivision surfaces, Hoppe et al. presented modified subdivision rules that include crease edges [Hop+94]. Those rules were later extended, e.g. by DeRose et al. [DKT98] and Biermann et al. [BLZ00] for increased control and flexibility. Schweitzer presented a geometric approach for creases that is based on creating so-called *ghost/phantom points* to obtain sharp features [Sch96b]. This concept was also used by Havemann [Hav05], as well as by Lacewell and Burley [LB07] to define sharp features as well as boundaries on subdivision surfaces. These specialized *crease rules* are detailed in Havemann's work [Hav05]. In contrast to the standard subdivision rules, in each refinement step, the positions of the new edge and vertex points is only influenced by control points along the crease edge.

Most of the publications mentioned above classify vertices depending on how many crease edges are connected to them. The shared nomenclature is:

- No crease edges: *smooth vertex*
- 1 crease edge: *dart vertex*
- 2 crease edges: *crease vertex*
- 3+ crease edges: *corner vertex*

Figure 2.9 shows the effects of different configurations of crease edges on a closed CC-surface model.

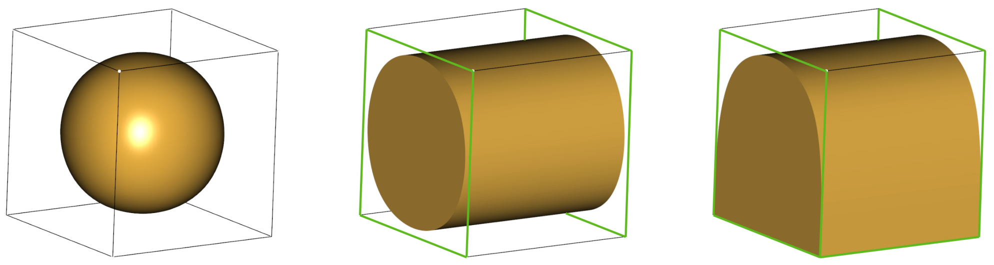

(a) No crease edges (b) Crease edges on the left and right face (c) Crease edges on the left, right, and bottom face

Figure 2.9.: Different arrangements of crease edges (green) and their effect on the limit surface. In (a), no edges are defined as crease edges. The standard subdivision rules apply. In (b), the edges of two opposite faces are defined as crease edges. Here, all vertices are classified as crease vertices. In (c), also the edges of the bottom face are defined as crease edges. The bottom vertices are now classified as corner vertices.

Crease Basis Functions

Crease edges substantially change the shape of a subdivision surface by allowing for non-smooth transitions on the surface, i.e. crease edges reduce the continuity between surface patches. Crease edges also alter the underlying basis functions required for evaluating the limit surface, leading to *crease basis functions*. While Schweitzer provides a first insight on how to derive crease basis functions using ghost points [Sch96b], a detailed analysis of creases and their effect on the underlying basis functions can be found in the work of Kosinka et al. [KSD14]. In the following, the concepts behind crease basis functions are explained in detail. Crease basis functions are used for evaluating the limit of CC-solid boundary cells in my constant-time volumetric limit evaluation approach (see Section 3.2).

The influence of crease edges on the behavior of the surface has been thoroughly examined by Havemann [Hav05]. As already stated by Schweitzer [Sch96b] and also mentioned by Zorin et al. [ZK02], he observes that achieving the modifications induced by a crease edge is possible by inserting ghost points, i.e. by extrapolating the edges attached to two endpoints of the crease edge.

For enabling crease edges without explicitly inserting new control points, the subdivision rules are adapted to obtain the crease rules. In order to keep the topology consistent at each level of subdivision, only one vertex in a regular 4×4 control mesh is allowed to have more than two adjacent crease edges. For irregular patches, only the edges radiating

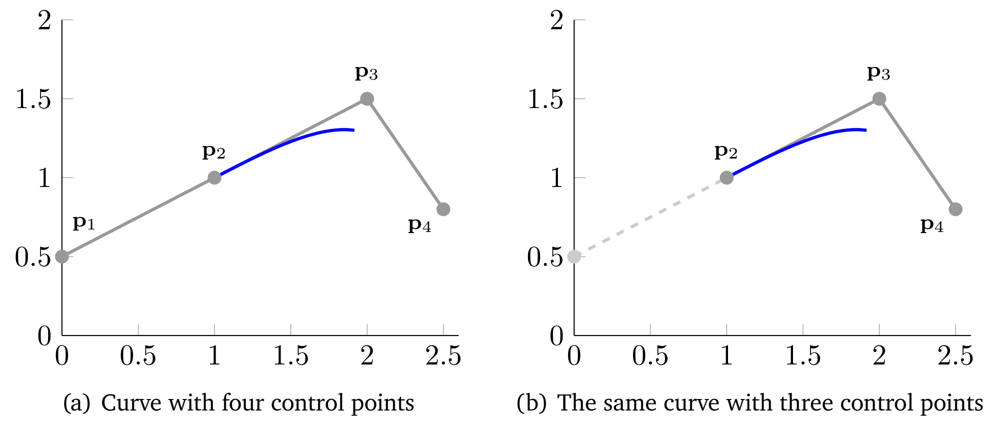


Figure 2.10.: The same B-spline curve defined by four control points and standard basis functions (a) and defined by three points with crease basis functions (b).

from the extraordinary vertex may be defined as crease edges. Again, after one level of Catmull-Clark subdivision, all local control meshes fulfill these requirements.

Figure 2.10(a) depicts a regular cubic B-spline curve with four control points $\mathbf{p}_1, \ldots, \mathbf{p}_4$, of which the first three are collinear and equidistant, such that $\mathbf{p}_1 - \mathbf{p}_2 = \mathbf{p}_2 - \mathbf{p}_3$. Any point $p(u)$ on the curve is defined analytically with the four cubic B-spline basis functions in a column vector as follows:

$$p(u) = \begin{pmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{pmatrix} \cdot \frac{1}{6} \begin{pmatrix} (1-u)^3 \\ 3u^3 - 6u^2 + 4 \\ -3u^3 + 3u^2 + 3u + 1 \\ u^3 \end{pmatrix} \tag{2.14}$$

Replacing $\mathbf{p}_1$ with $2\mathbf{p}_2 - \mathbf{p}_3$ yields a formulation with just three control points:

$$C(u) = \begin{pmatrix} \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \end{pmatrix} \cdot \frac{1}{6} \begin{pmatrix} u^3 - 6u + 6 \\ -2u^3 + 6u \\ u^3 \end{pmatrix} \tag{2.15}$$

With these basis functions, the same curve can be defined with just three control points, as shown in Figure 2.10(b). These three cubic functions in Equation (2.15) are the *crease basis functions* and are depicted in Figure 2.11. They replace the standard basis functions when evaluating the limit surface for Catmull-Clark patches that contain crease edges.

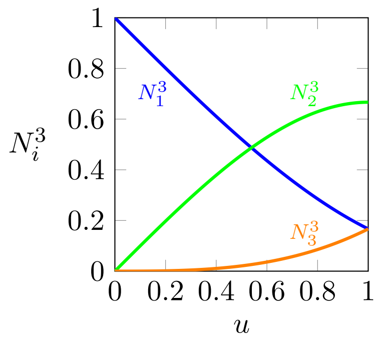


Figure 2.11.: The modified crease basis functions. N_1^3 (blue) and N_2^3 (green) change compared to the standard cubic B-spline basis functions shown in Figure 2.2. N_3^3 (orange) remains unchanged. As N_0^3 (red) vanishes, the first control point does not influence the shape of the curve. Furthermore, the second function N_1^3, i.e. the first crease basis function, starts at 1 for $u = 0$.

2.2. Volumetric Representations

Subdivision surfaces as well as B-spline and NURBS surfaces only describe the shape of a 3D object. Although closed surfaces implicitly represent a volumetric object, they are not considered to be volumetric representations, since they do not contain any information about the interior of the volume they enclose, e.g. varying material properties.

In contrast, volumetric representations provide *nodes* in the interior of the model for storing volumetric information. The most common volumetric representations are:

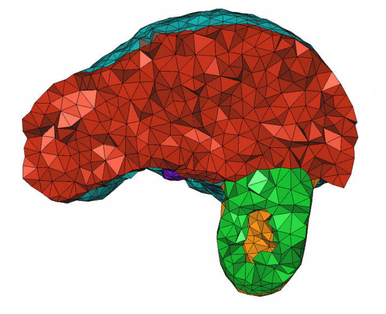
- Regular voxel grids, often used for visualizing medical data, as well as in the context of additive manufacturing
- Discrete volumetric meshes, mainly hexahedral and tetrahedral meshes, used for *Finite Element Analysis* (FEA)
- Trivariate spline models – spline volumes – used for *Isogeometric Analysis* (IGA) and for multi-material additive manufacturing
- Subdivision volumes used for local refinement of discrete FE meshes, but also for IGA and multi-material additive manufacturing as an alternative to trivariate splines

Similar to discrete surface meshes, voxel grids are out of scope for this dissertation. The other three representations are presented in more detail in the following subsections.

2.2.1. Tetrahedral and Hexahedral Meshes

When simulating solid objects, all finite element approaches share the same requirement: a discrete volumetric mesh. Tetrahedral and hexahedral meshes are the most common discrete volumetric representation for FEA. These meshes are built from a set of connected *elements*, i.e. tetrahedra or hexahedra, respectively. Each interior face is shared by two elements while each boundary face only belongs to one element. As tetrahedral or hexahedral meshes are seldom modeled by hand, they are most commonly created from a given B-Rep model with the help of so-called *meshing* algorithms.

Mesh generation or *meshing* describes the process of converting a surface model, e.g. represented as NURBS B-Rep, into a discrete mesh for simulation. For thin-shell structures, e.g. sheet metal parts, discrete surface meshes such as triangular or quad meshes are sufficient. For solid objects, the (continuous) surface representation is transformed into a discrete volumetric representation by performing tetrahedral or hexahedral meshing.


(a) Tetrahedral mesh with three regions of different resolution. Image taken from [All+16].

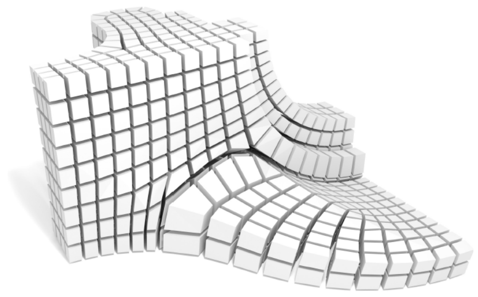

(b) Hexahedral mesh. Image taken from [Liu+18].

Figure 2.12.: Tetrahedral (a) and hexahedral mesh (b). The tetrahedral mesh is partitioned in three regions (shown in red, green and yellow) with different resolutions.

This is a complex and error-prone task, especially for non-watertight surface models, as well as for models with sliver edges or very small features.

In an iterative design and simulation process, (re-)meshing must be performed for every geometric or topological change in the design model, apart from cases where the mesh can be morphed to fit the shape of the modified surface model. However, mesh morphing is only applicable for small modifications. For all other cases, the costs for meshing are multiplied by the number of design changes.

Different approaches exist for converting a B-Rep model into a volumetric mesh. For tetrahedral meshes, Alliez et al. [All+05; Jam+15], Geuzaine and Remacle [GR09], and Si [Si15] present such meshing algorithms. While most meshing algorithms require manifold surface meshes, some methods are less restrictive about their input geometry, such as the approach by Hu et al. [Hu+18]. The mesh resolution has to be chosen adequately according to the complexity of the input geometry. Small features are often removed from the original CAD model before meshing. However, an approximation error always remains. Figure 2.12(a) shows an example tetrahedral mesh generated by CGAL [All+16]

Especially for engineering applications, high mesh quality is an important requirement for the mesh generation process. Shewchuk [She02] as well as Field [Fie00] give good overviews over different aspects of mesh quality and how they can be satisfied. The meshing algorithms mentioned above focus on creating high-quality tetrahedral meshes.

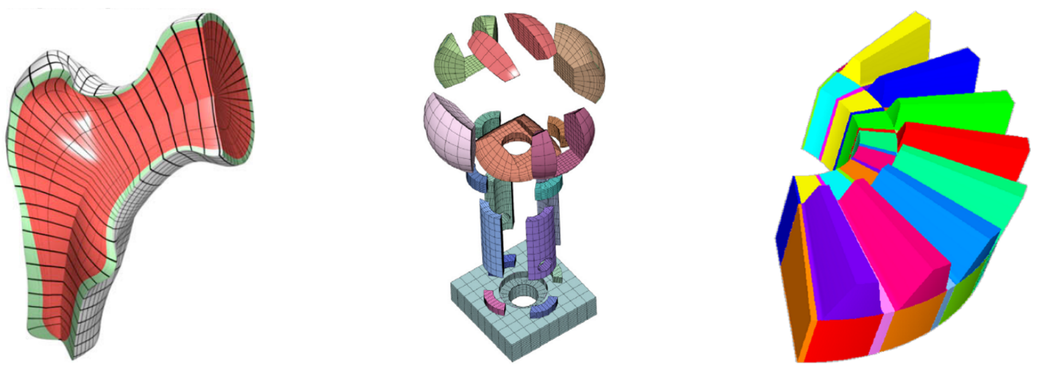
Hexahedral meshes are more difficult to create than tetrahedral meshes due to the additional topological constraints of hexahedra compared to tetrahedra. In recent years, promising results have been shown with novel hexahedral meshing approaches presented by Lyon et al. [LBK16], Solomon et al. [SVB17], Liu et al. [Liu+18], and Corman et al. [CC19]. While some hexahedral meshing approaches start from a discrete surface mesh, most techniques require a tetrahedral mesh as input. The core challenge of hexahedral meshing is to compute the so-called *singularity graph*. This graph determines where irregularities have to be inserted into the mesh in order to generate a valid topology. The resulting mesh consists of a set of regular blocks, bounded by the singular edges. Almost all solid objects require singular edges in order to be meshed into a hexahedral mesh. Figure 2.12(b) shows a hexahedral mesh created with the approach by Liu et al. [Liu+18]. *Hex-dominant* meshing reduces the complexity of the hex meshing problem by allowing tetrahedra, prisms/wedges, and other polyhedra next to hexahedra in the mesh.

2.2.2. Spline Volumes

Trivariate splines – spline volumes – are the volumetric extension of spline surfaces presented in Section 2.1.1. Consisting of cells, in analogy to segments and patches, each cell is parametrized with three parameters u, v, and w, all $\in [0, 1]$. The trivariate basis functions are the tensor product of three univariate basis functions. For a tricubic B-spline volume, 64 basis functions are defined on a grid of $4 \times 4 \times 4$ control points.

Resulting from the tensor product requirement of trivariate splines, control meshes are limited to purely hexahedral meshes with regular topology. Irregularities require tiling, i.e. connecting multiple spline volumes, which limits the continuity between the volumes to C^0 [ME16]. However, as described in Section 2.2.1 and also explained by Solomon et al. [SVB17], as well as by Liu et al. [Liu+18], hexahedral meshes for complex objects have to have irregularities (i.e. singular edges) in order to properly follow the input geometry. Cohen et al. [Coh+10] provide recommendations on how to best represent solid 3D objects with trivariate splines. Even these recommendations contain irregularities.

Alternatively, complex models can also be represented by trimming trivariate splines. Just as spline surfaces can be trimmed with trimming curves, spline volumes can be trimmed with surfaces. Applying the same trimming surface on neighboring spline volumes introduces gaps as the result cannot be represented identically for both volumes. Dokken et al. [DSB19], as well as Massarwi et al. [MAE19] both presented methods for re-parametrizing the trimmed domain and structuring it into a set of untrimmed tensor-product spline volumes for simulation.



(a) Multi-material B-spline volume. Image taken from [Coh+10].

(b) Trivariate B-spline model created from Boolean operations. Image taken from [ME16].

(c) Block-structured trivariate NURBS model. Image taken from [DSB19].

Figure 2.13.: Trivariate B-spline and NURBS models created from different approaches. The object shown in (a) is described by a single B-spline volume. The model shown in (b) is created via Boolean operations on trivariate B-spline primitives. (c) shows a trivariate model created from multiple NURBS volumes.

As for spline surfaces, B-splines and NURBS are the most common variants of spline volumes. Trivariate splines are used in the context of Isogeometric Analysis (see Section 2.3.2), but also in the field of multi-material additive manufacturing as shown by Ezair et al. [EDE17]. Although, Dokken et al. stressed the need for trivariate CAD systems already in 2009 [Dok+09], spline volumes are not supported by common CAD tools. Therefore, spline volumes have to be either converted from existing B-Rep models or created from scratch.

The conversion of a B-Rep model built from spline surfaces into a trivariate spline representation can be seen as a special kind of hexahedral meshing (see also Section 2.2.1). Al Akhras et al. investigated this problem but their approach uses a triangulation of the boundary instead of the original NURBS/B-spline surfaces defined in the CAD system [Al+16]. Another approach has been presented by Wang et al. for creating trivariate T-splines from boundary triangulations [Wan+13]. Lin et al. presented an approach that converts a tetrahedral mesh into a set of trivariate B-splines [Lin+15] – requiring tetrahedral meshing of the B-Rep model in the first place. For all three methods, the resulting trivariate model consists of multiple blocks – with C^0 continuity between them – and does not correlate with the initial NURBS or B-spline surfaces.

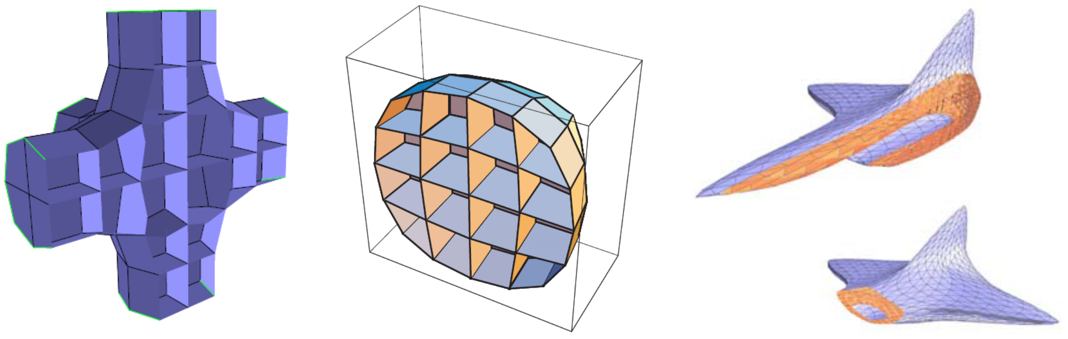
Aigner et al. presented a method for converting NURBS B-Rep models created from sweeping operations, as they often occur in CAD, into trivariate representations [Aig+09]. Although this approach re-uses the original spline surfaces, it only covers a specific class of 3D objects – those that can be modeled by sweeping.

Prototypical volumetric modeling frameworks such as the one presented by Massarwi and Elber [ME16] try to avoid this conversion step by initially designing the object with trivariate NURBS or B-splines. Massarwi's and Elber's modeling environment for B-spline volumes is based on combining primitives such as cubes, spheres, and cylinders with Boolean operations. In this context, they also investigated the compatibility of patches or volumes with different degrees and different numbers of segments. Based on Massarwi's and Elber's work, Ezair et al. presented an approach for modeling solid objects containing volumetrically graded material distributions, using trivariate B-spline volumes [EDE17].

2.2.3. Subdivision Volumes

As an extension to the existing subdivision schemes for surfaces, volumetric subdivision algorithms have been developed to represent volumetric 3D models. In addition to vertices, edges, and faces, a volumetric control mesh consists of cells and has inner faces to separate them. Based on polyhedral control meshes, volumetric subdivision schemes create new vertices, edges, faces, and cells with every subdivision step, converging to the so-called *limit volume*. The set of all points in the interior of the model is defined by this continuous volumetric limit. Unlike spline volumes, volumetric subdivision schemes, especially Catmull-Clark (CC) Solids [JM99], can exhibit almost arbitrary topologies in their control meshes, resulting in a high flexibility in the design process (see also Section 3.1).

Joy and MacCracken [JM99] as well as Bajaj et al. [Baj+02] presented volumetric subdivision approaches on hexahedral meshes – Joy's and MacCracken's scheme [JM99] being the volumetric extension of Catmull-Clark surfaces [CC78]. Other methods were presented that operate on tetrahedral meshes, e.g. by Schaefer et al. [SHW04]. While the above mentioned volumetric subdivision schemes are approximating ones, Chang et al. [CMQ03] as well as McDonnel et al. [MCQ04] presented interpolating schemes for tetrahedral and hexahedral meshes, respectively. A good overview about existing subdivision schemes can be found in the 2003 survey paper by Chang and Qin [CQ03]. Figure 2.14 shows examples for Catmull-Clark solids [JM99], MLCA subdivision [Baj+02], and Chang et al.'s tetrahedral subdivision [CMQ03].



(a) Catmull-Clark solids (cut). Image taken from [BHU10b].

(b) MLCA subdivision of a cube (cut). Image taken from [Baj+02].

(c) Interpolatory tetrahedral subdivision (cut). Image taken from [CMQ03].

Figure 2.14.: Examples for three different volumetric subdivision schemes. While (a) and (b) show approximating schemes on hexahedral meshes, (c) shows an interpolating scheme on tetrahedral control meshes.

As for the *limit surface* of subdivision surfaces, the *limit volume* of subdivision solids can be evaluated mathematically for some subdivision schemes, e.g. for Catmull-Clark solids [JM99]. Similar to subdivision surfaces, different volumetric subdivision schemes have different requirements to the (in this case volumetric) control mesh and result in different continuities for their limit representation. However, the limit evaluation for subdivision solids is not as well explored as for subdivision surfaces. For CC solids, Burkhart et al. [BHU10b] presented a first, partially iterative limit evaluation approach in the context of physically based simulation.

As volumetric control meshes introduce *extraordinary edges* (EEs) as well as *extraordinary vertices* (EVs), many more possible combinations of irregular topologies exist in the volumetric case, compared to the surface case. Especially when multiple EEs meet in a volumetric EV, the topology to be handled during evaluation is far more complex than in the surface case. In the context of hexahedral meshes, Liu et al. provide a theoretical discussion of all possible extraordinary combinations that have to be handled [Liu+18].

Catmull-Clark solids are the basis for the methods presented throughout this dissertation. Therefore, a detailed explanation of the concepts behind CC solids can be found in Section 3.1.

2.3. Design Analysis

Analyzing a given design for its physical properties via physically based simulation involves solving a set of *partial differential equations* (PDEs) that describe the physical phenomena to be simulated. Those PDEs are usually defined over the continuous domain of the entire object.

In the following, two important concepts for preforming physically based simulation are presented. *Finite Element Analysis* (FEA), presented in Section 2.3.1 divides the simulation domain into discrete parts (the so-called *finite elements*) and discretizes the equations locally for every element. Common representations for FEA are tetrahedral and hexahedral meshes (see Section 2.2.1). *Isogeometric Analysis* (IGA) also divides the simulation domain into *elements* but uses the same (continuous) representation that is used to represent the geometry for solving the PDEs. Section 2.3.2 presents IGA on B-spline and NURBS representations, while Section 2.3.3 presents IGA on subdivision geometry.

2.3.1. Finite Element Analysis

Finite Element Analysis employs the *Finite Element Method* (FEM) described in the textbook by Zienkiewicz et al. [ZTT77]. FEA is widely used for physically based simulation, e.g. in the domain of structural mechanics. The field has matured over the years and many professional FE frameworks exists in academia, as well as in industry. Examples are Abacus [Das20c], CalculiX [DW19], Ansys [Inc20a], and Siemens NX [Inc20b]. As FEA has evolved in engineering, adapted and specialized approaches have been developed for computer animation and video games for creating physically based animations, e.g. by Müller et al. [Mül+02], Mezger and Straßer [MS06], and Nealen et al. [Nea+06]. Figure 2.15 shows the typical process of performing FEA, discretizing a CAD B-Rep model into a tetrahedral mesh, performing the simulation and visualizing the simulation results in a color-coded 3D view.

As mentioned above, the Finite Element Method partitions the continuous simulation domain into discrete elements such as tetrahedra or hexahedra. The particular choice of the element type depends on the application and the shape of the domain. Every element has *nodes* – storing its degrees of freedom (DoFs) – and a set of basis functions that are used to solve the PDEs. For tetrahedra with linear basis functions, the nodes are equivalent to the four vertices of each tetrahedron. The basis functions can be elevated to higher degrees (e.g. quadratic or cubic) by introducing additional nodes on the edges,

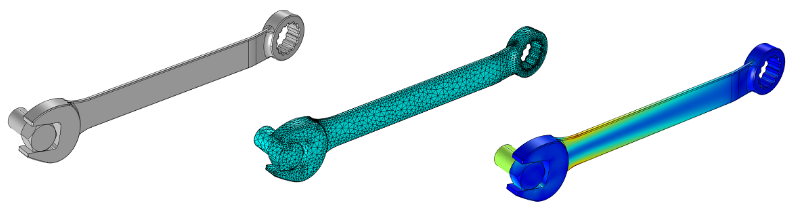


Figure 2.15.: The process of performing FEA. A B-Rep CAD model (left) is discretized into a tetrahedral mesh (center) for performing the simulation. The simulation results are visualized on the right, color-coded from blue to red. Image taken from [Fre14].

faces and inside the elements as described by Zienkiewicz et al. [ZZT05]. Later, Weber et al. adapted these improvements for the computer graphics community [Web+11; Web+15]. Grinspun et al. presented a framework for combining different element types (e.g. triangles, quadrilaterals, tetrahedra, or hexahedra) with different basis functions, such as linear, higher-order, or Loop subdivision [Loo87] basis functions, to conveniently set up tailored simulations [GKS02]. Nevertheless, the resolution of the finite element mesh has to be chosen in a way that the degrees of freedom can resolve the physical effects properly.

In the following, the mathematical concepts behind FEA are presented in detail. As this dissertation does not focus on simulation theory, but on the underlying representation schemes, the concepts will be presented for static – time-independent – structural analysis with a linear elastic material model. The extension to a transient simulation can be performed as explained in the textbook by Zienkiewicz et al. [ZZT05].

The Basics of Structural Mechanics

In static structural mechanics simulations, the displacement of the object is calculated from a set of boundary conditions – e.g. fixations and external forces. Calculating the displacement for the static case is a matter of determining the equilibrium state between external forces and internal stress. This means that all external forces acting on the object are counteracted by internal forces resulting in the sum of all forces being zero. Splitting the continuous simulation domain Ω into infinitesimal domains $V \subset \Omega$ leads to the following equilibrium equation with the boundary S of V and a vector $\mathbf{b}$ containing the external forces:

$$\int_S \mathbf{f}\,\mathrm{d}S + \int_V \mathbf{b}\,\mathrm{d}V = 0 \tag{2.16}$$

The external forces $\mathbf{f}$ on the surface are described by the stress tensor σ and the surface normal n with $\mathbf{f} = \sigma\mathbf{n}$. Applying Gauss's divergence theorem leads to the following equation:

$$\int_V \nabla \cdot \sigma + \mathbf{b}\,\mathrm{d}V = 0. \tag{2.17}$$

As this needs to hold for any domain V, the integration can be omitted, which results in the following PDE:

$$\nabla \cdot \sigma + \mathbf{b} = 0, \tag{2.18}$$

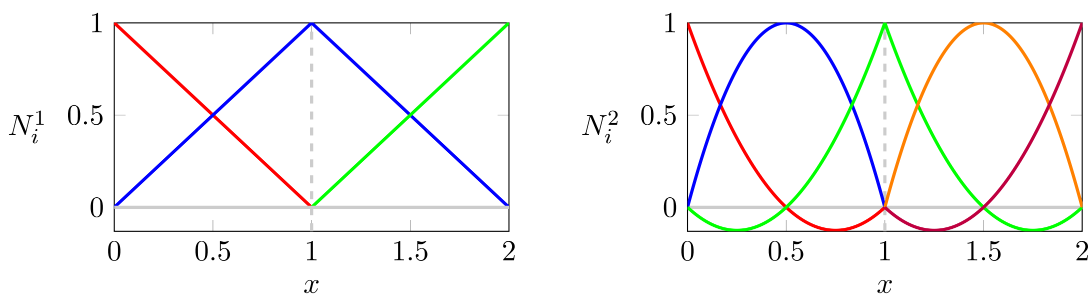
The solution of Equation (2.18) describes the equilibrium configuration. For the equation to have a solution, boundary conditions need to be defined. These can be in the form of *Dirichlet boundary conditions*, where the displacement function $\mathbf{u}(\mathbf{x})$ is fixed at a section of the boundary or in the form of *Neumann boundary conditions*, where external forces are applied at the boundary.

Elements and Basis Functions

For solving the PDE in Equation (2.18), the domain Ω is discretized into a set of *finite elements*. This discretization of most often performed by tetrahedral or hexahedral meshing (see Section 2.2.1). The partition of the domain into n_{el} finite elements $\mathbf{E}_{\mathrm{i}}$ also translates to the displacement function and can be written as follows:

$$\Omega = \bigcup_{i=1}^{n_{\mathrm{el}}} \mathbf{E}_{\mathrm{i}} \tag{2.19}$$

Each element has a finite number of nodes $\mathbf{p}_i$, defining its shape. For structural mechanics in $\mathbb{R}^3$, each node provides three degrees of freedom, i.e. the displacement in x, y, and z. The value of the displacement function $\mathbf{u}$ at a node $\mathbf{p}_i$ is defined as $\mathbf{u}_i$. To obtain a continuous function, these discrete values are interpolated with polynomial basis functions N_i^p of a certain degree p. In structural mechanics, tetrahedra are the most commonly used element type. The number of nodes per tetrahedron $\mathbf{T}_{\mathrm{i}}$ depends on the degree of the basis functions, resulting in 4 nodes for linear basis functions, 10 for quadratic ones, and 20 for cubic ones.



(a) Linear basis functions ($p = 1$) (b) Quadratic basis functions ($p = 2$)

Figure 2.16.: Basis functions of degree 1 (a) and 2 (b), typically used for FEM. Both figures depict the basis functions for two elements. Note the discontinuity at the boundary between the two elements (dashed line).

With the basis functions N_i^p, the displacement $\mathbf{u}(\mathbf{x})$ can be written as follows, with n being the total number of nodes of the finite element mesh:

$$\mathbf{u}(\mathbf{x}) \approx \sum_{i=1}^{n} \mathbf{u}_i N_i(\mathbf{x}), \tag{2.20}$$

The degree of continuity for $\mathbf{u}$ inside one element is determined by the degree p of the polynomial basis functions N_i^p. Figure 2.16 depicts the univariate basis functions for $p = 1$ and $p = 2$. Since these functions are defined locally for each element, $\mathbf{u}$ is C^p-continuous on the interior of the element. However, by default, the basis functions only guarantee C^0 continuity across element borders. Higher continuity can be constructed algorithmically by increasing the degree of the basis functions. The additional nodes introduced with higher degrees are utilized to create and maintain the continuity between elements. For general tetrahedral meshes, C^1 continuity can be achieved with degree $p = 9$, while C^2 requires basis functions of degree $p = 17$ [LS07]. Using basis functions of such high degree is impractical as the computational effort induced by the additional nodes outweighs the benefits of having increased continuity – raising the number of C^0 elements is much more efficient. This presents one of the significant shortcomings of this method in contrast to the subdivision-based methods presented in Section 2.3.3. Catmull-Clark subdivision basis functions inherently provide higher continuity between elements.

Due to the locality of the basis functions, the displacement $\mathbf{u}$ can be defined for each tetrahedron $\mathbf{T}$ as follows, with m denoting the number of nodes per tetrahedron:

$$\mathbf{u}|_{\mathbf{T}}(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{u}_i^{\mathbf{T}} N_i^{\mathbf{T}}(\mathbf{x}), \tag{2.21}$$

Equation (2.21) can also be written as a matrix-vector multiplication:

$$\mathbf{u}|_{\mathbf{T}}(\mathbf{x}) = \mathbf{N_T u_T} = \begin{pmatrix} N_1^{\mathbf{T}}(\mathbf{x}) & 0 & 0 & \cdots & N_m^{\mathbf{T}}(\mathbf{x}) & 0 & 0 \\ 0 & N_1^{\mathbf{T}}(\mathbf{x}) & 0 & \cdots & 0 & N_m^{\mathbf{T}}(\mathbf{x}) & 0 \\ 0 & 0 & N_1^{\mathbf{T}}(\mathbf{x}) & \cdots & 0 & 0 & N_m^{\mathbf{T}}(\mathbf{x}) \end{pmatrix} \begin{pmatrix} u_1^{\mathbf{T}} \\ v_1^{\mathbf{T}} \\ w_1^{\mathbf{T}} \\ \vdots \\ u_m^{\mathbf{T}} \\ v_m^{\mathbf{T}} \\ w_m^{\mathbf{T}} \end{pmatrix} \tag{2.22}$$

The strain tensor describes the correlation between the displacement and the external forces. Depending on the chosen material model, different strain tensors can be applied. For linear elasticity, the *linear strain tensor* ε is defined as follows:

$$\varepsilon = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{\partial v}{\partial y} & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) & \frac{\partial w}{\partial z} \end{pmatrix} \tag{2.23}$$

Inserting Equation (2.22) into the definition of ε results in the following formulation:

$$\varepsilon|_{\mathbf{T}} = \mathbf{S}\ \mathbf{u}|_{\mathbf{T}} = \mathbf{SN_T u_T} = \mathbf{B_T u_T} = \begin{pmatrix} \frac{\partial N_1^{\mathbf{T}}}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_m^{\mathbf{T}}}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1^{\mathbf{T}}}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_m^{\mathbf{T}}}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1^{\mathbf{T}}}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_m^{\mathbf{T}}}{\partial z} \\ \frac{\partial N_1^{\mathbf{T}}}{\partial y} & \frac{\partial N_1^{\mathbf{T}}}{\partial x} & 0 & \cdots & \frac{\partial N_m^{\mathbf{T}}}{\partial y} & \frac{\partial N_m^{\mathbf{T}}}{\partial x} & 0 \\ \frac{\partial N_1^{\mathbf{T}}}{\partial z} & 0 & \frac{\partial N_1^{\mathbf{T}}}{\partial x} & \cdots & \frac{\partial N_m^{\mathbf{T}}}{\partial z} & 0 & \frac{\partial N_m^{\mathbf{T}}}{\partial x} \\ 0 & \frac{\partial N_1^{\mathbf{T}}}{\partial z} & \frac{\partial N_1^{\mathbf{T}}}{\partial y} & \cdots & 0 & \frac{\partial N_m^{\mathbf{T}}}{\partial z} & \frac{\partial N_m^{\mathbf{T}}}{\partial y} \end{pmatrix} \begin{pmatrix} u_1^{\mathbf{T}} \\ v_1^{\mathbf{T}} \\ w_1^{\mathbf{T}} \\ \vdots \\ u_m^{\mathbf{T}} \\ v_m^{\mathbf{T}} \\ w_m^{\mathbf{T}} \end{pmatrix} \tag{2.24}$$

This allows for calculating the strain tensor for a tetrahedron $\mathbf{T}$ using the derivatives of the basis functions N_i and the local coordinates u, v, and w.

Weak Form and Stiffness Matrix

A common approach for solving the PDEs is to apply the so-called *weak form* as explained in the textbook by Zienkiewicz et al. [ZZT05]. This formulation allows the discretized displacement function to be of a lower continuity, thus permitting lower-order basis functions and allowing C^0 continuity across element borders.

The weak form of Equation (2.18) is obtained by multiplying an arbitrary test function $\mathbf{w} : \Omega \rightarrow \mathbb{R}^3$ and integrating over the entire domain Ω as follows:

$$\int_\Omega \mathbf{w}^T \left(\nabla \cdot \sigma + \mathbf{b}\right) \mathrm{d}V = 0 \tag{2.25}$$

As explained by Weber [Web15], Equation (2.25) can be rewritten as follows:

$$\int_\Omega \varepsilon(\mathbf{w})^T \mathbf{D} \varepsilon(\mathbf{u}) \, \mathrm{d}V - \int_\Omega \mathbf{w}^T \mathbf{b} \, \mathrm{d}V = 0 \tag{2.26}$$

Note that $\varepsilon(\mathbf{w})$ and $\varepsilon(\mathbf{u})$ are in *Voigt notation*, i.e. written as a vector of unique entries representing the symmetric tensor. As for the displacement function $\mathbf{u}$, the test function $\mathbf{w}$ is discretized for each tetrahedron $\mathbf{T}$:

$$\mathbf{w}|_\mathbf{T}\left(\mathbf{x}\right) = \mathbf{N}_\mathbf{T} \mathbf{w}_\mathbf{T} \tag{2.27}$$

Inserting Equations (2.24) and (2.27) into Equation (2.26) leads to the following formulation of the simulation problem and its equilibrium state:

$$\begin{aligned} &\int_\mathbf{T} \varepsilon(\mathbf{w})^T \mathbf{D} \varepsilon(\mathbf{u}) \, \mathrm{d}V - \int_\mathbf{T} \mathbf{w}^T \mathbf{b} \, \mathrm{d}V \\ &= \int_\Omega \left(\mathbf{B}_\mathbf{T} \mathbf{w}_\mathbf{T}\right)^T \mathbf{D} \mathbf{B}_\mathbf{T} \mathbf{u}_\mathbf{T} \, \mathrm{d}V - \int_\mathbf{T} \left(\mathbf{N}_\mathbf{T} \mathbf{w}\right)^T \mathbf{b} \, \mathrm{d}V \\ &= \mathbf{w}_\mathbf{T}^T \int_\mathbf{T} \mathbf{B}_\mathbf{T}^T \mathbf{D} \mathbf{B}_\mathbf{T} \, \mathrm{d}V \mathbf{u}_\mathbf{T} - \mathbf{w}_\mathbf{T}^T \int_\mathbf{T} \mathbf{N}_\mathbf{T}^T \mathbf{b} \, \mathrm{d}V \\ &= \mathbf{w}_\mathbf{T}^T \left(\mathbf{K}_\mathbf{T} \mathbf{u}_\mathbf{T} - \mathbf{f}_\mathbf{T}\right) \overset{!}{=} 0 \end{aligned} \tag{2.28}$$

$\mathbf{K_T}$ is called the element stiffness matrix and is of size $3m \times 3m$, where m is the number of nodes of the tetrahedron T:

$$\mathbf{K_T} = \int_{\mathbf{T}} \mathbf{B_T^{\mathit{T}} D B_T} \mathrm{d}V \tag{2.29}$$

To solve the simulation problem for the equilibrium state, the Equation (2.28) has to be solved for all tetrahedra simultaneously. Therefore, the global stiffness matrix $\mathbf{K}$ is assembled from all $\mathbf{K_T}$. $\mathbf{K}$ is of size $3n \times 3n$, where n is the total number of nodes of the tetrahedral mesh. $\mathbf{K_T}$ is calculated for each tetrahedron $\mathbf{T}$ and its entries are inserted into $\mathbf{K}$. Each entry of $\mathbf{K_T}$ describes the relation between two nodes of the tetrahedron $\mathbf{T}$. The row and column of where to insert this entry into $\mathbf{K}$ is determined by global indices of these two nodes. If $\mathbf{K}$ already has an entry at that position, the value is added to the existing one. This is the case when the two nodes are shared between multiple tetrahedra. Details on an efficient implementation of the matrix assembly are described in the textbook by Smith and Griffiths [SG98].

The equation for the global system writes as follows:

$$\mathbf{w}^T (\mathbf{Ku} - \mathbf{f}) = 0 \tag{2.30}$$

Equation (2.30) needs to be satisfied for an arbitrary test function $\mathbf{w}$. Therefore, a solution to Equation (2.30) is obtained by solving the following system of linear equations:

$$\mathbf{Ku} = \mathbf{f} \tag{2.31}$$

The solution vector $\mathbf{u}$ represents the resulting discretized displacement function, describing the equilibrium configuration. Each entry of $\mathbf{u}$ describes the displacement of the corresponding vertex in the tetrahedral mesh. Finally, interpolating these discrete values with the basis functions employed for the discretization (see Equation (2.20)) yields the displacement function $\mathbf{u}(\mathbf{x})$. This function can be used to calculate the stress and the strain tensor.

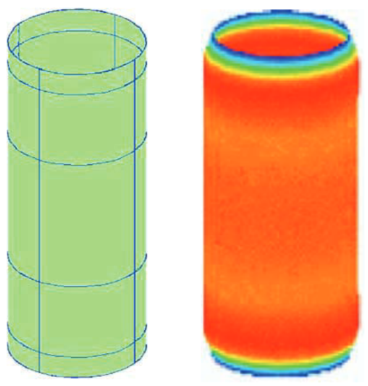
Solving the Linear System

Solving the static linear elasticity problem with FEA boils down to assembling and solving the linear system shown in Equation (2.31). Such sparse linear systems can be solved with a number of different solvers. Direct solvers use factorization to solve the system, while iterative solvers, such as a *Jacobi* solver or a *Conjugate Gradient* (CG) solver iteratively converge towards the solution. When applying an iterative solver, a trade-off between the accuracy of the solution and the required computation time can be achieved by limiting the maximum number of iterations. Further details can again be found in the textbook by Smith and Griffiths [SG98].

Classical CPU-based solvers are studied well and are the state of the art since many years. A set of different solvers is available e.g. as part of the *Eigen* library [JG08]. Weber et al. [Web+13] presented data structures and algorithms for solving sparse linear systems efficiently on GPUs. Recently, also commercial FEA frameworks such as Ansys have started to exploit the GPU [Inc20a]. By using the computational potential of modern GPUs, huge improvements to the performance have been achieved.

2.3.2. Spline-Based Isogeometric Analysis

Isogeometric Analysis (IGA) tries to avoid the problems related to generating a discrete mesh from a continuous CAD representation, by using the same representation for design and simulation. Spline-based IGA was introduced by Hughes et al. in 2005 [HCB05] for surfaces, as well as for solid models. Figure 2.17 shows examples of performing IGA on a


(a) IGA on NURBS surfaces

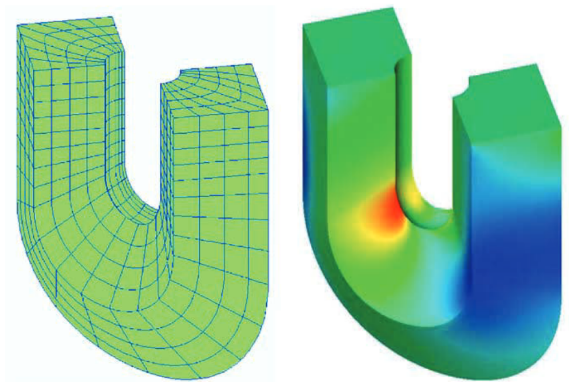

(b) IGA on NURBS volumes

Figure 2.17.: IGA on NURBS surfaces and volumes. (a) shows pressure applied to the inner side of a cylinder represented with a single NURBS surface. (b) shows the result of applying external forces to a horseshoe model represented with a single NURBS volume. Images taken from [HCB05].

thin-shell model of a cylinder, as well as on a solid model of a horseshoe. In both cases, the corresponding NURBS representation is used for describing the geometry and for performing the simulation.

When simulating thin-shell objects, the B-Rep CAD model – consisting of B-spline or NURBS surfaces – can be used directly. However, the model might consist of multiple connected surfaces and might include trimming curves. Kim et al. [KSY09], as well as Schmidt et al. [SWB12] presented methods for integrating the trimming information into the analysis. While most approaches for spline-based IGA of surface models rely on B-spline or NURBS surfaces, several methods using alternative spline representations exist. Bazilevs et al. [Baz+10] showed how to perform IGA on *T-spline* surfaces. Johannenssen et al. [JKD14] presented an IGA approach on *LR B-spline* surfaces.

Solid models have to be represented by trivariate B-spline or NURBS volumes. Cohen et al. [Coh+10] provide recommendations on how to represent several different primitives with trivariate splines, aiming at good parametrization quality for the simulation. Since CAD systems such as CATIA [Das20b] or SolidWorks [Das20a] only support spline surfaces, a conversion from the bivariate representation to the trivariate one is still required as described in Section 2.2.2. Alternatively, the spline volumes can be created from scratch using the methods also described in Section 2.2.2. Once the trivariate representation is available, iterations of design and simulation can be performed without switching representations.

As also explained in Section 2.2.2, trivariate spline models only provides C^0 continuity across element borders [ME16]. Similar to finite element meshes, higher continuity between elements can be constructed algorithmically for bivariate and trivariate spline models, but implies additional computational effort, especially when modifying the geometry. Cottrell et al. showed that having C^1 or even C^2 continuity between elements improves the simulation results when solving elliptic PDEs, e.g. for heat simulation or structural analysis [CHR07].

Using the B-spline or NURBS representation for design and simulation implies also using the corresponding spline basis functions for solving the partial differential equations. Therefore, spline-based IGA methods need a specialized environment to perform the simulation and have thereby only limited compatibility with publicly available or commercial FEA systems such as Ansys [Inc20a] or CalculiX [DW19].

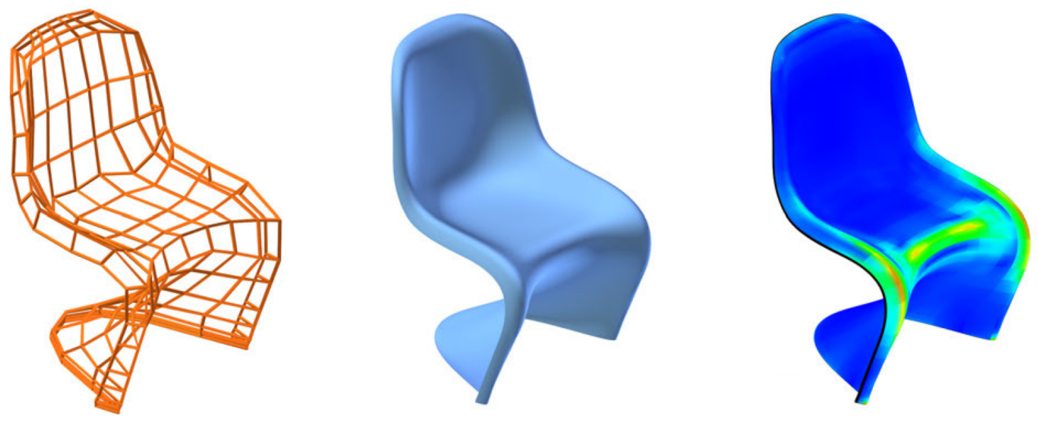

(a) Control mesh (b) Limit surface (c) Simulation results

Figure 2.18.: IGA on subdivision surfaces. The images show the control mesh (a), the limit surface (b), as well as the color-coded simulation results (c) of a thin-shell model of a chair. Images taken from [RAF15].

2.3.3. Subdivision-Based Isogeometric Analysis

Picking up the idea of using one representation for both, geometric design and simulation, approaches for IGA on subdivision surfaces have evolved. Figure 2.18 shows an example for performing IGA on subdivision surfaces.

Grinspun et al. presented an approach for computing structural mechanics problems on subdivision surfaces [Gri+99]. Similarly, Cirak et al. [Cir+02] and Wawrzinek et al. [WHP11] utilized subdivision surfaces for simulating thin shell structures. Furthermore, Green and Turkiyyah presented the usage of the previously mentioned *ghost points* (see Section 2.1.2) for defining boundary conditions on open subdivision surfaces [GT04]. Nguyen et al. show how to solve Poisson's equation on Catmull-Clark surfaces [NKP14]. In their study, they also provide a qualitative comparison of different FEM and IGA approaches, as well as point out the need for adapted quadrature rules when integrating over Catmull-Clark surfaces. As already mentioned in Section 2.1.2, Wawrzinek and Polthier [WP16], as well as Barendrecht et al. [BBK18] presented improved integration techniques for that purpose.

While simulations on subdivision surfaces are most often performed on the basis of Catmull-Clark surfaces, Dikici presented an approach for IGA on Doo Sabin subdivision surfaces [DSO12]. Also, Pan et al. presented an IGA approach based on Loop subdivision surfaces [Pan+15]. Riffnaller-Schiefer et al. bridged the gap to bivariate spline-based IGA with their approach for IGA on NURBS-compatible subdivision surfaces [RAF16].

Although many simulation algorithms exist for subdivision surfaces, IGA on subdivision solids is rather unexplored. Up to now, volumetric subdivision is mostly used for global or local refinement of discrete simulation meshes as described by [BHU10a]. Due to their volumetric nature, using subdivision solids in an isogeometric scenario is very tempting. However, a fast limit evaluation technique is required for efficiently calculating the entries of the linear system, but is substantially more complex for solids than for surfaces. In 2010, Burkhart et al. [BHU10b] presented a first IGA approach on Catmull-Clark solids that evaluates the limit by applying a high number of local subdivision steps near extraordinary edges and vertices. An example from Burkhart et al.'s approach can be seen in Figure 2.19.

Besides an efficient limit evaluation, other important aspects for IGA on subdivision solids have also not been investigated yet, i.e. accurate numerical integration and mesh quality. In Chapter 5, I address these open points and present my improved IGA approach.

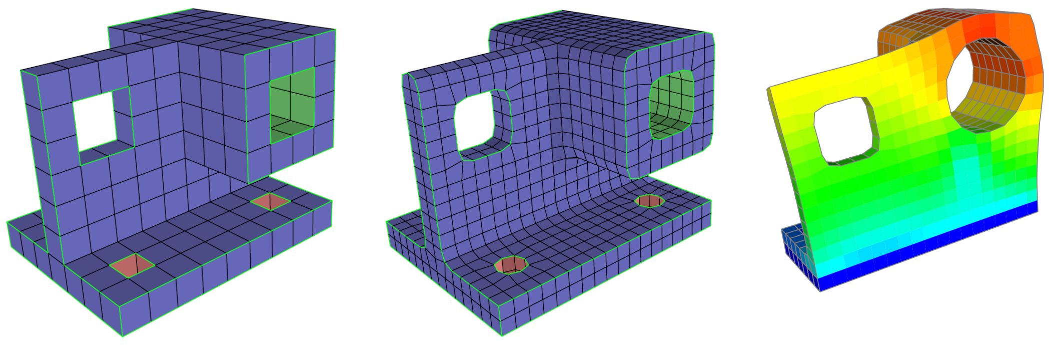

(a) Volumetric control mesh (b) Subdivided mesh (c) Simulation results (cut)

Figure 2.19.: Example for IGA on Catmull-Clark solids. (a) shows the CC-solid control mesh and the load case with constrained areas in orange and forced areas in green. Crease edges are also highlighted in green. (b) shows the subdivided mesh prior to the simulation. (c) shows the color-coded simulation results. A cut through the model visualizes the inner CC-solid cells. Images taken from [BHU10b].

2.4. Summary

Covering both domains, geometric modeling and physically based simulation, as well as providing a smooth transition between them, is mandatory when aiming for an integrated modeling and simulation workflow. While the individual approaches presented in this chapter all have benefits in their specific area of application, efficient links to the other building blocks of integrated modeling and simulation are still missing.

Discrete FE meshes provide a high flexibility when approximating complex 3D objects of almost arbitrary shape. FE algorithms have matured over the decades and many high-quality FE frameworks are available – commercially and for free. However, most information from the original design model is lost during the meshing process, making it difficult to derive design changes from the simulation results. The FE meshes themselves are tedious to model or adapt interactively due to their many degrees of freedom.

Trivariate spline representation are well suited for simulating engineering parts as analytic shapes can be represented accurately. However, the conversion of existing CAD models is challenging and few interactive volumetric modeling environments exist. Complex 3D objects have to be represented by trimming or tiling the spline volumes, reducing the continuity to C^0.

While many volumetric subdivision approaches have been presented over the years, these works concentrated on the corresponding subdivision scheme itself and not on how to create the models in the first place. The volumetric subdivision models had to be created either algorithmically or explicitly by hand. To the best of my knowledge, no interactive modeling environment for volumetric subdivision control meshes existed prior to this dissertation.

From the simulation point of view, the work of Burkhart et al. [BHU10b; Bur11] is the closest to this dissertation. However, Burkhart et al. focused on how to utilize Catmull-Clark solids for simulation. As for the approaches presented in the context of spline-based IGA, the creation of the CC-solid models as well as the interaction between modeling and simulation have not been addressed. Regarding the simulation, in addition to efficient limit evaluation, important aspects such as accurate numerical integration and the analysis of mesh quality remain open for investigation.

3. Subdivision Solids and Volumetric Modeling

Subdivision surfaces are the dominant representation scheme for 3D models in the entertainment industry. A conceptually infinite refinement generates the so-called *limit* – the smooth surface of the geometry – from a mesh of discrete control points. However, iteratively applying the subdivision rules to approximate the limit is inefficient. Stam overcame this deficiency with his groundbreaking algorithms [Sta98a; Sta98b] for evaluating limit points in constant time (see also Section 2.1.2). When processing subdivision surfaces, efficient limit point evaluation is vital. Therefore, Stam's algorithms decisively contributed to the wide acceptance of subdivision surfaces, i.e. Catmull-Clark surfaces [CC78], in many industries, including manufacturing. However, subdivision solids have yet to be taken up to the same extent, although they have great potential in applications such as physically based simulation (see Chapter 5) and efficient data representation, e.g. for scalar fields. In their 2003 survey paper, Chang and Qin already documented the growing interest in and importance of subdivision solids [CQ03]. However, an efficient volumetric limit evaluation algorithm as well as a set of feasible volumetric modeling operations were missing prior to this dissertation.

Section 3.1 introduces Catmull-Clark solids as the volumetric representation on which the approaches presented in this dissertation are based. Section 3.2 presents my constant-time volumetric limit evaluation approach for Catmull-Clark solids. Section 3.3 presents volumetric modeling operations for creating and manipulating CC-solid control meshes.

Section 3.2 contributes to answering Research Question 1:

RQ1: How can limit evaluation for CC solids be performed more efficiently?

Section 3.3 contributes to answering Research Question 2:

RQ2: How can volumetric control meshes be created and modified?

3.1. Volumetric Catmull-Clark Subdivision

Catmull-Clark (CC) solids provide a set of subdivision rules to refine a polyhedral control mesh into a smooth limit volume. For regular topologies, CC solids are equivalent to trivariate uniform cubic B-splines. However, CC solids do not require a tensor product topology in their control mesh and can therefore represent a complex object as a single subdivision volume. The CC-solid basis functions inherently provide C^2 continuity away from irregularities.

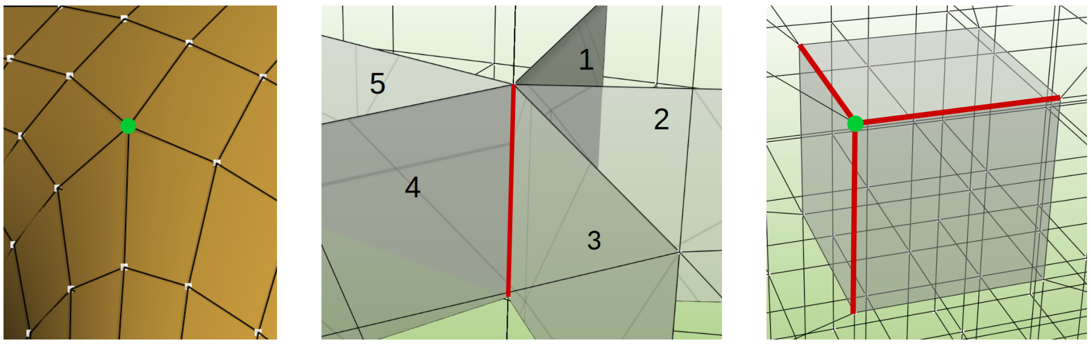
In general, CC-solid control meshes are closed polyhedral meshes with the following topological properties:

- Every face on the boundary is connected to one cell.
- Every face inside the mesh is connected to two neighboring cells.
- Every edge connects two vertices and is shared by at least two faces.

In the following, the concepts of *extraordinary vertices* (EVs) and *extraordinary edges* (EEs) are introduced and defined. It is important to understand these definitions as well as the difference between the surface case and the volumetric case, as the concepts of EEs and EVs are used extensively throughout this dissertation. Figure 3.1 shows visualizations of a surface EV, an EE, as well as a volumetric EV.

- **Extraordinary Vertex (surface):** A surface control point with a valence other than 4. The valence of a control point being the number of edges attached to it.
- **Extraordinary Edge:**
  - An edge inside the volumetric control mesh with a valence other than 4, or
  - an edge at the boundary of the volumetric control mesh with a valence other than 3. The valence of an edge being the number of attached faces.
- **Extraordinary Vertex (volume):** A volumetric control point where at least two EEs meet for a single cell.

As the subdivision rules for Catmull-Clark solids (see Section 3.1.1) are formulated generically for arbitrary polyhedra, CC-solid control meshes are not restricted to hexahedral cells. Please note, that although arbitrary polyhedra can be subdivided using the CC solids subdivision rules, not all types of polyhedra subdivide into hexahedra after a finite number of subdivision steps. Supported types of polyhedra include hexahedra, tetrahedra, wedges/prisms, and extruded polygons (e.g. pentagons, hexagons, . . .) with quadrilateral side faces.



(a) Surface EV (b) Extraordinary edge (c) Volumetric EV

Figure 3.1.: Examples of an extraordinary vertex on the surface with a valence of $N = 5$ (a), an extraordinary edge with $N = 5$ (b), as well as an extraordinary volumetric vertex being the shared vertex of three EEs, each with $N = 3$ (c). The EVs are highlighted in green, the EEs are visualized in red.

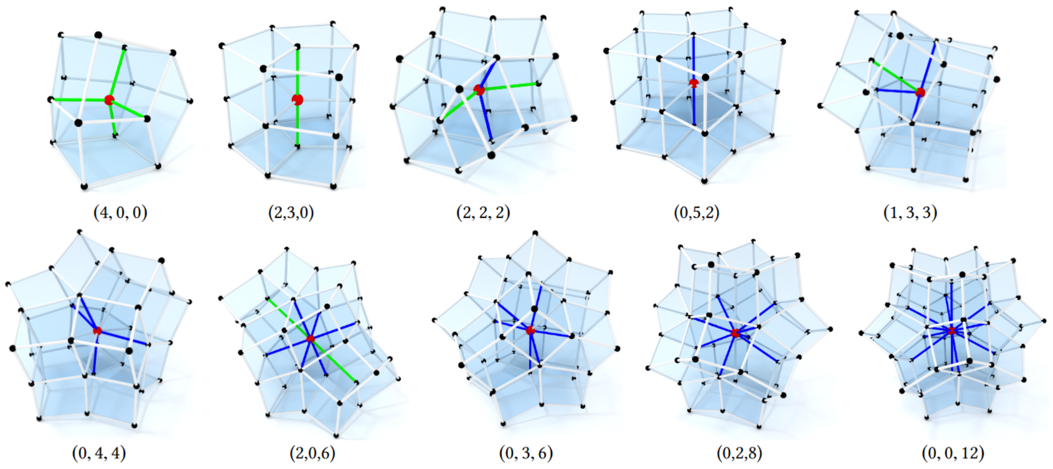


Figure 3.2.: Complex polyhedra that subdivide into hexahedra after one step of CC-solid subdivision. The EEs that are created inside the polyhedron when subdividing it are highlighted in green for a valence of $N = 3$ and in blue for $N = 5$. EVs are shown in red. The numbers below each model denote the numbers of newly created edges with valences 3, 4, and 5, respectively. The images were taken from the 2018 paper by Liu et al. [Liu+18].

In their 2018 paper, Liu et al. show additional types of complex polyhedra that also subdivide only into hexahedral cells [Liu+18]. However, subdividing a non-hexahedral cell always introduces EEs, and for most cases also an EV in the center of the cell. Liu et al. also present a systematic way for enumerating all irregular configurations. Figure 3.2 shows all topological configurations that create edges of valences 3, 4, and 5 after one subdivision step, visualizing the EEs and EVs that were generated during the process.

Pyramids with $n > 3$ sides are not supported in CC-solid control meshes. Although such pyramids can be subdivided with the CC-solid subdivision rules, limit evaluation is not possible. n-sided pyramids subdivide into n hexahedra at the base of the pyramid and one polyhedron with $2n$ quadrilateral faces at the top. Subdividing this complex polyhedron creates two new instances of the same polyhedron along with $2n$ new hexahedra. Figure 3.3 shows the subdivision of a four-sided pyramid and highlights the non-hexahedral polyhedra produced during the process. The problem of partitioning a 4-sided pyramid into hexahedra is also known as *Schneider's pyramid* [Sch96a]. Although several solutions have been presented over the years, e.g. by Verhetsel et al. [VPR19], all produce large numbers of EEs and EVs and are not related to the CC-solid subdivision rules.

In addition to their geometry, CC-solid models can represent volumetrically varying properties. Such models can be used in applications for multi-material additive manufacturing (see Section 6.3), multi-material structural analysis, and efficient representation of vol-

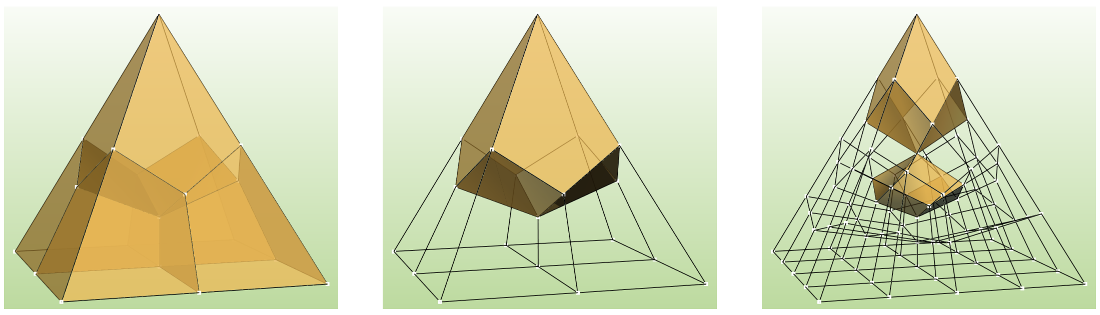
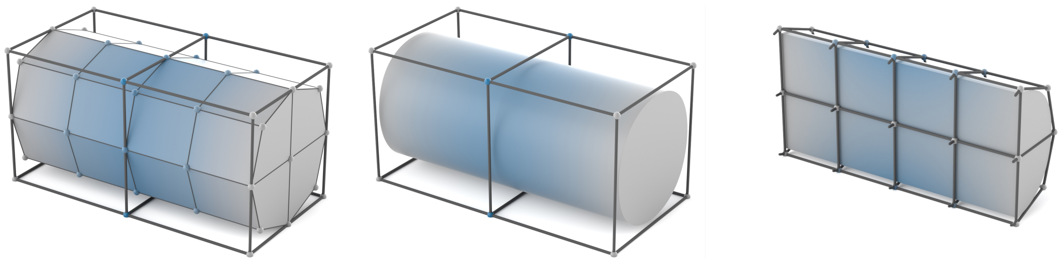

(a) First subdivision (b) The non-hexahedral polyhedron (c) Second subdivision

Figure 3.3.: CC-solid subdivision performed on a 4-sided pyramid. (a) shows the first subdivision with four hexahedra at the base and a non-hexahedral polyhedron at the top (highlighted in (b)). (c) shows the non-hexahedral polyhedra after two subdivision steps.

umetric fields, e.g. simulation results. Each CC-solid control point is enhanced with a discrete property value – scalar value, vector, or tensor – such as density, thermal conductivity, or stiffness. The limit property values are defined by the same volumetric subdivision basis functions as the 3D positions and provide the same C^2 continuity away from EEs and EVs. Figure 3.4 shows an example of a CC-solid model with volumetrically graded stiffness.



(a) Subdivision (b) Limit surface (c) Limit volume (cut)

Figure 3.4.: Material properties assigned at the control points and their effect on the CC-solid model. (a) shows the effects on the surface after one subdivision, while (b) shows the corresponding limit surface. (c) shows a cut through the subdivided object, showing inner control points and their effect on the limit volume. White color represents a low stiffness, while blue color represents a high one.

3.1.1. The Subdivision Rules for Catmull-Clark Solids

With Joy's and MacCracken's original subdivision rules for Catmull-Clark solids [JM99], the CC-solid basis functions can only be expressed as a tensor product for regular cases (see also Bajaj et al.'s work on MCLA subdivision [Baj+02]). However, having the volumetric (trivariate) subdivision basis functions as a tensor product of bivariate and univariate Catmull-Clark/B-spline basis functions is an important requirement for the volumetric limit evaluation method presented in Section 3.2. Therefore, I adapt the subdivision rules w.r.t. the original rules [JM99]. While Burkhart et al. presented a modified edge rule [BHU10a], the vertex rule must be modified as well to achieve the tensor product property around EEs and EVs. In the following, I also present the new vertex rule. For regular edges and vertices, the modified rules are identical to the original ones.

The Catmull-Clark solid subdivision rules used throughout this dissertation write as follows:

1. For every cell, add a new vertex c – the so-called *cell point* – at the cell's midpoint C_{mid}.
$$c = C_{mid} \tag{3.1}$$

2. For every face, add a new vertex f – the *face point* – at the weighted average of the face's midpoint F_{mid} and the midpoints of the two adjacent cells C_{mid1} and C_{mid2}.
$$f = \frac{C_{mid1} + C_{mid2} + 2F_{mid}}{4} \tag{3.2}$$

3. For every edge, add a new vertex e – the *edge point* – at the weighted average of the edge's midpoint E_{mid} and the averaged midpoints of all adjacent cells C_{avg} and faces F_{avg}, with N being the valence of the edge.
$$e = \frac{C_{avg} + 2F_{avg} + (\mathrm{N}-3)E_{mid}}{\mathrm{N}} \tag{3.3}$$

4. Move every original vertex to its new location v at the weighted average of its original location V and the locations of all neighboring vertices V_e that share an edge, vertices V_f that share no edge but a face, and vertices V_c that share no face but a cell with the vertex. n_e, n_f, and n_c denote the number of edges, faces, and cells connected to the original vertex.
$$\begin{gathered} v = \frac{w_v V + \sum_{V_e}(4\mathrm{N}^2 - 7\mathrm{N})V_e + \sum_{V_f} 6V_f + \sum_{V_c} V_c}{8{n_c}^2} \\ w_v = 8{n_c}^2 - \sum_{V_e}(4\mathrm{N}^2 - 7\mathrm{N}) - 6n_f - n_c \end{gathered} \tag{3.4}$$

5. Create the new edges, faces and cells by connecting each *cell point* with its corresponding *face points* and each *face point* with its corresponding *edge points*.

For elements on the outer surface, the original subdivision rules for Catmull-Clark surfaces apply [CC78], rendering the outer surface of a CC-solid subdivision volume equivalent to a CC subdivision surface. As for Catmull-Clark surfaces, sharp features can be defined using crease edges (see Section 2.1.2). Figure 3.5 shows an example of two CC-solid subdivision steps applied on a control mesh consisting of a single hexahedral cell.

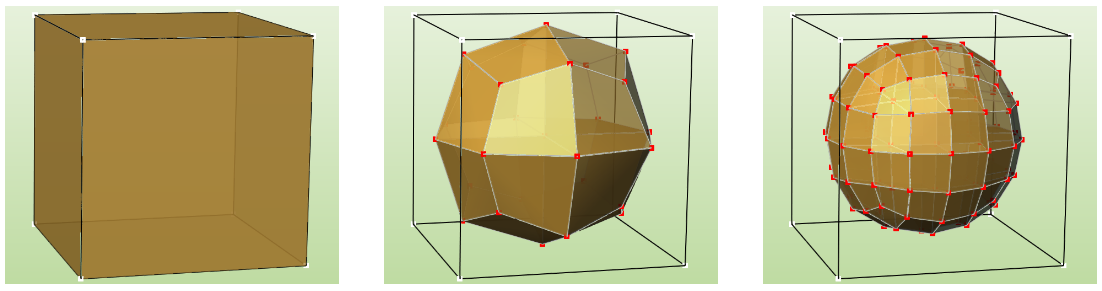

(a) Original control mesh (b) One subdivision step (c) Two subdivision steps

Figure 3.5.: Iterative subdivision with volumetric Catmull-Clark subdivision rules on a hexahedral control mesh. (a) shows the control mesh. (b) and (c) show the results of the subdivision process. In addition to the boundary faces, the cells and inner faces are subdivided accordingly.

3.1.2. Precomputable Subdivision Matrices

As in the case of Catmull-Clark surfaces, CC-solid subdivision can be performed with subdivision matrices. One step of subdivision from level $k-1$ to level k can be expressed as a matrix-vector multiplication of the corresponding subdivision matrix S_k and the control points p_{k-1} of level $k-1$:

$$p_k = S_k * p_{k-1} \tag{3.5}$$

The subdivision matrix S_k contains the weights needed to calculate p_k from p_{k-1}, according to the given subdivision rules. As one step of CC-solid subdivision creates one additional vertex per edge, face and cell of the control mesh, the new number of vertices $|p_k|$ equals the number of original vertices $|p_{k-1}|$ plus the number of edges $|e_{k-1}|$, faces $|f_{k-1}|$ and cells $|c_{k-1}|$ of the previous level. Therefore S_k is of size $|p_k| \times |p_{k-1}|$. Figure 3.6 shows the structure of S_1 for a CC-solid control mesh that consists of a single hexahedral cell with 8 vertices, 12 edges, and 6 faces (see also Figure 3.5(a)).

Subsequent subdivision steps can be performed by either using the resulting control points of the previous level or by chaining the subdivision matrices together as follows:

$$\begin{aligned} p_{k+1} &= S_{k+1} * p_k \\ &= S_{k+1} * S_k * p_{k-1} \end{aligned} \tag{3.6}$$

This matrix representation of the subdivision rules allows for computing a combined subdivision matrix $\hat{S}_n$ that directly generates the subdivided mesh of subdivision level n from the original control points p_0:

$$p_n = \hat{S}_n * p_0 \tag{3.7}$$

$\hat{S}_n$ is the product of the subdivision matrices S_k of all subdivision levels k from 1 to n:

$$\hat{S}_n = \prod_{k=n}^{1} S_k \tag{3.8}$$

As these matrices depend just on the topology of the CC-solid model and are independent from the actual position of the control points, $\hat{S}_n$ can be precomputed and re-used in scenarios where the topology does not change.

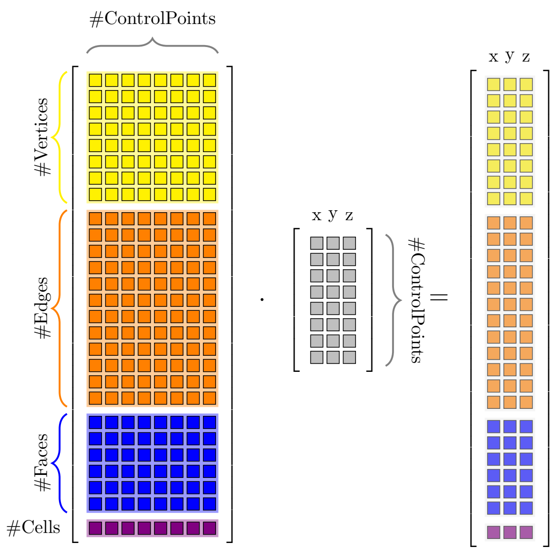


Figure 3.6.: Structure of the subdivision matrix S_1 (left) multiplied with the original control points p_0 (center) to calculate the subdivided points p_1 (right) for a CC-solid control mesh with a single hexahedral cell. Given by the CC-solid subdivision rules, S_1 consists of one row per vertex, edge, face, and cell of the original control mesh.

3.1.3. Local Parametrization

For Catmull-Clark solids, every cell is parametrized in its local coordinate system (u, v, w). Analogously to the patches of CC surfaces, every CC-solid cell forms a unit cube in its local parameter space $(u, v, w) \in [0, 1]^3$. Iteratively applying the subdivision rules as well as evaluating the limit resembles a mapping $\mathbf{f}$ from the local parameter space (u, v, w) of every control mesh cell to the Euclidean space $(x, y, z) \in \mathbb{R}^3$ (see Figure 3.7).

At every parameter point (u, v, w) in a CC-solid cell, the Jacobian matrix $\mathbf{J}$ contains the partial derivatives of the point's limit position (x, y, z) in Euclidean space with respect to each parameter u, v and w:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w} \end{pmatrix} \tag{3.9}$$

The Jacobian matrix is used for several calculations when working with Catmull-Clark solids, e.g. for performing numerical integration as shown in Section 5.1.4, as well as for calculating the parametrization quality measure presented in Section 5.2.1.

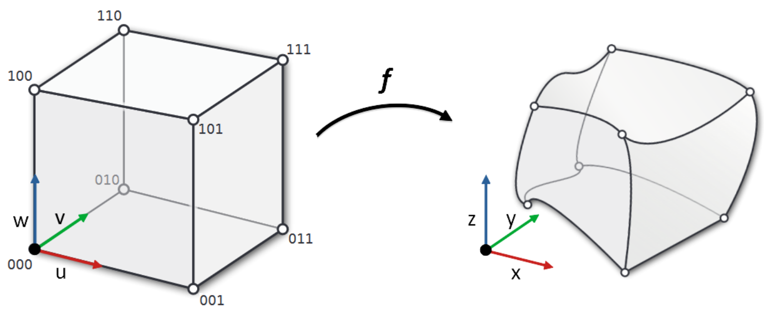


Figure 3.7.: The transition function $\mathbf{f}$ from local parameter space $(u, v, w) \in [0, 1]^3$ to the Euclidean space (x, y, z). The unit cube in its local parameter space (left) is transformed into Euclidean space where its shape is defined by its control points and basis functions (right). For Catmull-Clark solids, $\mathbf{f}$ resembles the transition from the control mesh to the limit volume. The image is adapted from the 2013 course by Erickson [Eri13].

3.2. Constant-Time Limit Evaluation for Catmull-Clark Solids

The content of this section is based on the following paper:

Direct Limit Volumes: Constant-Time Limit Evaluation for Catmull-Clark Solids
C. Altenhofen, J. Müller, D. Weber, A. Stork and D. W. Fellner

Published in *Pacific Graphics Short Papers*
2018
The Eurographics Association

The structure of volumetric models inherently differs from that of surface models. While surface meshes consist of vertices, edges and faces, volumetric meshes are defined by cells, each being bounded by a set of faces – which are again bounded by edges and vertices. Also, subdivision solids are evaluated on a per-cell, instead of a per-face basis. Extending Stam's ideas to the volumetric case uncovers challenging topological configurations that require novel solutions for direct and exact evaluation. These configurations comprise volumetric irregularities, boundaries and sharp features.

While every edge of a closed 2-manifold surface mesh must have two neighboring faces in order to form a valid topology, edges of volumetric meshes can have arbitrarily many faces (at least two). Therefore, while Catmull-Clark surfaces might contain extraordinary vertices, CC solids might contain extraordinary edges (EEs) as well as volumetric extraordinary vertices (EVs) – the definitions can be found in Section 3.1 – that require special handling when evaluating the limit. Furthermore, in $\mathbb{R}^3$, subdivision surfaces typically form a closed 2-manifold, whereas subdivision solids always have a boundary that requires special treatment.

In this section, I present my efficient limit evaluation approach for Catmull-Clark solids. I present new direct evaluation techniques for irregular volumetric topologies (Section 3.2.2) and boundary cells (Section 3.2.3), which allow for calculating the limit of CC subdivision solids at arbitrary parameter values in constant time. As in the surface case, regular topologies can be evaluated directly, using cubic B-spline basis functions. Irregular topologies have to be subdivided locally until the target point (u, v, w) can be evaluated. As for Stam's approach, the key idea is to perform all required local subdivision steps with a local subdivision matrix that does not change throughout the steps. This way, the eigenstructure of this subdivision matrix can be used to combine all local subdivision steps and evaluate the limit in constant time. However, in the volumetric case, local subdivision does not isolate irregularities as easily as in the surface case.

The approach is faster than the existing evaluation technique by Burkhart et al. [BHU10b] for every topological configuration or target parameter (u, v, w) that requires more than two local subdivision steps. For one or two subdivision steps, the overhead induced by applying the eigenstructure exceeds the computational cost of explicitly using the local subdivision matrix.

3.2.1. The Regular Case

The generalization of the bivariate B-spline surface evaluation to the trivariate case is straightforward. A regular CC-solid cell is defined by a $4 \times 4 \times 4$ grid of control points as shown in Figure 3.8. Each limit point e is evaluated with three parameters u, v and $w \in [0, 1]$ as follows:

$$e(u, v, w) = \sum_{i=0}^{63} \mathbf{p}_i N_i(u, v, w) \tag{3.10}$$

p_i denote the 64 local control points influencing the cell. The trivariate basis functions $N_i(u, v, w)$ are defined as a product of three univariate cubic B-spline basis functions:

$$N_i(u, v, w) = N_{\lfloor i/16 \rfloor}(u)\ N_{\lfloor i/4 \rfloor \bmod 4}(v)\ N_{i \bmod 4}(w) \tag{3.11}$$

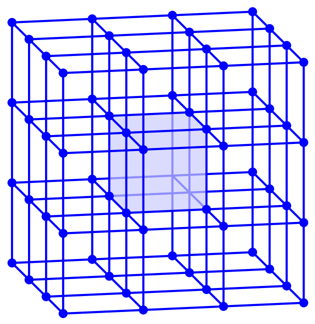
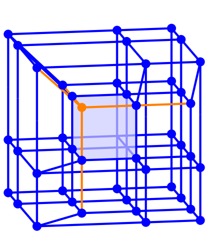
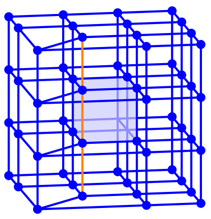
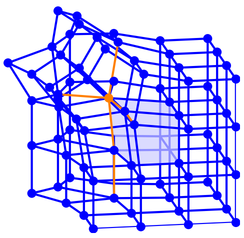


Figure 3.8.: Regular local control mesh. The cell to be evaluated is shown in light blue.


(a) Irregular control mesh with three EEs, each with a valence of $\mathrm{N} = 3$


(b) Layered control mesh for $\mathrm{N} = 3$


(c) Irregular control mesh with valences 5, 5, and 4

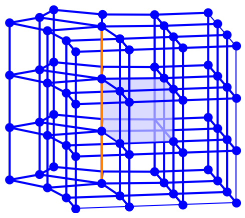

(d) Layered control mesh for $\mathrm{N} = 5$

Figure 3.9.: Fully irregular (a, c) as well as layered (b, d) local control meshes with EEs of valences $\mathrm{N} = 3$ and $\mathrm{N} = 5$. The EEs and EVs are marked in orange. The cell to be evaluated is highlighted in light blue.

3.2.2. Irregular Cases

As local control meshes in a CC-solid model do not form regular $4 \times 4 \times 4$ grids near irregularities, cells with an irregular topology, i.e., cells that contain extraordinary edges and/or vertices, cannot be processed using standard B-spline evaluation. Figure 3.9(a) shows an example of a local control mesh with three EEs of valence 3 and one EV. If multiple EEs are present in the cell to be evaluated, all EEs in the local control mesh must be radiating from the EV. If this is not the case, the mesh has to be subdivided once to achieve this property.

When evaluating one of the highlighted cells from Figure 3.9, it is first necessary to define the local (u, v, w) coordinate system. If the cell contains more than one EE – and therefore also an EV – the origin of the (u, v, w) coordinate system is placed at the EV. Otherwise, the origin of the coordinate system is chosen as one of the endpoints of the EE. From the origin, a left-handed coordinate system is created with the w axis pointing along the EE, or one of the EEs, respectively.

Consistent Numbering Scheme

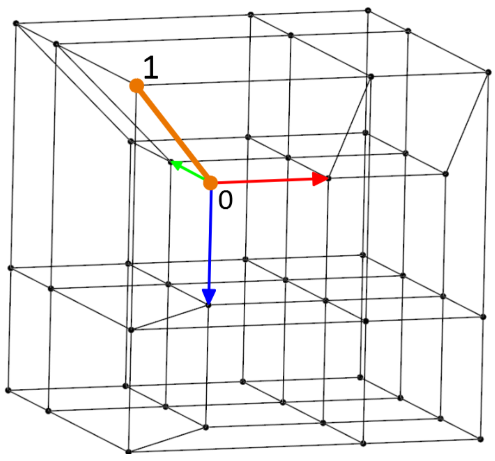
In order to allow for the limit of irregular cells to be evaluated in constant time, their control points have to be numbered, such that the topology of the local control mesh stays consistent in each subdivision step. Additionally, the index of each point and its corresponding subdivided point must be the same in each subdivision step.

In contrast to the surface case, a local volumetric control mesh is not uniquely defined by just a single valence N and can have many different topological configurations in the direct neighborhood of its EV. The topology for the local control mesh of an irregular CC-solid cell is defined by the valences N_u, N_v, and N_w of all three potential EEs radiating from the EV. Liu et al. present a systematic approach for enumerating all possible combinations [Liu+18]. Figure 3.2 shows the configurations consisting of edges with valences 3, 4, and 5. The consistent numbering scheme presented in this section supports all irregular configurations enumerated by Liu et al.

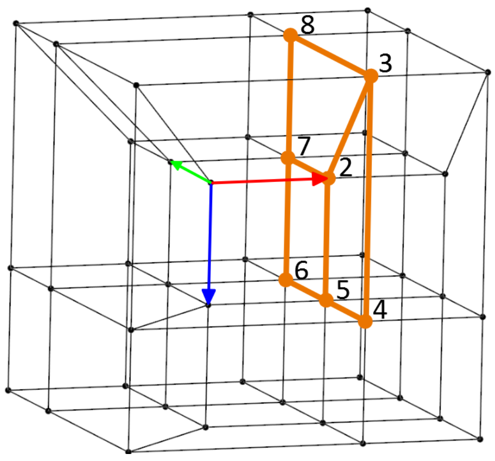
Figures 3.10 and 3.11 show an example of numbering the control points (CPs) of an irregular cell with three EEs of valence 3. The numbering starts at the extraordinary vertex, assigning indices to the EV and to all CPs that share an edge with the EV not resembling the u, v, or w axis of the local coordinate system (see Figure 3.10(a)). The volumetric data structure described in Section 3.3.1 is employed to iterate over all control points in a consistent way, assigning a unique index to each control point. In the next steps, the remaining control points oriented around the potential EEs are processed. In each direction u, v, and w, these CPs resemble layers of the control mesh, oriented perpendicular to the corresponding EE. Each layer forms a two-dimensional control mesh with the same valence as the EE. This allows for locally using Stam's 2D numbering scheme [Sta98a]. The inner layers of CPs are collected as the first $2\mathrm{N}+1$ CPs in Stam's scheme as shown in Figures 3.10(b), (c), and (d). As certain CPs are shared between the three directions, new indices are assigned to $2\mathrm{N}_u+1$, $2\mathrm{N}_v+1-3$, and $2\mathrm{N}_w+1-5$ CPs, respectively. Finally, the outer layers are processed for all three directions by collecting the first $2\mathrm{N}+8$ CPs in Stam's scheme as shown in Figures 3.11(a), (b), and (c). These steps assign indices to $2\mathrm{N}_u+8$, $2\mathrm{N}_v+8-4$, and $2\mathrm{N}_w+8-7$ CPs, respectively.

From these indices, the $M \times K$ subdivision matrix $\bar{\mathbf{A}}$ is constructed in analogy to the surface case (see Section 2.1.2), where K and M are the total number of CPs in the original and the subdivided local control mesh, respectively. To calculate the eigenstructure, $\bar{\mathbf{A}}$ is transformed into a square $K \times K$ matrix $\mathbf{A}$ by removing the last $M-K$ rows. As shown in Figure 3.11(d), the rows that are removed from the matrix correspond to the three outer layers of the subdivided control mesh, consisting of the following number of control points:

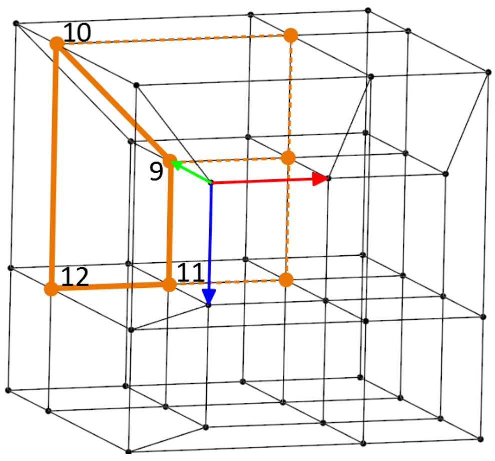
$$M - K = (2\mathrm{N}_u + 17) + (2\mathrm{N}_v + 17 - 5) + (2\mathrm{N}_w + 17 - 9) \tag{3.12}$$



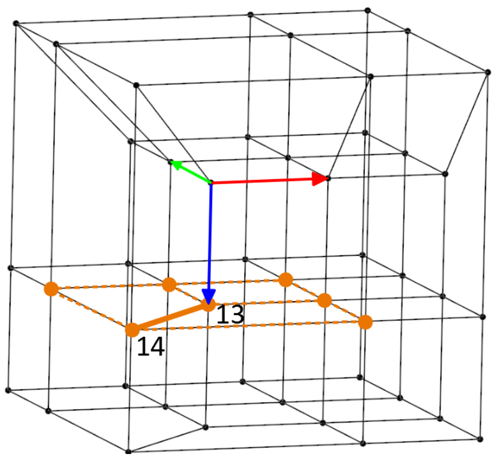
(a) Fully irregular part



(b) Inner layer in u direction

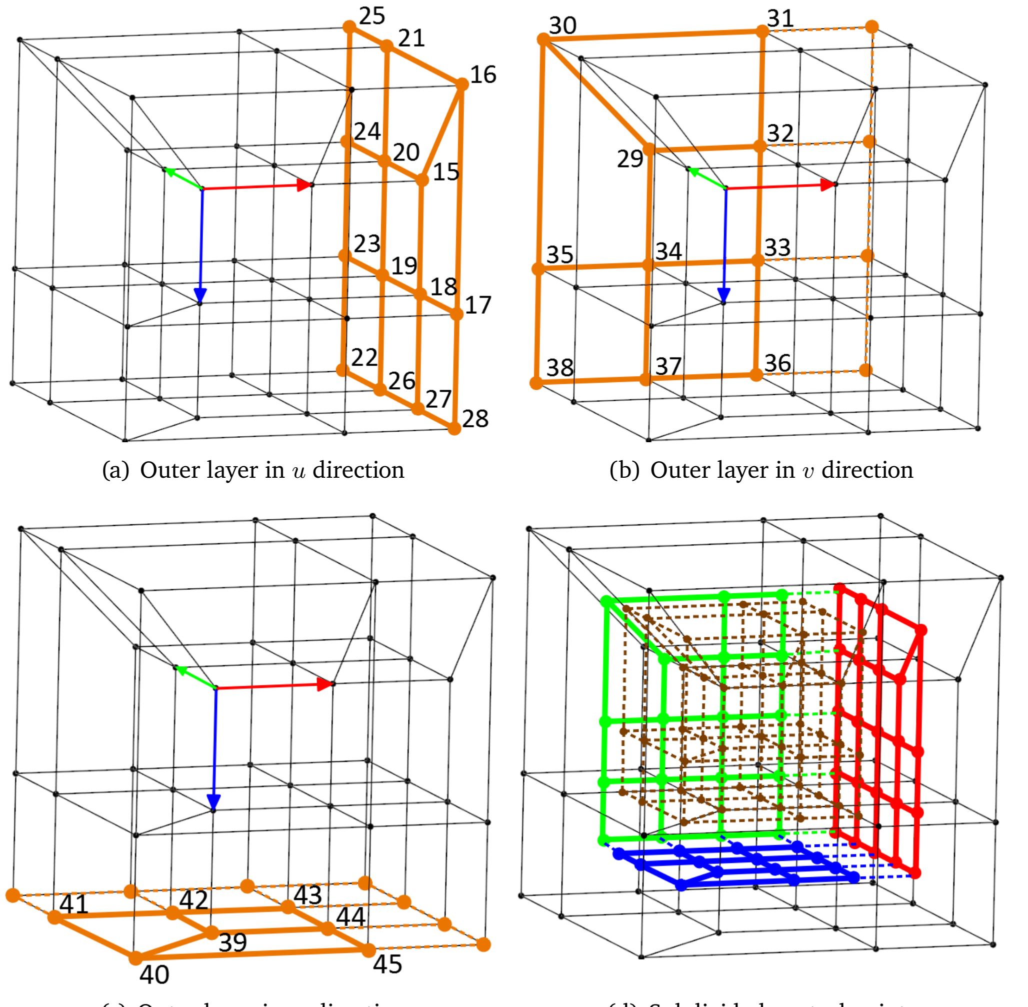


(c) Inner layer in v direction



(d) Inner layer in w direction

Figure 3.10.: First four steps for consistently numbering the control points of irregular cells. The example shows an irregular cell with three EEs of valence $\mathrm{N}_u = \mathrm{N}_v = \mathrm{N}_w = 3$. The local (u, v, w) coordinate system is shown in red (u axis), green (v axis), and blue (w axis). The control points that are added in each step are highlighted in orange and show their index in the corresponding figure. (c) and (d) also show previously collected control points connected with thin orange lines for illustrating the analogy to Stam's 2D numbering scheme shown in Figure 2.8.



(a) Outer layer in u direction

(b) Outer layer in v direction

(c) Outer layer in w direction

(d) Subdivided control points

Figure 3.11.: Last three steps (a, b, c) of the consistent numbering scheme for irregular CC-solid cells. The first four steps are shown in Figure 3.10. Again, previously collected control points are connected with thin orange lines for illustrating the analogy to Stam's 2D numbering scheme. (d) shows the subdivided control points created by applying the subdivision matrix $\bar{\mathbf{A}}$. The three outer layers highlighted in red (u), green (v), and blue (w) are not part of the squared subdivision matrix $\mathbf{A}$.

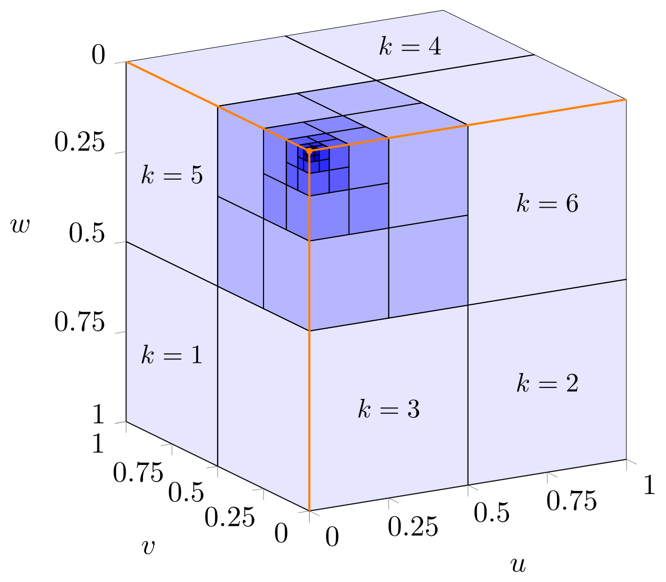


Figure 3.12.: Partitioning of the parameter space during local subdivision. Potential EEs as well as the potential EV are marked in orange. If only one EE exists in the cell to be evaluated, the w axis will be oriented along this EE.

Partitioning and Direct Evaluation

Locally subdividing an irregular cell results in a partitioning around the extraordinary vertex with eight *sub-cells* per subdivision. This process is depicted in Figure 3.12. The sub-cells are numbered with $k = 0, .., 7$. While in the surface case, 3 out of 4 sub-patches become regular (see Section 2.1.2), only four out of eight sub-cells become regular in the volumetric case ($k = 0, 1, 2, 4$). The sub-cells $k = 3, 5, 6$ may still contain EEs, while $k = 7$ features the same topological configuration as the original cell.

With a naïve approach, one has to subdivide until the point to be evaluated lies inside a fully regular sub-cell. If the point is close to an EE but not close to the EV, e.g. $(u, v, w) = (0.01, 0.01, 0.33)$, the local (u, v, w) coordinate system has to be rearranged in order to continue the subdivision process. This invalidates the local numbering scheme and requires the use of multiple different eigenstructures, resulting in additional matrix-matrix multiplications – and therefore in a non-constant-time behavior.

As also observed by Bajaj et al., repeated subdivision steps form layered structures in the neighborhood of extraordinary edges [Baj+02]. While fully irregular constellations as in Figures 3.9(a) and (c) feature multiple EEs in a single cell, layered control meshes consist of four layers of an irregular 2D control mesh as in Figures 3.9(b) and (d). Within each layer, the valence of the EE is always equal to the valence of the 2D EV. Similar to Bajaj et al.'s approach for their MLCA subdivision scheme [Baj+02], the trivariate basis functions are constructed using a tensor product of the bivariate subdivision basis functions $\varphi(u,v)$ for the corresponding two-dimensional configuration and a regular cubic B-spline basis function $N_i(w)$. The resulting trivariate basis function $N_{i,j}(u,v,w)$ corresponds to the j-th control point in layer i of the local control mesh:

$$N_{i,j}(u,v,w) = \varphi_j(u,v)N_i(w) \qquad i = 0,\ \dots, 3,\ j = 0,\ \dots, K-1 \tag{3.13}$$

K denotes the number of control points in each layer. The bivariate subdivision basis functions $\varphi(u,v)$ are constructed using Stam's constant-time limit evaluation technique for Catmull-Clark surfaces [Sta98a] as described in Section 2.1.2.

Given by the partitioning shown in Figure 3.12, the number of local subdivision steps n required for the evaluation is defined as follows:

$$n = \lceil \min\left\{-\log_2(u),\ -\log_2(v),\ -\log_2(w)\right\}\rceil \tag{3.14}$$

As the (u,v,w) coordinate system does not have to be rearranged, the eigenstructure of the subdivision matrix $\mathbf{A}$ can be used. The required local subdivisions are performed with the diagonal matrix Λ^{n-1} containing the eigenvalues of $\mathbf{A}$:

$$C_n = \bar{\mathbf{A}}\mathbf{V}\Lambda^{n-1}\mathbf{V}^{-1}C_0 \tag{3.15}$$

Finally, the volumetric limit points are evaluated using the trivariate basis functions N and the eigenstructure of the subdivision matrix:

$$\begin{aligned} e(u,v,w) &= C_0^T\mathbf{V}^{-T}\Lambda^{n-1}\mathbf{V}^T\bar{\mathbf{A}}^T\mathbf{P}_k^T N\left(\phi_{k,n}(u,v,w)\right) \\ &=: C_0^T\varphi(u,v,w) \end{aligned} \tag{3.16}$$

$\mathbf{P}_k$ denotes the volumetric picking matrix with $k = 0,\dots,6$ defined analogously to the 2D case. $\phi_{k,n}$ maps (u,v,w) into the local parameter space of sub-cell k at subdivision level n. I call $\varphi(u,v,w)$ the *trivariate subdivision basis functions*.

3.2.3. Boundary Cells

Boundary cases that usually do not arise for subdivision surfaces are omnipresent in volumetric subdivision models. Embedding a subdivision volume in $\mathbb{R}^3$ requires the model to have cells with one or multiple boundary faces. As the topology of boundary cells differs significantly from the interior cells, the evaluation of such cells must be examined thoroughly. In the following, I present a method for evaluating the limit volume of CC-solid boundary cells in constant time, utilizing the so-called *crease basis functions*. These basis functions are derived from the geometric concept of *ghost/phantom points* presented by Schweitzer [Sch96b] and by Havemann [Hav05]. Figure 3.13 shows four distinct boundary cases that have to be handled individually.

In the first case, depicted in Figure 3.13(a), the top layer represents the boundary. The mesh features a $4\times4\times3$ configuration. As stated above, boundary faces in the trivariate case can be treated analogously to crease edges in the bivariate case. Thus, the boundary can be represented by a corresponding 2D configuration of crease edges (see Figure 3.13(c)). For direct evaluation of the highlighted boundary cell, the corresponding regular B-spline basis function is substituted by the crease basis function C. Each of the three individual crease functions (see Section 2.1.2) corresponds to one layer of the control mesh, where the first function, starting at value 1, corresponds to the boundary layer. The new boundary basis functions $B_i(u,v,w)$ then write as follows:

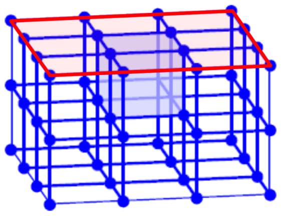
$$B_i(u,v,w) = N_{\lfloor i/16 \rfloor}(u)\; N_{\lfloor i/4 \rfloor \bmod 4}(v)\; C_{i \bmod 4}(w) \tag{3.17}$$

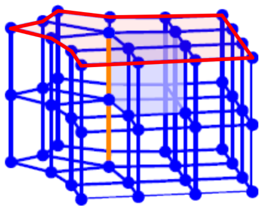
The second case, depicted in Figure 3.13(b), contains an extraordinary edge orthogonal to the boundary. Beside at its boundary, the mesh features the same layered configuration as the mesh in Figure 3.9(d). Therefore, the direct evaluation is performed similarly to case (1) but using irregular 2D subdivision basis functions in u and v, combined with the crease basis functions in the w dimension:

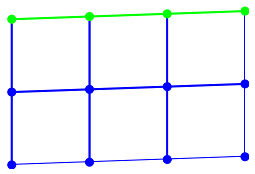
$$B_{i,j}(u,v,w) = \varphi_j(u,v)C_i(w) \qquad i = 0,1,2,\ j = 0,\ \ldots, K-1 \tag{3.18}$$

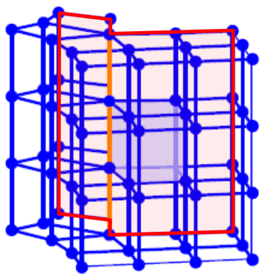
In the third boundary case shown in Figure 3.13(d), the EE is part of the boundary. The 2D configuration with the corresponding crease edges is shown in Figure 3.13(f). Calculating the subdivision basis functions for the corresponding 2D configuration in u, v and multiplying a regular B-spline basis function in the w dimension as in Equation (3.13), results in the corresponding trivariate basis functions that describe the limit.

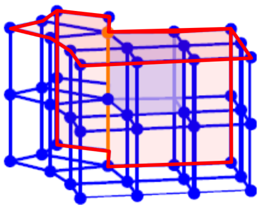
In the fourth case shown Figure 3.13(e), the top layer as well as the backside are covered with boundary faces. This boundary case is evaluated in analogy to the non-boundary irregular case (see Section 3.2.2), by using the eigenstructure of the local subdivision matrix to implicitly perform n local subdivision steps. After those, the sub-cell containing the target point corresponds to one of the boundary cases (1) - (3) and the direct evaluation can be performed accordingly. Volumetric limit points inside such boundary cells can finally be evaluated using Equation (3.16).


(a) Boundary case (1)


(b) Boundary case (2)


(c) Corresponding 2D configuration


(d) Boundary case (3)


(e) Boundary case (4)

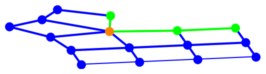
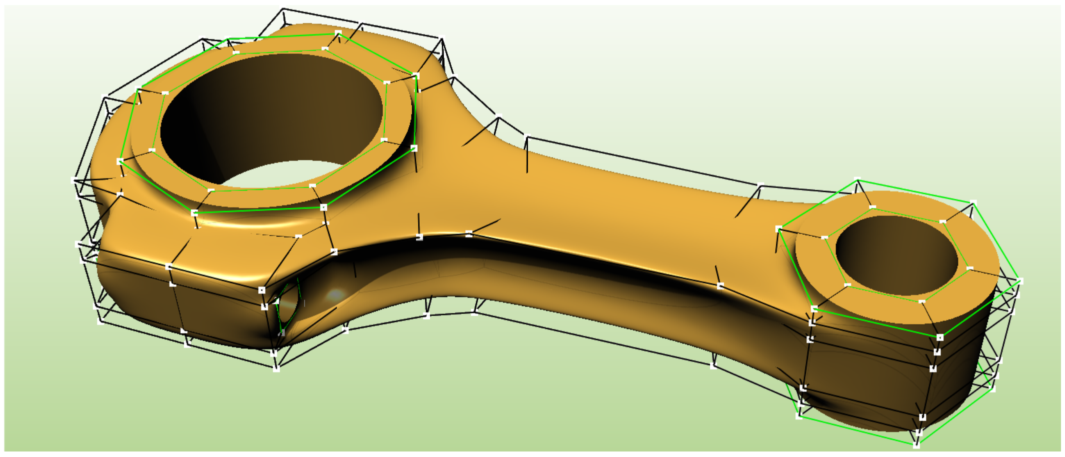
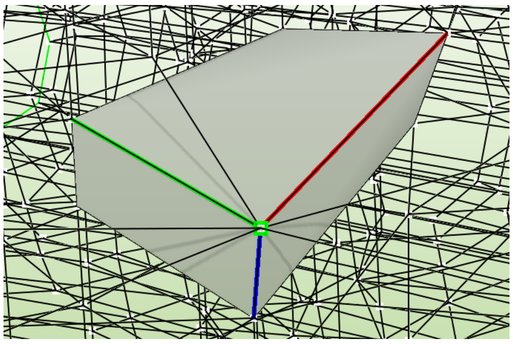

(f) Corresponding 2D configuration

Figure 3.13.: Four different boundary cases, consisting of a regular structure with one boundary layer (a), an irregular structure with one boundary layer on top (b), an irregular structure with an EE as part of the boundary (d) and an irregular structure with boundaries orthogonal to, as well as alongside the EE (e). The two-dimensional configuration corresponding to the cases (a) and (b) is depicted in (c). The one for (d) and (e) is depicted in (f). The boundary is shown in red, crease edges in green, EEs and EVs in orange.

If a hexahedral cell contains boundary faces on opposite sides, the control mesh has to be subdivided once. The resulting cells automatically form one of the four boundary cases presented here. Additionally, the method supports crease edges on the boundary of the model. Again, it is assumed that only edges radiating from the EV are defined as crease edges. Figure 3.14 shows a complex CC-solid model of a connecting rod, containing


(a) Connecting rod model


(b) Control cell with local coordinate system

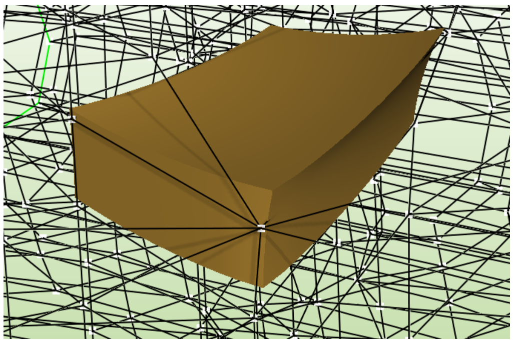

(c) Limit cell

Figure 3.14.: Limit evaluation of a complex CC-solid model of a connecting rod. The control mesh contains several EEs, EVs, as well as crease edges. (b) and (c) show the evaluation of an irregular cell having two EEs, one of which is also part of the boundary. (b) also shows the cell's local u (red), v (green), and w axis (blue).

multiple EEs, EVs and crease edges. All cells can be evaluated in constant time, using the approach presented here.

In addition to the 3D positions of the limit points, the *limit values* of scalar properties stored per control point can be evaluated as well. As in the two-dimensional case, the method also supports the calculation of derivatives.

3.2.4. Implementation and Performance

The constant-time limit evaluation approach presented here enables efficient evaluation of Catmull-Clark subdivision solids at arbitrary parameter values.

The algorithm for evaluating irregular cells consists of the following three steps:

1. Construction of the subdivision matrix
2. Calculation of its eigenstructure
3. Evaluation with the subdivision basis functions

As for the surface case, the subdivision matrix and its eigenstructure only have to be calculated once for each local control mesh configuration. In the volumetric case, these configurations are characterized by the valences of the extraordinary edges, as well as the arrangement of boundary faces and crease edges for the corresponding cell. Using a hash function that takes these arguments, every topological configuration is mapped to the corresponding subdivision matrix and its eigenstructure, enabling fast look-up during the evaluation process. The actual evaluation consists of four matrix-matrix multiplications and the exponentiation of a diagonal matrix as shown in Equation (3.16).

I measured the time required for each part of the algorithm for three different topologies. They have been designed to include an irregular interior cell, a boundary cell of case (4) (see Section 3.2.3) and a corner cell with three boundary faces. Table 3.1 lists the absolute and relative time required for each part. Since calculating the subdivision matrix and its eigenstructure accounts for a major portion of the computational cost, precomputation significantly speeds up the process.

For comparison, I also implemented the subdivision-based limit evaluation approach by Burkhart et al. [BHU10b]. Table 3.2 shows the inter-dependencies between the parameter values (u, v, w), the parameters k and n (see Section 3.2.2), as well as the number of subdivision steps and required coordinate system rearrangements for the Burkhart et al.'s approach [BHU10b].

Topology	**Subdiv. Matrix**	**Eigen-structure**	**Evaluation**	**Total**
Interior cell N = 5	0.02 ms (0.13%)	15.37 ms (99.10%)	0.12 ms (0.77%)	15.51 ms
Boundary case (4) N = 5	0.35 ms (5.62%)	5.81 ms (93.26%)	0.07 ms (1.12%)	6.23 ms
Corner boundary case	0.25 ms (14.62%)	1.42 ms (83.04%)	0.04 ms (2.34%)	1.71 ms

Table 3.1.: Time required for each part of the direct evaluation algorithm. Calculating the subdivision matrix and the eigenstructure are the largest parts but they can be precomputed and cached for the evaluation.

For Burkhart's approach, the computational costs increase linearly with the number of subdivisions required for the evaluation. In contrast, the direct approach evaluates each limit point in around 0.072 milliseconds, regardless of k and n. As shown in Table 3.2, iteratively performing the subdivisions is faster for less than three subdivision steps. However, whenever more than two subdivision steps are required, the direct method starts to outperform the subdivision-based approach. If the (u, v, w) coordinate system does not have to be rearranged, the break-even point is at an average of 2.57 subdivisions. Moreover, my approach results in a speedup of $2\times$ for six subdivisions and a speedup of $3\times$ for nine subdivisions.

(u, v, w)	k	n	**Direct Evaluation**	**Required Subdivisions**	**Req. Rear-rangements**	**Subdiv. Evaluation**
(0.51, 0.51, 0.51)	0	1	0.071 ms	1	0	0.024 ms
(0.01, 0.26, 0.26)	2	2	0.073 ms	2	0	0.046 ms
(0.51, 0.51, 0.15)	4	1	0.070 ms	3	0	0.085 ms
(0.01, 0.01, 0.01)	0	7	0.074 ms	7	0	0.170 ms
(0.01, 0.01, 0.33)	3	2	0.072 ms	7	5	0.196 ms

Table 3.2.: Time required for evaluating different parameter values u, v, w with my direct evaluation approach as well as with the non-direct, subdivision-based approach by Burkhart et al. [BHU10b], for a cell with boundary case (4) and $\mathrm{N} = 5$ as shown in Figure 3.13(e). The table also includes the values for k and n, as well as the number of subdivision steps and coordinate system rearrangements for the subdivision-based approach.

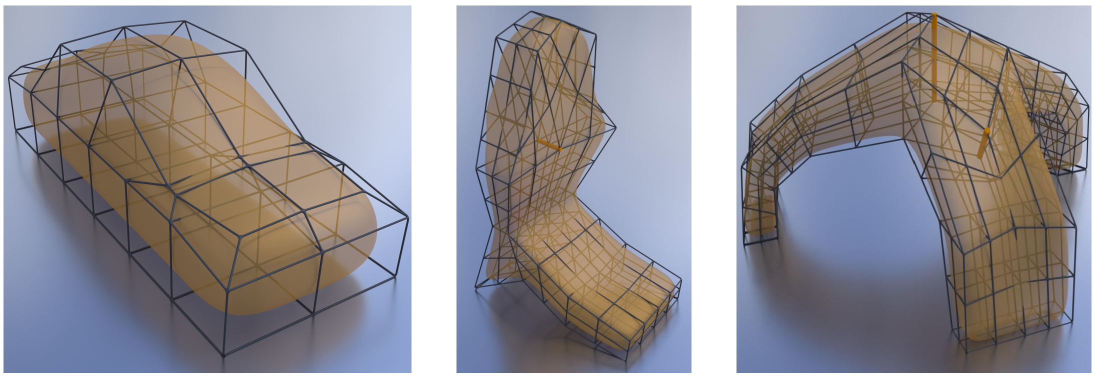
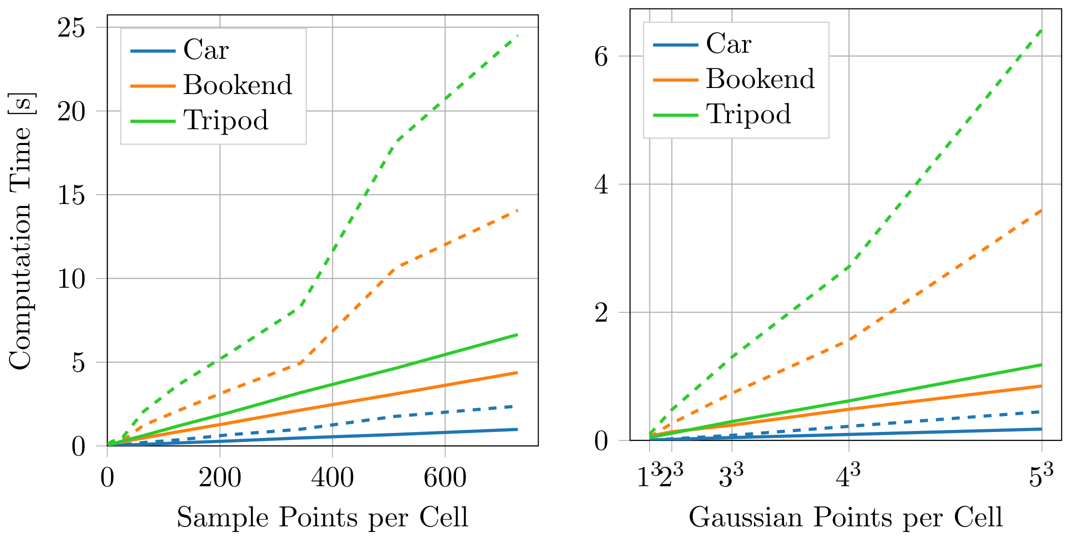

(a) Car model (b) Bookend model (c) Tripod model

Figure 3.15.: Visualization of the three models that were used for the performance measurements.

I further assessed my method using the three example models shown in Figure 3.15. The *car* model (a) consists of 24 regular boundary cells. The *bookend* model (b) is made up of 88 subdivision cells, with two EEs of valence 5 (10 irregular cells). The *tripod* model (c) has a total of 136 cells, with two EEs of valence 3 and two EEs of valence 5 (16 irregular cells). I evaluated their limit at an increasing number of regular sample points (motivated by applications such as slicing e.g. for 3D printing), as well as at Gauss-Legendre quadrature points (motivated by the application of subdivision solids in physically based simulations) using my direct evaluation technique as well as my implementation of the subdivision-based approach by Burkhart et al. [BHU10b]. Figure 3.16 shows the results of the performance measurements for both experiments. As can be seen, the direct evaluation approach scales linearly with the number of sample points, evaluating every single limit point in constant time, while the subdivision-based technique shows non-linear performance. However, when just one or two subdivision step are required, performing the subdivisions explicitly is faster than applying the eigenstructure matrices $\mathbf{V}$, Λ, and $\mathbf{V}^{-1}$.

The method presented in this section works for all polyhedral control meshes that are subdivided into hexahedra after one step of volumetric Catmull-Clark subdivision. However, not all types of polyhedra can be subdivided into hexahedra, as explained in Section 3.1.



(a) Regularly distributed sample points (b) Gauss-Legendre quadrature points

Figure 3.16.: Computation time for evaluating the limit at regular sampling points (a), as well as at Gauss-Legendre quadrature points (b) using three different CC-solid models (car, bookend, tripod). I compare my direct evaluation approach (visualized with solid lines) with the subdivision-based approach by Burkhart et al. [BHU10b] (dashed lines).

3.3. Volumetric Modeling Operations

The content of this section is based on the following papers:

Implicit Mesh Generation using Volumetric Subdivision
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner

Published in the *13th Workshop on Virtual Reality Interaction and Physical Simulation*
2017
The Eurographics Association

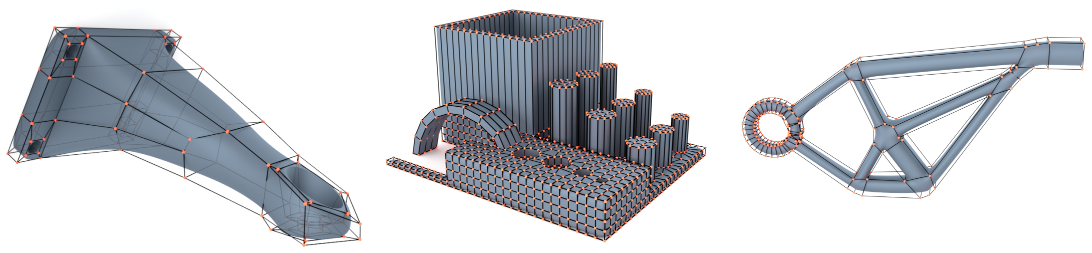
Continuous Property Gradation for Multi-Material 3D-Printed Objects
C. Altenhofen, T. H. Luu, T. Grasser, M. Dennstädt, J. S. Mueller-Roemer, D. Weber, and A. Stork

Published in *Solid Freeform Fabrication Symposium*
Volume 29, 2018
The Minerals, Metals & Materials Society TMS

Common tools for polygonal modeling used in design and computer animation, such as Blender [Fou20] or Maya [Aut18], as well as commonly used CAD software such as CATIA [Das20b] or SolidWorks [Das20a], only support surface modeling. Although closed surface meshes implicitly describe a solid object, they are not considered volumetric meshes. In addition to vertices, edges and faces, a volumetric mesh consists of cells and has inner faces to separate these cells.

Different techniques can be employed to create volumetric subdivision models. *Polyhedral modeling* (freeform and block-based modeling) is the volumetric extension of the polygonal surface modeling approaches found in software such as Blender [Fou20] and Maya [Aut18]. Procedural modeling with analytical functions as well as sketch-based modeling are also possible. Established techniques can be adapted to create volumetric meshes instead of surface meshes. However, providing topologically valid volumetric meshes requires special volumetric modeling operations. Figure 3.17 shows three CC-solid models created with different modeling techniques.

In this section, I present my polyhedral modeling approach based on volumetric Euler operations.



(a) Freeform (b) Block-based (c) Sketch-based

Figure 3.17.: Three volumetric subdivision models created with different modeling techniques. (a) has been created with freeform polyhedral modeling, while (b) has been created with a block-based approach. (c) is the result of sketching.

3.3.1. The Half-Face Data Structure

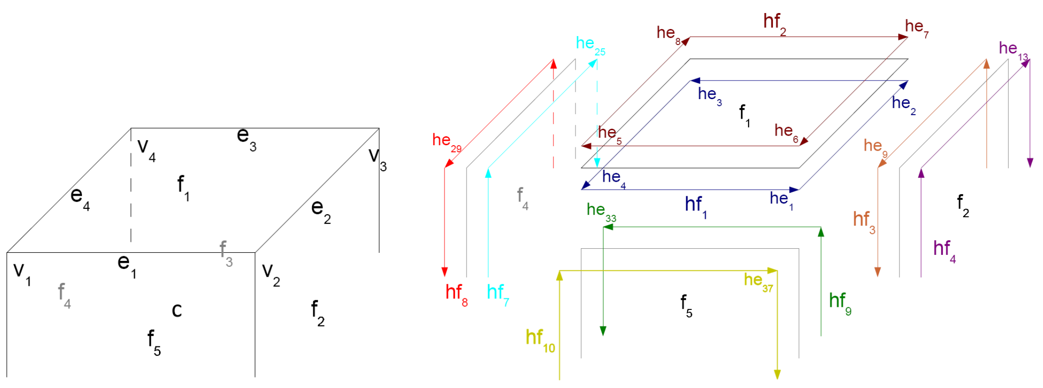
For the internal representation of Catmull-Clark solids, I implemented a pointer-based *half-face* data structure. It can be seen as a volumetric extension to the well-known *half-edge* data structure. Faces are represented as a pair of two half-faces and an additional entity is introduced to represent cells. Similar to half-edges, every half-face holds a reference to its opposing half-face and to the previous and the next half-face inside the loop of half-faces that defines the corresponding cell. Iterating over these references in different ways allows for fast computation of all relevant neighboring relations needed for volumetric modeling, volumetric subdivision, and constant-time volumetric limit evaluation. The data structure is designed in a flexible way, enabling the representation of any type of polyhedron.

In their work on trivariate spline models, Elber et al. also use a half-face approach for their volumetric representation *V-Rep* [ME16; EDE17]. CGAL offers a similar data structure called *Linear Cell Complex* or more general *Combinatorial Map* [Jam+15]. OpenVolumeMesh (OVM) also provides an implementation of a half-face data structure [KBK13]. Mueller-Roemer et al. presented an index-based half-face data structure for GPUs [MAS17]. Although achieving great performance, the data structure is designed for massively parallel processing, instead of modifying individual vertices, edges, faces or cells.

The main difference between these half-face data structures and my implementation is the representation of half-edges. The data structures mentioned above split every edge into two half-edges. Every half-face connected to a given edge uses either the one or the other half-edge, depending on its orientation. My data structure splits every edge into one half-edge per connected half-face. Although increasing the memory footprint as shown in Table 3.3), this allows for iterating over all connected half-faces more efficiently, using the *radial* and *opposite* half-edges (see below). Additionally, as my data structure is based on pointers rather than on indices, updating unchanged parts of the mesh is not necessary when modifying the mesh topology by inserting or deleting elements. The references just have to be updated locally. This results in a minimal computational effort when interactively modeling volumetric 3D objects.

Within my half-face data structure, volumetric polyhedral models are constructed from vertices, half-edges, edges, half-faces, faces, and cells. Every entity stores the following relations to its related entities:

- Vertex v
  - a list of all outgoing half-edges
- Half-Edge he
  - its starting vertex
  - its edge
  - its half-face
  - its *previous* and *next* half-edge in the loop defining its half-face
  - its *opposite* half-edge on the same edge, but belonging to the other face connected to that edge in the same cell
  - its *radial* half-edge on the same edge, but on the opposite half-face – on the same face, but in the neighboring cell
- Edge e
  - one of its half-edges
- Half-Face hf
  - its face
  - its cell
  - one of its half-edges, which define the loop inside this half-face
  - its *previous* and *next* half-face in the loop defining its cell
  - its *opposite* half-face in the same face, but in the neighboring cell
- Face f
  - its two half-faces
- Cell c
  - one of its half-face, which define the loop inside this cell



(a) Upper part of a hexahedral cell (b) Half-faces and half-edges of (a)

Figure 3.18.: Elements of the half-face data structure for representing the upper part of a hexahedral cell. (a) shows the vertices v_i, edges e_i, faces f_i and the cell c that define the hexahedron. (b) shows the corresponding half-edges he_i and half-faces hf_i. As can be seen, every half-face has its own loop of half-edges. Half-faces and half-edges defining face f_3 are left out for improved visibility.

Figure 3.18 shows the individual components defining the upper part of a hexahedral cell. The hierarchy of the different entities is given by the design of the data structure as follows:

$$v \leftrightarrow he/e \leftrightarrow hf/f \leftrightarrow c \tag{3.19}$$

Relations that only span two neighboring levels in the hierarchy, e.g. collecting all edges or half-edges of a face or half-face, can be performed straight-forwardly. Relations that span multiple levels, such as collecting all cells of a vertex or all half-edges of a cell, require additional computations. For operations that modify a large number of elements, such as loop-cutting or global refinement, computing the required relations for the entire mesh is more efficient than computing them individually for each element affected by the operation. However, storing all the neighborhood information results in a high memory footprint, especially for complex models.

Memory Footprint and Performance

For evaluating the memory footprint as well as the performance of my half-face data structure, multiple CC-solid models have been subdivided and queried for frequently used neighborhood relations. Table 3.3 shows the memory consumption as well as the computation time for performing the queries with my data structure as well as with OpenVolumeMesh [KBK13] for three different models. The queries comprise getting all cells connected to a given vertex, as well as the inverse operation of getting all vertices belonging to a given cell.

Although my data structure requires about $12.7\times$ as much memory as OVM, it is faster when querying neighborhood information. For querying the bottom-up relations of cells connected to a given vertex, my data structure provides an average speedup of $1.7\times$. For the top-down relations, it is faster by an average of $2.8\times$.

Model	n	**Cells**	**Memory**		**Vertex-Cells [ms]**		**Cell-Vertices [ms]**	
			OVM	**Mine**	**OVM**	**Mine**	**OVM**	**Mine**
Coyote	0	48	11.5 kB	141 kB	0.06	0.06	0.12	0.01
	1	354	74.2 kB	938 kB	0.42	0.26	0.96	0.26
	2	2832	533 kB	6.63 MB	3.84	3.06	7.1	3.24
	3	22.6k	3.94 MB	50.3 MB	31.84	27.86	61.36	25.5
	4	181k	30.6 MB	391 MB	248.5	201.1	422.9	177.3
	5	1.4M	241 MB	3.01 GB	1986	1599	3361	1399
Engine Mount	0	48	12.6 kB	158 kB	0.10	0.02	0.1	0.02
	1	390	82.7 kB	1.02 MB	0.56	0.3	0.92	0.14
	2	3120	592 kB	7.37 MB	3.60	3.26	7.12	3.4
	3	25.0k	4.36 MB	55.7 MB	32.2	28.78	57.88	26.34
	4	200k	33.8 MB	432 MB	276.7	224.0	465.3	209.7
	5	1.6M	266 MB	3.32 GB	2201	1764	3721	1673
Connect-ing Rod	0	146	32.2 kB	407 kB	0.18	0.0	0.36	0.14
	1	1132	223 kB	2.77 MB	1.42	0.8	2.7	0.82
	2	9056	1.61 MB	20.6 MB	11.32	10.52	21.3	9.96
	3	72.4k	12.4 MB	158 MB	101.7	82.8	172.4	77.30
	4	580k	97.1 MB	1.21 GB	808.2	656.4	1350	617.9

Table 3.3.: Comparison of my half-face data structure with OpenVolumeMesh (OVM) [KBK13]. For both data structures, three different CC-solid models have been stored and queried for frequently used neighborhood information on multiple subdivision levels n.

3.3.2. Volumetric Euler Operators

When creating and modifying volumetric meshes, e.g. control meshes for CC solids, a set of basic modeling operations is required. Following the concept of *Euler operators*, these operations insert or remove topological elements, such as vertices, edges, or faces, while keeping the mesh *consistent* for every operation. *Consistency* refers to the fact, that e.g. a vertex cannot be inserted without inserting at least one edge. More generally, *consistency* during modeling is define by the Euler characteristic $\chi(\mathbf{K})$ for a cell complex $\mathbf{K}$ (see also the text book by Armstrong [Arm13]):

$$\chi(\mathbf{K}) = \sum_{k=0}^{\infty}(-1)^k \alpha_k \tag{3.20}$$

α_k defines the number of elements of dimension k in the mesh. For volumetric meshes, α_0 refers to the number of vertices, α_1 to the number of edges, α_2 to the number of faces, and α_3 to the number of cells.

Resulting from Equation (3.20), for consistent, closed surfaces meshes, the Euler-Poincaré formula [Arm13] has to hold:

$$V - E + F = H + 2(S - G) \tag{3.21}$$

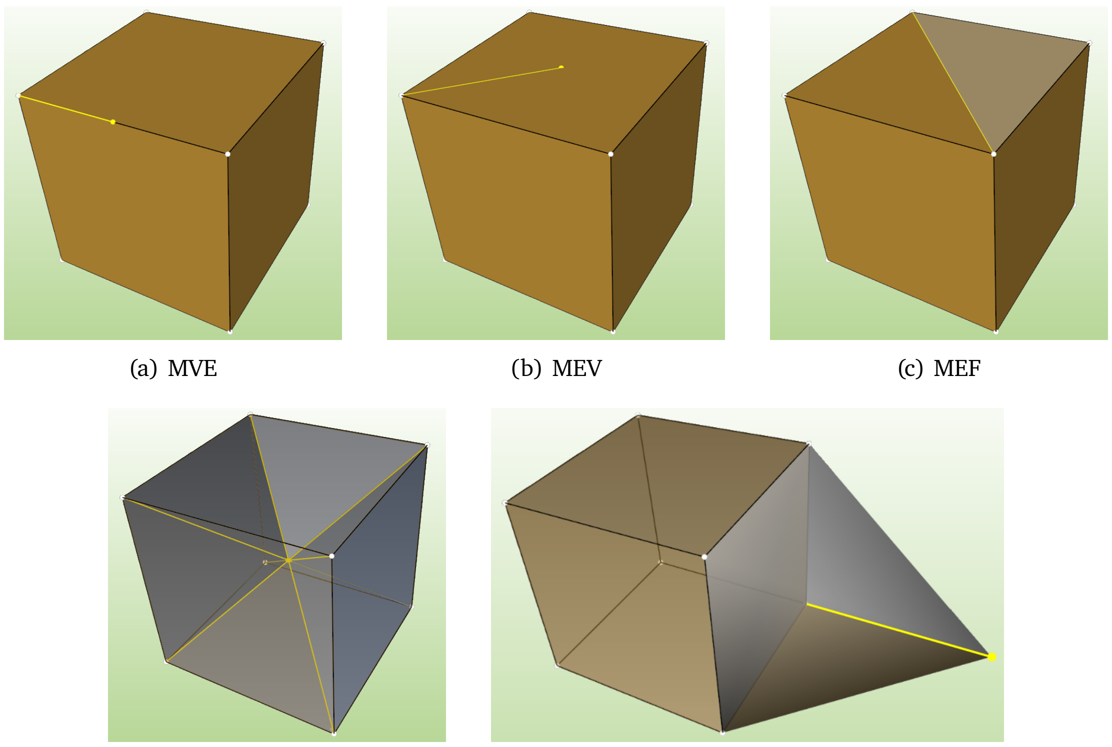
with V, E, and F being the number of vertices, edges and faces, respectively. H denotes the number of holes, while S denotes the number of shells, and G is the *genus* of the mesh. For a surface mesh of a cube with $V = 8$ vertices, $E = 12$ edges, and $F = 6$ faces, this leads to an Euler characteristic of $8 - 12 + 6 = 0 + 2(1 - 0) = 2$.

When handling volumetric meshes, $\chi(\mathbf{K})$ is defined as follows:

$$\chi(\mathbf{K}) = \sum_{k=0}^{3} \alpha_k = \alpha_0 - \alpha_1 + \alpha_2 - \alpha_3 = V - E + F - C = 1 \tag{3.22}$$

For a volumetric mesh of a cube with $V = 8$ vertices, $E = 12$ edges, $F = 6$ faces, and $C = 1$ cell, the Euler characteristic is $\chi = 1$.

Assuming that $V - E + F - C = 1$ holds before applying the corresponding operator, the modeling operators presented here keep the Euler characteristic of the mesh constant and are therefore Euler operators. In the following, I present five Euler operators and their complementary operators for creating and deleting vertices, edges, faces and cells, keeping the topology of the volumetric mesh valid. More complex modeling operations, such as



(a) MVE (b) MEV (c) MEF

(d) MVC (e) MEVFC

Figure 3.19.: Five different Euler operators applied to a volumetric mesh of a cube, consisting of 8 vertices, 12 edges, 6 faces and one cell. While the first three operators (a-c) are adopted from surface modeling, the last two can be considered fully volumetric Euler operators as they actually create new cells. Newly created vertices, edges, faces, and cells are highlighted.

the ones implemented in the integrated modeling and simulation framework presented in Section 6.1, can be performed by applying multiple of these operators sequentially. All volumetric control meshes used in this dissertation were created using modeling operations based on these five Euler operators.

The first three Euler operators, MVE, MEV, and MEF, as well as their complementary operators KVE, KEV, and KEF, are analog to the classical Euler operators for surface meshes. A description of surface Euler operators can also be found in the work by Eastman and Weiler [EW79]. The last two operators are special operators for volumetric meshes.

1. MakeVertexEdge MVE: This operator creates a new Vertex in an existing edge, splitting the edge in two. Therefore, the operator also automatically creates a new edge. Figure 3.19(a) shows the result. The new vertex and the new edge are highlighted. The complementary operator is *KillVertexEdge (KVE)*.

The Euler characteristic of MVE is given by:

$$(V+1)-(E+1)+F-C=V-E+F-C+1-1=1 \tag{3.23}$$

Accordingly, the Euler characteristic of KVE is given by:

$$(V-1)-(E-1)+F-C=V-E+F-C-1+1=1 \tag{3.24}$$

2. MakeEdgeVertex MEV: This operator inserts a new vertex into an existing face. Since the new vertex has to be connected to at least one edge, the operator also creates a new edge, connecting the new vertex to an existing vertex of the face. Figure 3.19(b) shows the result. Again, the new vertex and the new edge are highlighted. The complementary operator is *KillEdgeVertex (KEV)*.

The equations for the Euler characteristic for MEV and KEV are the same as for MVE (Equation (3.23)) and KVE (Equation (3.24)), respectively.

3. MakeEdgeFace MEF: This operator inserts a new edge that connects two existing vertices of a face. As the face is split into two faces by the new edge, the operator also automatically creates a new face. Figure 3.19(c) shows the result, with the new edge and the new face being highlighted. The complementary operator is *KillEdgeFace (KEF)*.

The Euler characteristic for MEF is given by:

$$V-(E+1)+(F+1)-C=V-E+F-C+1-1=1 \tag{3.25}$$

while the Euler characteristic of KEF is given by:

$$V-(E-1)+(F-1)-C=V-E+F-C+1-1=1 \tag{3.26}$$

4. MakeVertexInCell MVC: This operator creates a new vertex inside an existing cell and connects the vertex to all existing vertices of that cell by creating new edges. Since the cell is split into multiple cells during the process, the operator also creates all the new faces and cells. Figure 3.19(d) shows the result, with all new elements being highlighted. The complementary operator is *KillVertexInCell (KVC)*.

Let V_c be the number of vertices, E_c the number of edges, and F_c the number of faces of cell c in which the new vertex will be inserted. Performing MVC on a cell c creates one new vertex, V_c new edges, E_c new faces, and $F_c - 1$ new cells. This results in an Euler characteristic of:

$$\begin{aligned}&(V+1)-(E+V_c)+(F+E_c)-(C+F_c-1)\\&=V-E+F-C+1-V_c+E_c-F_c+1\\&=V-E+F-C+1-(V_c-E_c+F_c-1)\end{aligned} \tag{3.27}$$

The original cell fulfilled the Euler characteristic:

$$V_c-E_c+F_c-1=1 \tag{3.28}$$

Therefore, inserting Equation (3.28) into Equation (3.27) results in the following after having performed MVC:

$$V-E+F-C+1-1=1 \tag{3.29}$$

When performing the reverse operation of KVC, let V_n, E_n, F_n, and C_n be the vertices, edges, faces, and cells directly connected to the vertex to be removed. Let elements with index c be the elements connected to the cells to be removed and elements with index r be part of the remaining volumetric mesh. This results in:

$$\begin{aligned}V&=V_r+V_c+1\\E&=E_r+E_c+E_n\\F&=F_r+F_c+F_n\\C&=C_r+C_n\end{aligned} \tag{3.30}$$

The following equivalences can be derived from the topology of the cells: E_n is equal to V_c, F_n is equal to E_c, and C_n is equal to F_c. Inserting (3.30) into (3.22) and taking into account these equivalences, gives:

$$\begin{gathered}V_r+V_c+1-(E_r+E_c+E_n)+F_r+F_c+F_n-(C_r+C_n)=1\\V_r+V_c+1-E_r-E_c-V_c+F_r+F_c+E_c-C_r-F_c=1\\V_r+1-E_r+F_r-C_r=1\\\Rightarrow V_r-E_r+F_r-C_r=0\end{gathered} \tag{3.31}$$

Removing $V_r - E_r + F_r - C_r$ from the equation gives:

$$\begin{gathered} V_c + 1 - (E_c + E_n) + F_c + F_n - C_n = 1 \\ V_c - (E_c + E_n) + F_c + F_n - C_n = 0 \\ \Rightarrow V_c - E_c + F_c = E_n - F_n + C_n \end{gathered} \tag{3.32}$$

Assuming that Equation (3.21) holds for the surface of the resulting polyhedron, consisting of only V_c, E_c, and F_c, results in:

$$V_c - E_c + F_c = 2 = E_n - F_n + C_n \tag{3.33}$$

Applying KVC removes E_n edges, F_n Faces, and $C_n - 1$ cells. Using Equations (3.31) and (3.33) ultimately results in the following for KVC:

$$\begin{gathered} (V_c + 1 - 1) - (E_c + E_n - E_n) + (F_c + F_n - F_n) - (C_n - (C_n - 1)) = 1 \\ V_c + 1 - 1 - E_c - V_c + E_n + F_c + E_c - F_n - F_c + C_n - 1 = 1 \\ E_n - F_n + C_n - 1 = 1 \\ 2 - 1 = 1 \end{gathered} \tag{3.34}$$

5. MakeEdgeVertexFaceCell MEVFC: This operator adds a new cell to an existing boundary face, also creating one new face, one edge and one vertex. Please note that the newly created face and cell have zero area and volume, respectively. The other Euler operators can be used to extend the face with new edges and the cell with new faces. Figure 3.19(e) shows the result of MEVFC, with again all new elements being highlighted. The complementary operator is *KillEdgeVertexFaceCell (KEVFC)*.

The Euler characteristic for MEFVC is given by

$$\begin{gathered} (V + 1) - (E + 1) + (F + 1) - (C + 1) \\ = V - E + F - C + 2 - 2 = 1 \end{gathered} \tag{3.35}$$

Accordingly, the Euler characteristic for KEFVC is given by

$$\begin{gathered} (V - 1) - (E - 1) + (F - 1) - (C - 1) \\ = V - E + F - C - 2 + 2 = 1 \end{gathered} \tag{3.36}$$

More complex modeling operations can be created as a combination of multiple Euler operators. For example, a new hexahedral cell can be added to the volumetric mesh by combining an MEVFC with multiple MEFs and MVEs. Combinations of different Euler operators are also used when connecting and merging cells, as well as for all operations required for performing CC-solid subdivision.

3.3.3. Local Refinement

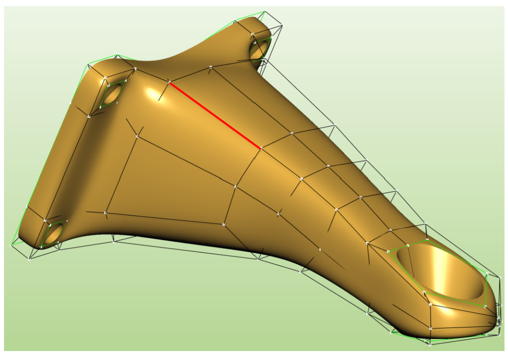
When adding geometric detail to certain areas of a given model or when increasing the resolution in certain areas of a simulation mesh, local refinement is the method of choice.

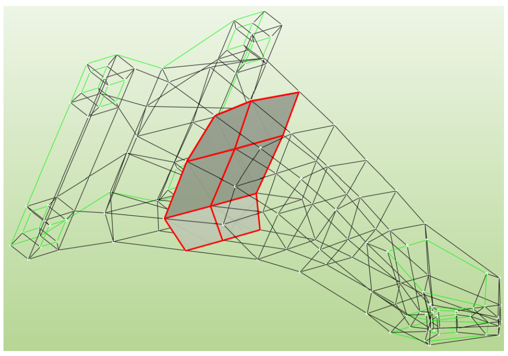
Yong et al. [YC05] presented a method for local refinement of Catmull-Clark surfaces, which can only partly be extended to CC solids. For CC surfaces, the transitions between refined and unrefined patches are realized by splitting the patch into three quadrilaterals. For CC solids, topological challenges arise when connecting refined and unrefined hexahedral cells. The connections require complex polyhedra that do not subdivide into hexahedra after a finite number of subdivision steps. Those issues are simply not present in the surface case. For CC Solids or hexahedral meshes in general, local refinement is more complicated than for e.g. tetrahedral meshes, since the refinement has to propagate to neighboring elements if T-junctions are to be avoided.

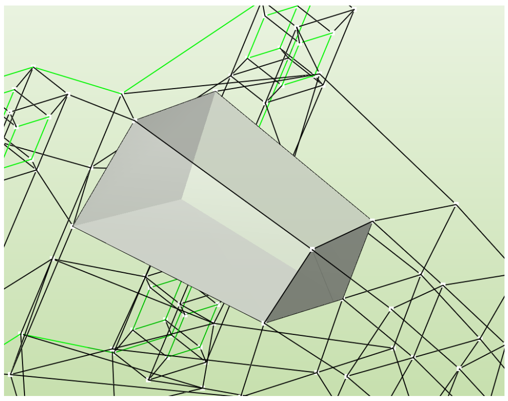
As an alternative to the approach by Yong et al. [YC05], loop-cuts can be applied to a set of connected cells, splitting them along one topological axis. Performing loop-cuts on surface meshes is a common operation in polygonal modeling tools such as Blender [Fou20]. Figure 3.20 shows the process of performing a loop-cut on a CC-solid control mesh. First, an edge is selected as the starting point of the cutting operation. The topological direction of the cut is orthogonal to the selected edge. Then, the edge itself, as well as its cells are split in half. While splitting the cells, their faces and edges are split accordingly. Next, the cut is propagated throughout the model, splitting the next set of edges, faces and cells. This step is repeated until all cells in the topological loop have been processed.

Loop-cutting can also be used to improve unbalanced aspect ratios or global differences in cell size across the mesh as shown in Figure 3.20(d). In the FEM context, equally sized elements with aspect ratios close to 1 are desired for good mesh quality (see Section 4.3). As the loop-cut propagates throughout the model in the direction of the cut, it is best used on cells that have just a few neighboring cells in the chosen direction. Otherwise, aspect ratios and cell sizes might be improved in one area and at the same time worsened in another.

Although performing loop-cuts introduces new control points, changes in the limit surface of the CC-solid model can be minimized during the cutting process. Instead of inserting the new control points at the center of every edge that is split, the CC-surface *edge rule* is applied to calculate the positions of the new control points on boundary edges (see Section 2.1.2). When performing several parallel cuts on neighboring boundary edges, the original control points connected to those edges are moved as well, using the CC-surface


(a) CC-solid model before loop-cutting


(b) Edges and faces created by the cut


(c) A control cell before loop-cutting

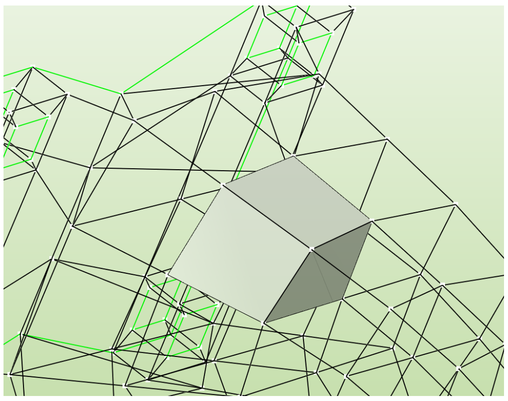
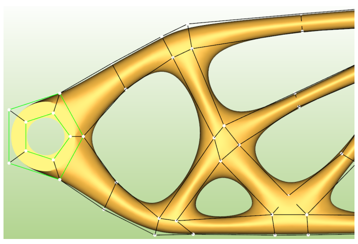
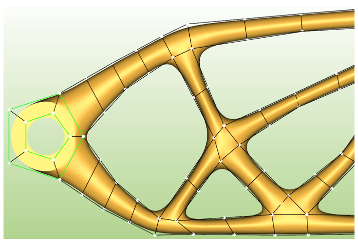
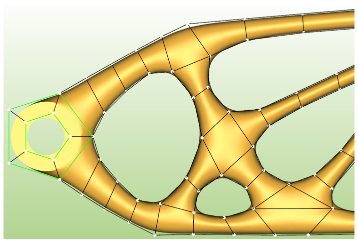

(d) One of the resulting cells after loop-cutting

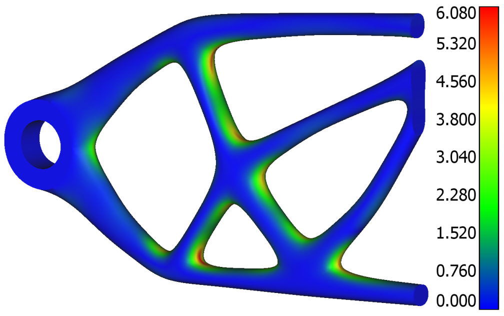
Figure 3.20.: Loop-cutting the engine mount model. (a) shows the initial model with the edge to cut being highlighted in red. (b) shows the edges and faces created by cutting the control mesh, highlighted in red and gray, respectively. (c) and (d) show the change in a cell's aspect ratio resulting from the cut.

crease vertex rule [Hav05]. Figure 3.21 shows several loop-cuts with and without applying these subdivision rules, as well as the measured distances for both methods. The model has a size of $491 \times 125 \times 15$ millimeters, while the individual struts have diameters between 10 and 20 mm. Performing the loop-cuts with the subdivision rules reduces the maximum absolute error from 6.08 to 1.10 mm for this example.

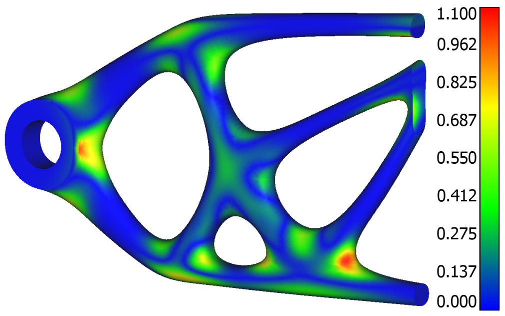

(a) Initial CC-solid model


(b) Model with several loop-cuts


(c) Loop-cuts using the subdivision rules



(d) Absolute difference between (a) and (b)



(e) Absolute difference between (a) and (c)

Figure 3.21.: Several loop-cuts on a CC-solid model. (a) shows the initial control mesh, as well as the limit surface. (b) and (c) show the model after performing the cuts splitting the edges at their center (b), or calculating the positions of the new control points using a subset of the subdivision rules, respectively (c). (d) and (e) show the absolute difference in millimeters between the limit surface before and after performing the cuts with either method. The error is color-coded from blue to red. Please note, that (d) and (e) use different maximum values for the color-mapping – 6.08 mm and 1.10 mm.

3.3.4. Parametrizing Volumetric Control Meshes

In CAD modeling, defining parameters is a frequently used technique when designing 3D objects. Lengths, angles, radii, extrusion distances, and other measures are defined as parameters in order to adapt certain features of the designed part in a controlled and consistent way. E.g., if the corresponding parameters are defined properly, changing the diameter of all screw holes in a given part from 6 to 8 millimeters can be performed with a single operation.

However, parameters such as hole diameters are most often not the kind of parameters needed for optimizing a given part for weight or stability – neither in a manual, nor in an automated optimization process. Additionally, when changing a parameter value, e.g. the length of a segment in a 2D sketch or a revolution radius, the entire CAD model has to be re-evaluated based on the history of CAD operations. For free-form geometry, such as subdivision or spline surfaces and volumes, every control point provides three degrees of freedom – its position in x, y, and z – for manipulating the shape of the object. However, manipulating the positions of all control points individually in an optimization scenario results in many parameters, increasing the risk for most optimization algorithms to terminate in local minima.

In this section, I present a new approach for defining parameters on CC-solid objects for versatilely modifying their shape with just a few degrees of freedom. The approach is based on geometric transformations relative to reference objects. These transformations are applied on groups of multiple control points, modifying their position accordingly. In contrast to parameters commonly used in CAD software, the parameters in this approach are defined after the 3D object has been modeled in its original configuration.

In order to define a parameter, a set of control points is grouped into a *parametrizable group*, which is then transformed by a *parametric operation*. A parametric operation is defined by a *reference object* as well as by a geometric transformation that is applied to every control point in the group. The approach offers – but is not limited to – four different reference objects:

- Planes
- Axes
- Points
- Segments

For these reference objects, the following geometric transformations are available:

- Translation – an additive or subtractive modification of the distance between each control point and the reference object
- Scaling – a multiplicative modification of this distance
- Rotation – a rotation of every control point around the reference object (only for axes and segments)
- Sliding – a translation of every control point along the reference object (only for segments)
- Curving – a projection of every control point from a straight segment onto a curved one (only for segments)

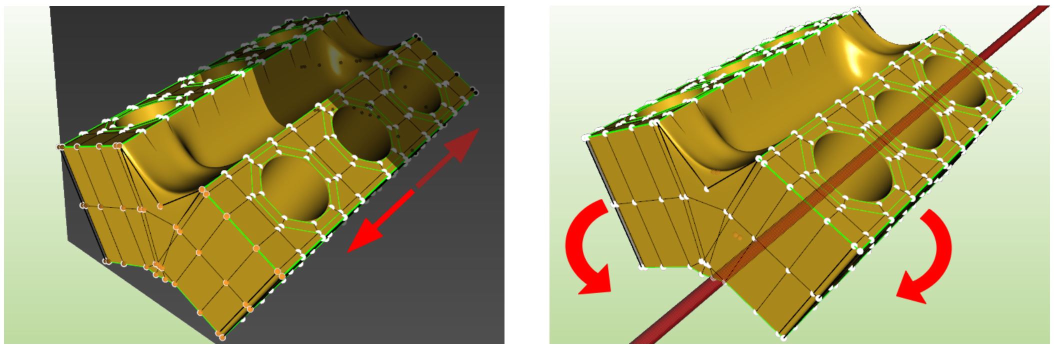
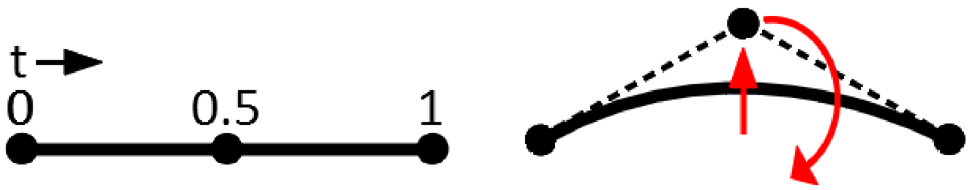

(a) Translating plane parameter (b) Rotational axis parameter

Figure 3.22.: An exemplary model of a V6 engine block with two different parameters applied to it. (a) shows the translating transformation using a plane parameter (shown in dark gray), modifying the distance between the three cylinders in each bank (the so-called cylinder spacing) while maintaining the shape of every individual cylinder. (b) shows a rotational transformation defined by an axis parameter visualized in red, modifying the angle between the two cylinder banks. The transformations are stylized with red arrows.

Plane operations: Reference planes describe infinite planes, such as the xy, or yz plane, or any other plane in Euclidean space. For transforming a group of control points relative to a reference plane, the shortest vector from each control point to the plane is calculated. Each control point is then transformed by either moving it by a fixed distance along this vector (translation) or by multiplying the vector with a certain value (scaling). A translation relative to a reference plane can be seen in Figure 3.22(a).

Axis-based operations: Reference axes describe infinite axes, such as the x, y, or z axis, or any other freely defined axis in Euclidean space. Scaling and translation work analogously to using a reference plane. In addition, axes provide a rotational degree of freedom, which allows to rotate every control point around the reference axis by a certain angle. Figure 3.22(b) shows a rotational operation around a reference axis.

Point operations: A reference point can be any 3D point. Translation and scaling are defined in the same way as described above for axes and planes. Rotational transformations are not supported.



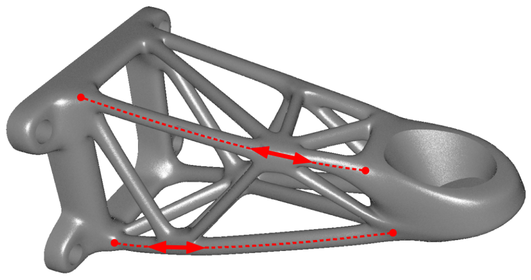
(a) Straight segment (b) Curved segment

Figure 3.23.: Concept of curved segments for performing parametric operations on CC-solid control points. (a) shows the straight segment with its three control points and its local parametrization $t \in [0, 1]$. (b) shows the curved segment with the middle control point being displaced. The displacement is defined by a radius and an angle. The operations available for curving the segments are shown as red arrows.

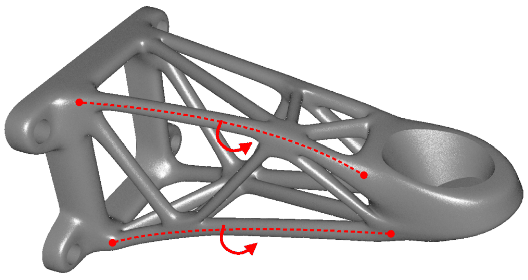
Operations on segments: While reference axes have infinite length, segments have clearly defined start and end points. In addition, segments have a third point in the middle, altogether forming a quadratic Bézier curve as shown in Figure 3.23. As positions on the curve are defined by its internal parametrization $t \in [0, 1]$, control points can be moved along the segment by mapping their x, y, z position to a value t on the segment. The sliding transformation moves all grouped control points by modifying their individual t values. Finally, each control point is mapped back to its new x, y, z position using the inverse mapping as before. Figure 3.24(a) visualizes the process of sliding two parametrizable groups along such segments.

When *curving* a segment, its middle point is transformed by an angle and a radius as shown in Figure 3.23(b). The control points assigned to that curving operation are mapped from their individual x, y, z positions to the coordinate system of the straight segment and then mapped back to x, y, z using the inverse mapping, but in the coordinate system of the curved segment. Figure 3.24(b) shows the results of curving two segments with a given angle and radius.

In addition to the application in interactive geometric modeling, this concept of parameters can be combined with the efficient tetrahedral mesh generation presented in Chapter 4, as well as with the IGA approach for CC solids presented in Chapter 5. This makes the approach well suited for efficient optimization loops using either tetrahedral FEA (as will be shown in the paper by Krämer et al. [KWA]) or IGA, as well as for interactive exploration of design spaces (see Section 6.2).



(a) Sliding transformations along two segments



(b) Curving two segments

Figure 3.24.: CC-solid model of a stylized engine mount, with parameters applied to it. (a) shows sliding transformations along two different segments. (b) shows the result of curving the two segment. The segments are visualized in red as dashed lines while the transformations are visualized as red arrows.

3.3.5. Avoiding Inversions in the Volumetric Model under Interactive Modeling

When designing just the shape of a 3D object, its interior is not important. For CC solids, the outer control points define the shape and therefore just those should be modified in an interactive modeling scenario. The outer surface of the limit volume is equivalent to a Catmull-Clark limit surface. Defining the shape of a 3D object by modeling a Catmull-Clark limit surface is a well-known paradigm used in 3D modeling software such as Blender [Fou20] and Maya [Aut18]. However, leaving inner control points unchanged while interactively modeling the surface can result in self-intersections and inverted cells. The same holds when applying parameters to surface control points as shown in Section 3.3.4. In this section, I present an approach for the inner control points to automatically adapt to modifications of the outer surface, increasing the range of modifications that can be performed without introducing self-intersections or inversions.

CC-solid control meshes being inherently volumetric enables using a physically based approach for modifying the inner control points, by treating the CC-solid model as a soft-body deformable. Mass-spring systems are a well-known concept for representing deformable models in computer graphics and animation as presented e.g. by Provot et al. [Pro+95], Liu et al. [Liu+13], and Hong et al. [Hon+16]. The two central components of such a system are the mass points and the massless springs, connecting the mass points with each other. When moving one mass point, the forces induced by compression and elongation of the individual springs result in moving all connected mass points until all forces are in equilibrium.

This concept is adopted for CC-solid control meshes, defining all control points as mass points and introducing two kinds of springs: *structural springs* and *shear springs*. The structural springs are defined along the edges of the control mesh, following the structure of the CC-solid model. The so-called shear springs connect every control point to all neighboring control points that share a cell, but no edge or face. Shear springs handle shear deformations, improve the stability of the system and add a certain degree of volume preservation to the cells. Figure 3.25 shows a visualization of structural springs and shear springs in a control mesh of $2 \times 4 \times 4$ hexahedral cells, as well as the effect of the mass-spring system on the inner control points when deforming the surface of the mesh.

The springs are modeled as linearly elastic. The force f of a spring between two mass points i and j is given by:

$$f = k_s(|x_{ij}| - l_{ij})\frac{x_{ij}}{|x_{ij}|} \tag{3.37}$$

x_{ij} is the difference between the two mass point positions, k_s is the stiffness coefficient and l_{ij} is the resting length of the spring. The resting lengths are defined by the undeformed state of the control mesh.

Forces applied to the system – in this case by translating the outer control points – result in new positions of the inner control points. The new resting state of the system is computed by minimizing the stress induced by the increased or reduced lengths of the springs, using Euler's integration scheme with an imaginary time step t. As modifying the control points is time-independent, t is only used to formulate the optimization problem. The entire system is solved until it converges.

As an alternative to forces, energies can be introduced into the system, which analogously have a resting state (where the total energy is 0). In contrast to spring-based forces, energies can depend on other geometry properties than just lengths, e.g. on angles, areas, or volumes. In any way, the system will become unstable if the deformations exceed a certain maximum.

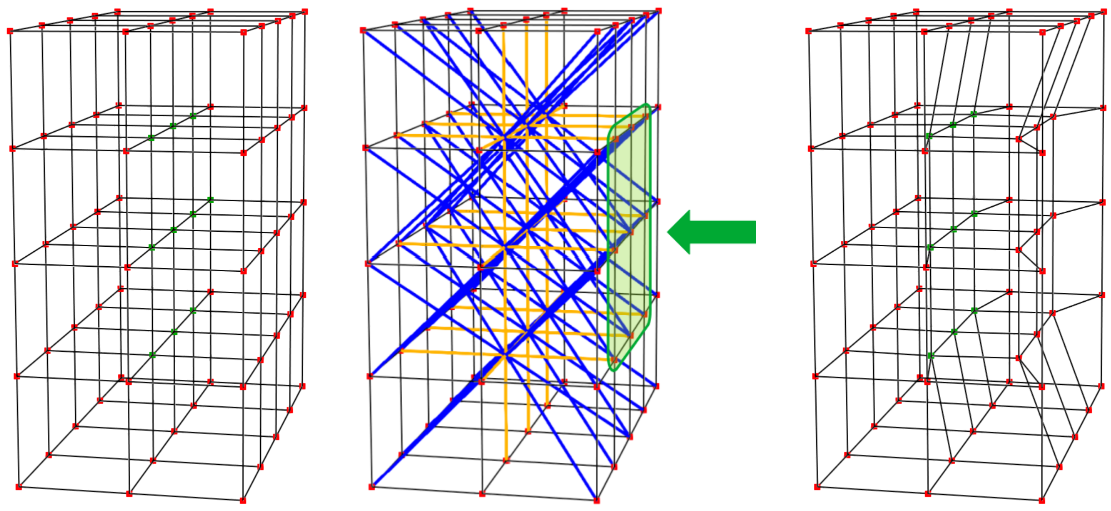

(a) Initial control mesh (b) Springs and interaction (c) Modified control mesh

Figure 3.25.: Automatic transformation of interior control points using the mass-spring approach. (a) shows the initial control mesh in a regular grid. Internal control points are shown in green, while control points on the boundary are shown in red. (b) shows the structural springs following the edges of the control mesh (orange), the shear springs going diagonally through the cells (blue), as well as an interaction deforming selected control points on the right towards the center of the grid. (c) shows the modified control mesh after moving the control points. The internal control points have been moved according to the forces induced by the mass-spring system.

3.4. Summary

For the work presented in this dissertation, Catmull-Clark solids are the volumetric representation of choice. CC solids inherently provide watertight volumetric meshes, representing a continuous volumetric limit defined by piecewise cubic basis functions.

As an adaptation to Joy and MacCracken's original subdivision rules [JM99], I presented modified rules that fulfill the tensor product property also for EEs and EVs, allowing to combine Stam's bivariate basis functions with univariate cubic B-spline basis functions in the third dimension. While Burkhart et al. already presented a modified edge rule [BHU10a], I developed a modified vertex rule to achieve this property.

Inherently with the additional dimension, evaluating the limit of CC solids is much more complex than for CC surfaces. Nevertheless, efficient limit evaluation is a key building block when processing CC-solid models for simulation, as well as for additive manufacturing. For volumetric subdivision, irregularities cannot be isolated as easily by local refinement as for subdivision surfaces. However, layered structures, which can be evaluated, are formed after conceptually performing n local subdivision steps. To that end, I derived the trivariate subdivision basis functions for irregular volumetric topologies, which enables evaluating the limit volume for CC solids in constant time without explicitly computing subdivision steps. Furthermore, I utilize crease basis functions to evaluate the limit volume for boundary cells in constant time. My direct evaluation technique can handle all topologies in a manifold volumetric mesh that subdivide into hexahedra. The approach has shown to be more efficient than the subdivision-based approach by Burkhart et al. [BHU10b] when more than two local subdivision steps are required for evaluating the target parameter point (u, v, w).

Building upon the concepts of Euler operators, I introduced new volumetric operators that perform basic topological modifications on polyhedral control meshes. Combining these operators allows for building complex volumetric modeling functionality for creating and modifying volumetric control meshes, i.e. for CC solids. The volumetric control meshes are represented in a half-face data structure that although requiring more memory than OpenVolumeMesh [KBK13], allows to more efficiently query neighborhood information frequently needed for interactive modeling. I also showed how to define and apply parameters to CC-solid control meshes. Finally, I presented a method for reducing inversions in the volumetric subdivision model when just modifying the outer surface.

Being able to represent, create, and modify volumetric control meshes for CC solids, as well as to efficiently evaluate the CC-solid limit volume, forms the foundation of the methods and techniques presented in the remaining chapters of this dissertation.

4. Efficient Tetrahedral Mesh Generation with Catmull-Clark Solids

The content of this chapter is based on the following papers:

Implicit Mesh Generation using Volumetric Subdivision
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner

Published in the *13th Workshop on Virtual Reality Interaction and Physical Simulation*
2017
The Eurographics Association

Volumetric Subdivision for consistent implicit Mesh Generation
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner

Published in *Computers & Graphics*
Volume 69, Supplement C, 2017
Elsevier

While 3D objects are usually modeled using continuous (closed) surface representations – so-called *Boundary Representations* (B-Reps) – consisting of B-spline or NURBS surfaces, physically based simulation approaches such as *Finite Element Analysis* (FEA) require discrete volumetric meshes. When meshing a surface model into a volumetric mesh for simulation, computing the positioning and topological configuration of the internal nodes is a complex task. Discontinuous and non-watertight surface models introduce additional challenges – at least affecting the performance of the mesh generation algorithm.

The approach presented here combines the well-known design properties of subdivision surfaces with a simplified and efficient mesh generation process for tetrahedral meshes. By using subdivision solids instead of surfaces for creating the initial 3D object, the volumetric information is encoded into the model already in the design phase. Utilizing the inherent volumetric nature of Catmull-Clark solids, it is possible to create consistent tetrahedral meshes very efficiently, while having the outer surface equivalent to a Catmull-Clark limit surface. Since the generation of a tetrahedral mesh from a given CC-solid control mesh can be expressed as a set of precomputable matrix-matrix and matrix-vector multiplications, the mesh generation approach presented here is much faster than those of widely used meshing tools such as CGAL [Jam+15] or TetGen [Si15]. In return, those approaches just require a watertight surface mesh instead of a CC-solid model as input.

As a consequence, combining CC solids with traditional tetrahedral meshes allows for faster iterations of design and simulation. While the tetrahedral mesh has to be re-generated when modifying the 3D object using only a surface mesh, the volumetric structure of CC solids allows for efficiently updating the tetrahedral mesh at a fraction of the computational costs. For changing topology, even re-generating the tetrahedral mesh from the CC-solid model is significantly faster. As nodes are shared between the CC-solid design mesh and the simulation mesh, the approach keeps a certain correspondence between the two meshes. This allows for clear hints to the user on where to adapt and improve the model in an iterative design and simulation scenario.

Apart from the computational costs for generating a tetrahedral mesh, having good mesh quality is an important requirement when performing Finite Element Analysis. Surface-based meshing algorithms such as the ones provided by CGAL [Jam+15] or TetGen [Si15], or by commercial software such as Siemens NX [Inc20b], are optimized to create high-quality meshes. Since these algorithms are only constraint by the outer surface of the 3D object, they have a lot of freedom for positioning the tetrahedral nodes in an optimal way. For the approach presented here, the quality of the tetrahedral mesh is directly linked to the configuration of the CC-solid control mesh and has to be investigated closely.

In Section 4.1, I present my efficient tetrahedral mesh generation approach based on Catmull-Clark solids. Section 4.2 contains the performance evaluation for the approach, including a comparison with the tetrahedral meshing algorithm provided by CGAL [Jam+15]. Section 4.3 evaluates the quality of the tetrahedral meshes generated with my approach compared to meshes created with CGAL. Several ways to improve the mesh quality are presented in Section 4.3.1 and 4.3.2. The evaluation results considering geometric quality measures as well as simulation results are presented in Section 4.3.3.

This chapter contributes to answering Research Question 3:

RQ3: How can the volumetric nature of CC solids be exploited when generating tetrahedral meshes?

4.1. Tetrahedral Mesh Generation

This section presents my efficient approach for generating a tetrahedral mesh from a CC-solid control mesh. After comparing different ways for partitioning a hexahedron into a set of tetrahedra, I present three methods for generating a tetrahedral mesh from the CC-solid control mesh with a given partitioning:

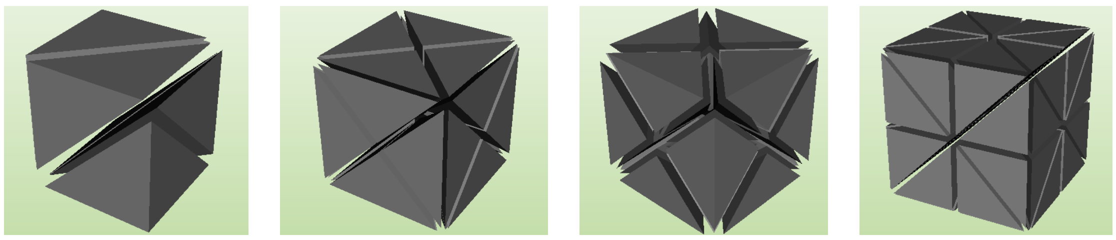
A) Performing n explicit subdivision steps to control the resolution of the tetrahedral mesh, then interpolating the subdivided mesh linearly to compute the positions of the tetrahedral nodes
B) Explicitly subdividing to control the mesh resolution and applying the CC-solid subdivision rules for calculating the positions of the tetrahedral nodes
C) Defining the tetrahedral nodes in the (u, v, w) parameter space of each CC-solid cell and calculating their positions by evaluating the limit of the CC-solid mesh

4.1.1. Partitioning a Hexahedral Cell into Tetrahedra

Many different partitionings have been developed throughout the years. Several partitionings were presented as part of the *GIBBON* framework [Moe18], as part of the *HTet* library [Lab17], and in Mueller's and Teschner's work [MT03]. Figure 4.1 shows four partitionings, which split one hexahedron into 6, 24, 35, and 48 tetrahedra, respectively.

The partitioning with 24 tetrahedra shown in Figure 4.1(b) corresponds to the well-known *Type-6* partitioning, e.g. used by Sorokina and Zeilfelder [SZ07], as well as by Dagnino et al. [DLR15]. It is the volumetric extension of the *Type-2* triangulation or *four-direction mesh* presented in the textbook by Lai and Schumaker [LS07]. The partitioning can be constructed by splitting the hexahedral cell with six splitting planes. Following this concept, I name the partitioning with 48 tetrahedra the *Type-9* partitioning, as it is constructed from nine splitting planes.

While a partitioning scheme that creates a low number of tetrahedra per hexahedron allows for a good control of the resolution of the tetrahedral mesh, the partitioning might not allow for proper surface approximation or can introduce orientational constraints regarding the compatibility of the interfaces between neighboring cells. Partitionings with asymmetric interfaces (see Figure 4.1(a)) will propagate this asymmetry throughout the mesh, introducing an orientational bias that will result in unwanted numerical effects in the FEM simulation.



(a) 6 Tets (b) 24 Tets – Type-6 (c) 35 Tets (d) 48 Tets – Type-9

Figure 4.1.: Different partitionings for dividing a hexahedral cell into tetrahedra. While the orientation of neighboring cells has to be compatible for the partitioning with 6 tetrahedra (a), the partitionings with 24 (b), 35 (c), and 48 tetrahedra (d) provide symmetric interfaces.

Since the mesh generation approach presented in this chapter is strongly linked to CC-solid subdivision, another aspect has to be considered when comparing partitioning schemes: The topology of the tetrahedral mesh should be compatible with the CC-solid subdivision rules. As described in Section 3.1.1, CC-solid subdivision is built upon four kinds of points:

1. *Cell points* – new points that are inserted for every cell
2. *Face points* – new points that are inserted for every face
3. *Edge points* – new points that are inserted for every edge
4. *Vertex points* – the original control points themselves

In order to approximate the subdivision geometry and in order to align with the CC-solid limit, a tetrahedral partitioning scheme should consist of nodes that represent as many as possible of the four kinds of points listed above. An ideal partitioning creates tetrahedral mesh nodes equivalent to cell points, face points, edge points, and vertex points.

Comparing the four partitionings shown in Figure 4.1, it can be seen that the first partitioning (Figure 4.1(a)) creates the smallest amount of tetrahedra, but due to its asymmetric interfaces to neighboring cells, constrains the orientation of their partitionings. Additionally, this partitioning only re-uses the nodes at the original vertices without creating edge, face or cell points. The other three partitionings (Figures 4.1(b) to (d), despite creating more tetrahedral elements, provide better compatibility with CC-solid subdivision and also create symmetric interfaces on all six sides. The scheme shown in Figure 4.1(c) creates nodes for face and vertex points. While the Type-6 partitioning (Figure 4.1(b)) includes nodes resembling cell, face and vertex points, it does not create any edge points. Only the Type-9 partitioning (Figure 4.1(d)) creates tetrahedral nodes that resemble all four kinds of points desired for the subdivision process and is therefore best suited for following the CC-solid subdivision rules when converting the CC-solid mesh to a tetrahedral mesh.

In the following, I focus on the Type-6 and the Type-9 partitioning. As described above, the partitionings with 6 and 35 tetrahedra are not suited for my tetrahedral mesh generation approach. The next subsections present three different methods for positioning the tetrahedral mesh nodes in the CC-solid model with a given partitioning.

4.1.2. Method A: Explicit Subdivision and Linear Interpolation

To generate the tetrahedral mesh, the volumetric control mesh is subdivided a certain number of times until the desired mesh resolution is reached. For an approximating but fast simulation, one or two subdivision steps might be enough, whereas for a more precise simulation, three or more steps may be needed, depending on the initial resolution of the CC-solid control mesh. The subdivision is performed by using the CC-solid subdivision rules as presented in Section 3.1.1. The n consecutive subdivision steps are performed as a single matrix-vector multiplication using the subdivision matrix $\hat{S}_n$ and the original control points p_0 as shown in Section 3.1.2.

The subdivided CC-solid mesh is transformed into a tetrahedral mesh using one of the tetrahedral partitionings presented in Section 4.1.1. Every partitioning implies a set of *mesh generation rules* in analogy to the CC-solid subdivision rules. For the Type-6 partitioning (Figure 4.1(b)), these rules write:

1. For every cell, create a new *cell point* at the center of the cell
2. For every face, create a new *face point* at the center of the face
3. Connect every face point to the original vertices of the corresponding face
4. Connect every face point and every original vertex to the cell point

For the Type-9 partitioning (Figure 4.1(d)), the corresponding rules are:

1. For every cell, create a new *cell point* at the center of the cell
2. For every face, create a new *face point* at the center of the face
3. For every edge, create a new *edge point* at the center of the edge
4. Connect every face point to the original vertices of the corresponding face
5. Connect every edge point to the face points of the faces that share the corresponding edge
6. Connect every face point, every edge point, and every original vertex to the cell point

The conversion process to create the vertices t_n of the tetrahedral mesh can also be represented as a matrix-vector multiplication of the so-called *tet generation matrix* T_n and the subdivided control points p_n. Similarly to the subdivision matrix, the entries of T_n are derived from the mesh generation rules of the chosen partitioning. The vertices t_n can also be expressed depending on the original control points p_0 and the subdivision matrix $\hat{S}_n$:

$$\begin{aligned} t_n &= T_n * p_n \\ &= T_n * \hat{S}_n * p_0 \end{aligned} \tag{4.1}$$

The number of vertices of the tetrahedral mesh $|t_n|$ and therefore the size of T_n depend on the chosen partitioning scheme. For the Type-6 partitioning (Figure 4.1(b)), $|t_n|$ equals to the number of vertices $|p_n|$ plus the number of faces $|f_n|$ and cells $|c_n|$ of the subdivided mesh. For the Type-9 partitioning (Figure 4.1(d)), the number of edges $|e_n|$ in the subdivided mesh is also added to $|t_n|$. As T_n converts the subdivided points p_n into the tetrahedral mesh points t_n, it is of size $|t_n| \times |p_n|$. Figure 4.2 shows the structure of T_n for dividing a single hexahedral cell using the Type-6 partitioning. When using the Type-9 partitioning scheme, T_n has the same structure as the subdivision matrix S_1 shown in Figure 3.6.

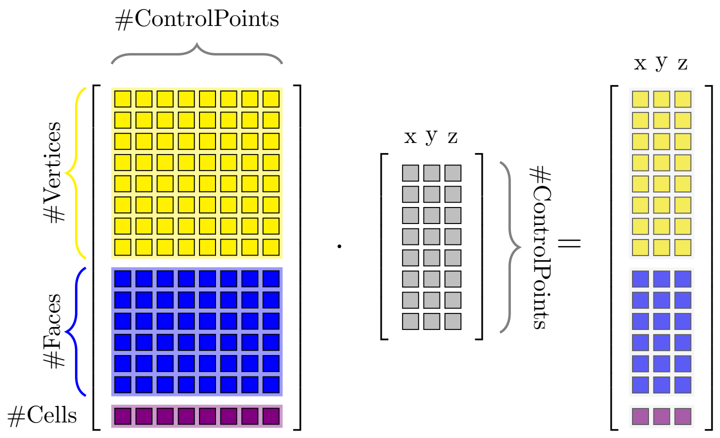
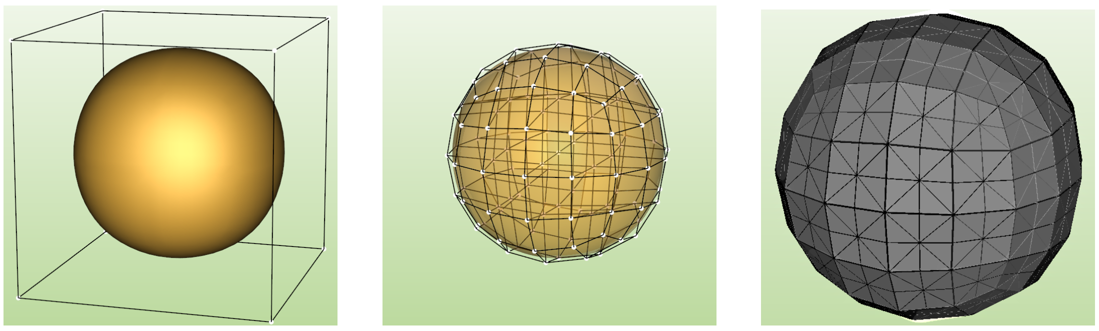


Figure 4.2.: Structure of the tetrahedral mesh generation matrix T_0 (left) multiplied with the control points p_0 (center) to calculate the tetrahedral mesh vertices t_0 (right) using the Type-6 partitioning shown in Figure 4.1(b). Derived from the mesh generation rules for the Type-6 partitioning, T_0 consists of one row per vertex, face, and cell of the CC-solid control mesh.



(a) Original CC-solid control mesh and its limit surface

(b) Control mesh and limit after two subdivision steps

(c) Close-up on the tetrahedral mesh generated from (b)

Figure 4.3.: Exemplary mesh generation process. A CC-solid control mesh consisting of one hexahedron (a) is subdivided twice (b) and then converted into a tetrahedral mesh (c) using the Type-9 partitioning.

The process of converting a CC-solid control mesh into tetrahedra as described in this section is visualized in Figure 4.3. Linearly interpolating the positions of the tetrahedral mesh nodes results in straight-edged blocks of tetrahedral elements that perfectly follow the geometry of the subdivided CC-solid control mesh but do not represent the actual limit surface. This method is recommended for the Type-6 partitioning, as it does not include all four kinds of points from the subdivision process.

4.1.3. Method B: Explicit Subdivision and Interpolation with Subdivision Rules

The actual geometry of a CC-solid model is represented by the evaluated limit and not by the subdivided control mesh. Therefore, it is desirable for the simulation mesh to match this limit as good as possible. Instead of inserting the tetrahedral mesh nodes at the center (i.e. the weighted average of all participating vertices) of each edge, face and cell using linear interpolation as described above, the CC-solid subdivision rules are applied to calculate the positions of the nodes. The connections between the nodes are established as described in Section 4.1.2. As for method A, a *tet generation matrix* $T_{n,s}$ can be used to efficiently perform the calculations in a matrix-vector multiplication. By calculating the positions of the tetrahedral mesh nodes with the CC-solid subdivision rules, the geometry of the tetrahedral mesh is closer to the limit of the CC-solid model and therefore closer to the actual geometry of the 3D object to be simulated.

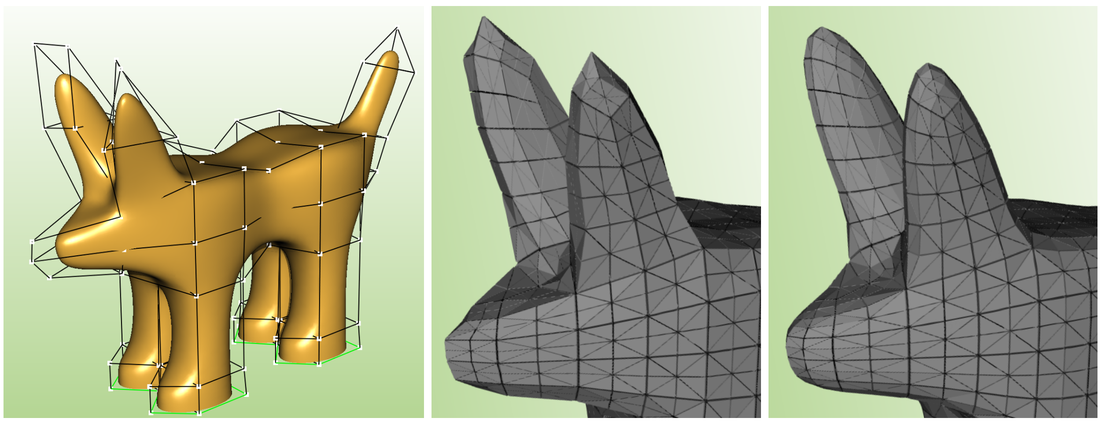

(a) CC-solid control mesh and limit surface (b) Linear interpolation (c) Applying the subdivision rules

Figure 4.4.: Tetrahedral mesh generation from the coyote model. After one subdivision step, the CC-solid model (a) is converted into 48 tetrahedra per cell, calculating the positions of the tetrahedral mesh node with linear interpolation (b) and by applying the subdivision rules (c).

As described in Section 4.1.1, a good partitioning creates tetrahedral nodes for cell, face, and edge points, as well as re-uses all original CC-solid control points. The method presented here has to be combined with partitionings that provide all four kinds of points, i.e. the Type-9 partitioning. Using this method with the Type-6 partitioning creates artifacts as the partitioning does not provide edge points. For those partitionings, the linear interpolation must be used.

Figure 4.4 shows a visual comparison of methods A and B for the *coyote* model. The improved surface approximation of method B can be seen especially around the ears, nose, and tail of the model.

4.1.4. Method C: Direct Limit Evaluation

Instead of explicitly subdividing the CC-solid control mesh and calculating the tetrahedral mesh nodes from the subdivided mesh as in methods A and B, the nodes can also be calculated from the limit volume of the CC-solid model. Therefore, the tetrahedral points required by the chosen partitioning are expressed in (u, v, w) parameter space.

A point in the center of a cell corresponds to (u, v, w) coordinates of $(0.5, 0.5, 0.5)$, face points correspond to $(0, 0.5, 0.5)$, $(1, 0.5, 0.5)$, $(0.5, 0, 0.5)$, and so on. The limit volume is evaluated for every parameter point (u, v, w) in each cell to calculate the (x, y, z) coordinates of the corresponding tetrahedral node. By using the limit instead of a discrete number of subdivision steps, all tetrahedral mesh nodes are guaranteed to lie on/in the actual geometry. As this might not be crucial for inner nodes, it improves the surface approximation of the tetrahedral mesh compared to methods A and B, and thereby reduces the discretization error.

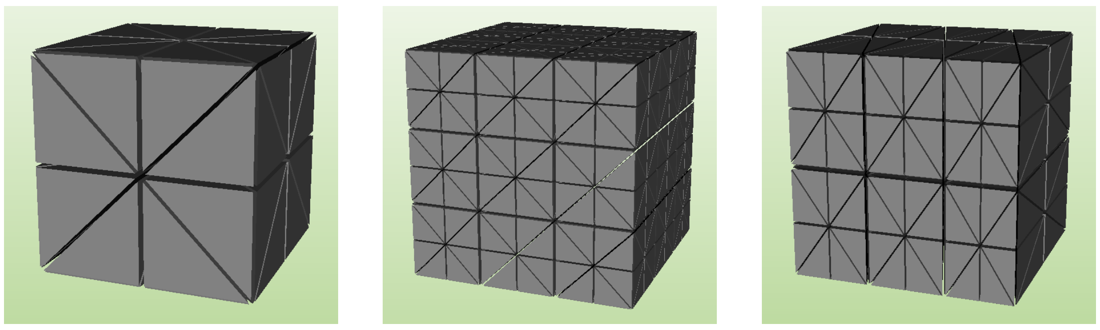
As presented in Section 3.2 and especially shown in Equations (3.10) and (3.16), a limit point in a given cell is defined as the weighted sum over all control points in the one-ring neighborhood of that cell – the weights being defined by the basis functions. Therefore, the limit evaluations of all nodes of the tetrahedral mesh can be expressed as a matrix-vector multiplication similar to the one presented in Section 4.1.2. Introducing the *limit tet matrix* T_l, the tetrahedral mesh nodes t_l can be calculated as follows:

$$t_l = T_l * p_0 \tag{4.2}$$

p_0 denotes the control points of the initial CC-solid control mesh. T_l consists of one row per tetrahedral node and one column per control point, having the individual weights of the basis functions in the corresponding columns. T_l is of size $|t_l| \times |p_0|$. If one or two initial subdivisions are required for the model to be evaluatable (see Section 3.2), they can be integrated into T_l by multiplying the corresponding subdivision matrices S_1 or $\hat{S}_2$, respectively. This results in a new limit tet matrix $\hat{T}_l$ with:

$$t_l = T_l * S_1 * p_0 = \hat{T}_l * p_0 \tag{4.3}$$

The resolution of the tetrahedral mesh is controlled by creating one or multiple instances of the chosen partitioning in each CC-solid cell. Figure 4.5 shows different numbers of instances of the Type-9 partitioning in a single CC-solid cell. The (u, v, w) coordinates are transformed accordingly for the nodes of each instance, e.g. from $[0, 1]^3$ to $[0, 0.5]^3$. Isotropic transformations create e.g. $2 \times 2 \times 2$, or $3 \times 3 \times 3$ instances (see Figure 4.5(b)), while anisotropic transformations allow for creating e.g. $1 \times 3 \times 3$ or $3 \times 2 \times 1$ instances in one cell (Figure 4.5(c)). Thereby, the aspect ratios of distorted cells can be improved as long the interfaces of tetrahedral nodes between neighboring cells are compatible.


(a) One instance of the Type-9 partitioning

(b) $3 \times 3 \times 3$ instances

(c) Anisotropic configuration of $3 \times 2 \times 1$ instances

Figure 4.5.: Different numbers of instances of the Type-9 partitioning in one CC-solid cell. While (a) and (b) show isotropic configurations, (c) shows an anisotropic one.

4.2. Performance Evaluation

This section shows performance measurements of my tetrahedral mesh generation approach in comparison to the approach provided by the state-of-the-art meshing library CGAL [Jam+15] for several CC-solid models. While my approach exploits the volumetric structure of the CC-solid model, CGAL just requires the tessellated surface as input. The measurements presented here were taken from single-threaded CPU implementations written in C++. The CGAL library was used in version 4.8. All programs were executed and measured on a Windows 10 desktop PC with an Intel Core i5-4570 CPU with a clock rate of 3.2 GHz and 16 Gigabytes of DDR3 1600 system memory.

In order to analyze the impact of the subdivision level n on the computation times of my mesh generation approach, a CC-solid model consisting of just one hexahedral cell has been subdivided up to five times. Table 4.1 shows the computation time for generating the matrices $\hat{S}_n$, $T_{n,6}$ $T_{n,9}$, $T_{n,s,9}$, and $\hat{T}_{l,9}$. The matrix indices indicate the interpolation method (linear, applying the subdivision rules, and evaluating the limit) as well as the partitioning (Type-6 and Type-9). Keep in mind that the matrices have to be re-generated only if the topology of the control mesh changes. The increase of computation time in relation to the subdivision level n is exponential. This is expected, since the number of resulting cells grows exponentially with the subdivision level by a factor of 8^n, since each hexahedral cell is subdivided into eight cells for every subdivision step. In relation to the number of tetrahedra, the computation time increases linearly.

Subdiv. Level n	Control Cells	Tetrahedra Type-6	Tetrahedra Type-9	Generation Time [ms] $\hat{S}_n$	$T_{n,6}$	$T_{n,9}$	$T_{n,s,9}$	$\hat{T}_{l,9}$
0	1	24	48	-	0.01	0.01	0.03	3.40
1	8	192	384	0.03	0.03	0.03	0.21	3.77
2	64	1536	3072	0.30	0.13	0.17	2.37	21.02
3	512	12288	24576	2.63	1.87	1.70	24.97	138.6
4	4096	98304	196608	27.53	17.67	23.21	260.5	982.4
5	32768	786432	1572864	298.7	149.3	192.1	2455	8195

Table 4.1.: Timings for generating the combined subdivision matrix $\hat{S}_n$ and the tet generation matrices $T_{n,6}$ $T_{n,9}$, $T_{n,s,9}$, and $\hat{T}_{l,9}$ for an increasing subdivision level n. As $n = 0$ corresponds to the original control mesh, subdivision is not required and only the tet generation matrices have to be computed. For $\hat{T}_{l,9}$, the subdivisions are expressed by increasing the number of instances per cell accordingly.

Table 4.2 shows the computation time for generating and applying the matrices $\hat{S}_n$ and $T_{n,s,9}$ in order to subdivide and convert the model into a tetrahedral mesh of a specific resolution. Again, matrix generation as well as matrix multiplication show a linear complexity in the number of resulting tetrahedra. This is expected, since the computations are linear matrix-vector multiplications (see Equation (4.1)). For comparison, Table 4.2 also shows the time needed to create a tetrahedral mesh with a similar number of tetrahedra using CGAL [Jam+15]. For CGAL's meshing algorithm, the parameter *cell size* c was chosen such that resulting number of tetrahedra is comparable with the CC-solid-based meshes. For the entire process of generating and multiplying the matrices, my approach is faster by a factor between $26\times$ and $196\times$. For a low number of tetrahedra, the overhead of initializing the CGAL data structures dominates the mesh generation process while the subdivision matrices and tet generation matrices are computed almost instantly.

For further evaluating the approach, the performance was measured for four example models. The test set consists of the *coyote* model shown in Figure 4.4(a), as well as of the *car*, *engine mount*, and *wrench* models shown in Figure 4.6. For the CC-solid approach, the Type-9 partitioning was used together with method B (applying the subdivision rules).

Figure 4.7 shows the results of these measurements. As before, the computation times increase linearly with the number of tetrahedra. This can be directly derived from the subdivision rules as they are also defined as operations of linear complexity for all cells, faces, edges and vertices of the control mesh. The algorithm behaves similarly for all four models. The topology of the control mesh, i.e. the ratio of regular vs. irregular cells

	My Approach				CGAL Meshing			
n	Tetra-hedra	Time [ms] Gen.	Mult.	Total	c	Tetra-hedra	Time [ms]	Factor
0	48	0.03	0	0.03	0.4	46	5.9	196.7
1	384	0.2	0	0.2	0.21	394	28.8	144.0
2	3072	2.37	0.02	2.39	0.11	3088	157.7	66.06
3	24576	25.03	0.22	25.25	0.056	24573	1092	43.25
4	196608	260.5	1.74	262.2	0.028	196437	7444	28.39
5	1572864	2456	20.92	2476	0.014	1571337	65689	26.53

Table 4.2.: Timings of my mesh generation approach compared to the tetrahedral meshing provided by the CGAL library [Jam+15]. The mesh generation based on CC solids consists of generating the matrices $\hat{S}_n$ and $T_{n,s,9}$, as well as of performing the multiplication $t_n = T_{n,s,9} * \hat{S}_n * p_0$. Cell sizes c for CGAL were chosen explicitly for creating approximately the same number of tetrahedra as with my approach for a given subdivision level n. As for Table 4.1, subdivision is not needed for $n = 0$.

and inner cells vs. boundary cells, has only a marginal effect on the computational cost of the approach. Tetrahedral meshing in CGAL also shows linear complexity for fixed parameters. Again, meshing performs similarly for all four example models. However, the time required for generating the tetrahedral meshes is lower by $26\times$ (car model) to $30\times$ (wrench) for the approach based on CC solids.

Exploiting the inherent volumetric characteristic of Catmull-Clark solids for tetrahedral mesh generation brings significant performance benefits compared to traditional meshing techniques, e.g. using CGAL [Jam+15]. Converting a CC-solid model into a tetrahedral mesh as described in this chapter provides an efficient mesh generation process that also allows for high feedback rates in an interactive modeling and simulation scenario. The approach creates a link between the volumetric representation of CC solids and the mature and well-established Finite Element Analysis based on tetrahedra.

When the geometry of the CC-solid model is modified (i.e. positions of the control points p_0 change) between simulation runs, the matrices $\hat{S}_n$ and T_n, or $\hat{T}_l$, respectively, remain unchanged and can be re-used. This allows to precompute these matrices and the product $T_n * \hat{S}_n$ for a given subdivision level n, resulting in even higher speedup of up to $3100\times$. However, changing the topology of the model by adding or removing vertices, edges, faces or cells or changing the subdivision level entails a re-computation of the matrices.

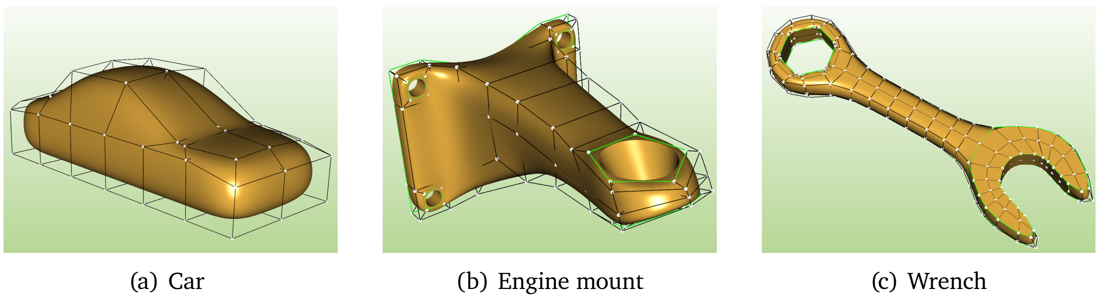


Figure 4.6.: Three CC-solid models used for the evaluation, additionally to the coyote model shown in Figure 4.4(a).

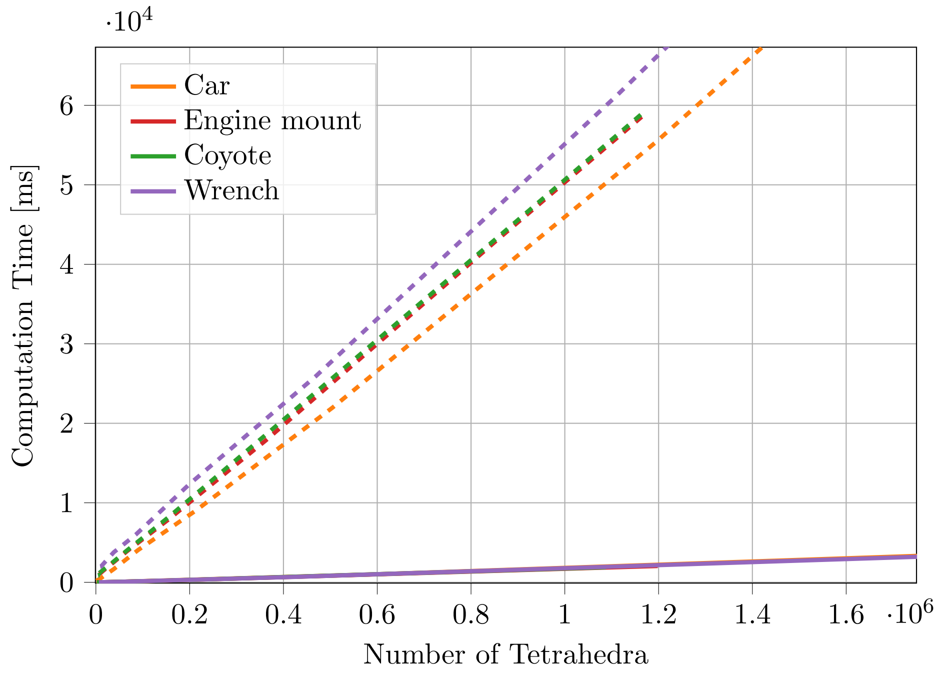


Figure 4.7.: Performance comparison of the tetrahedral mesh generation based on CC solids (solid lines) and the meshing performed with CGAL [Jam+15] (dashed lines). The four models used for the evaluation are shown in Figures 4.4(a) and 4.6. For each approach, the models behave similarly, because the computation time scales linearly with the number of tetrahedra for both approaches – almost independently from the shape of the model.

4.3. Consistency and Mesh Quality

Mesh quality is an important factor when it comes to simulation. As a general rule, mesh quality can be derived from the uniformity of the mesh elements as proposed by Shewchuk [She02] and Field [Fie00]. This means that all mesh elements – all tetrahedra – should have approximately the same size and a preferably uniform aspect ratio. State-of-the-art meshing tools such as CGAL [Jam+15], TetGen [Si15], or Gmsh [GR09] attempt to create optimized tetrahedral meshes in order to meet these quality requirements as good as possible.

In contrast, for the three methods presented in Sections 4.1.2 to 4.1.4, the quality of the tetrahedral mesh is directly linked to the topological and geometrical configuration of the CC-solid control mesh that is used to design the 3D object. Larger cells on the control mesh will result in larger simulation elements and smaller cells will result in smaller ones. The same applies to the aspect ratio of the cells. Figure 4.8 shows two tetrahedral meshes generated from the engine mount model for comparison. Cohen et al. presented several approaches for creating good-quality volumetric models already in the design phase [Coh+10]. However, the corresponding control meshes might not be convenient to work with in an interactive modeling environment.

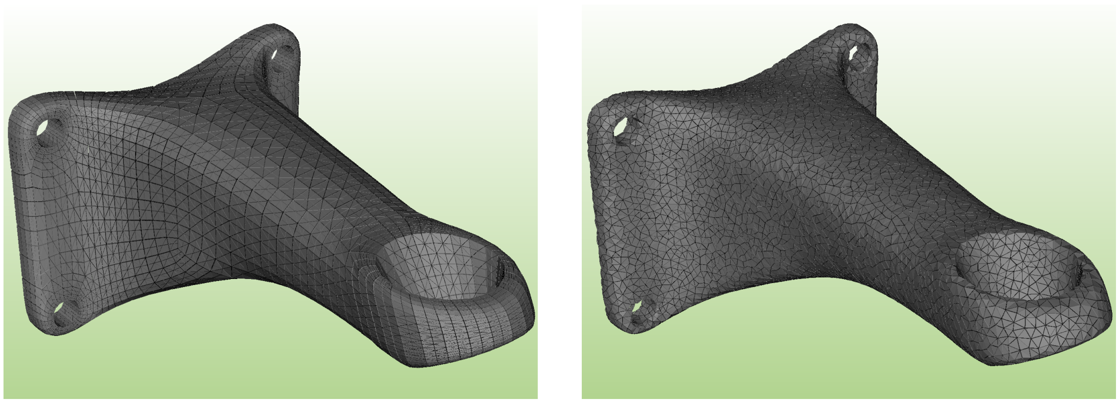
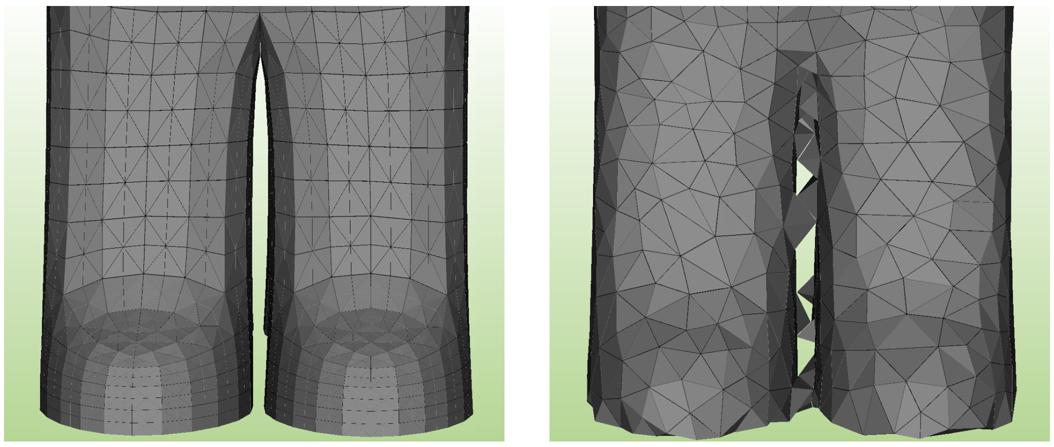

(a) Tetrahedral mesh generated from CC solids (b) Tetrahedral mesh generated with CGAL

Figure 4.8.: Comparison of tetrahedral meshes generated with my approach (a), as well as using CGAL (b). The structure of the CC-solid control mesh (Figure 4.6(b)) can be recognized in the tetrahedral mesh in (a).

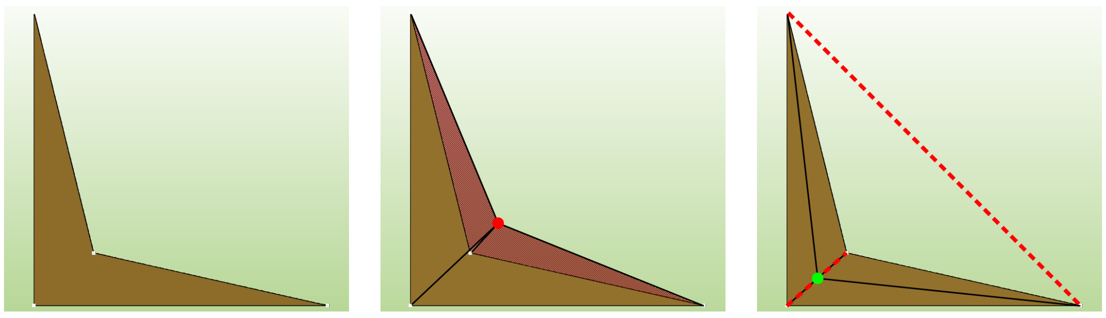


(a) Tetrahedral mesh generated from CC solids (b) Tetrahedral mesh generated with CGAL

Figure 4.9.: Legs of the coyote model converted into a tetrahedral mesh using my approach (a), as well as using CGAL (b). While the mesh generated from CC solids properly separates both legs, they are partly connected in the mesh created by CGAL. Both meshes consist of about 30.000 tetrahedra.

Considering the topology of the tetrahedral mesh, having this link between the CC-solid control mesh and the simulation mesh is beneficial. While CGAL or other meshing tools might connect separated areas of the model if gaps are below a certain threshold, the mesh generation approach based on Catmull-Clark solids preserves the topology of the original model. Figure 4.9 shows an example where multiple tetrahedra span from one leg of the coyote model to the neighboring one, although the legs are separated in the initial geometry. In most cases, such errors can be avoided by manually fine-tuning the parameters of the corresponding mesh generation algorithm – a time-consuming process.

Apart from erroneous topologies, having ill-shaped elements in the mesh still leads to unstable simulations. Large deviations in angles or aspect ratios of the elements will reduce the overall quality of the simulation result as well as the convergence of the chosen FEM solver. To some extent, the aspect ratios of CC-solid cells can be improved by applying loop-cuts as presented in Section 3.3.3 and by choosing anisotropic configurations for the method described in Section 4.1.4. However, self-intersections and inverted elements are far more critical than badly shaped elements as those might result in a diverging solve and thus rendering the simulation results unusable. In the following, I present two methods for improving the quality of the tetrahedral mesh during the mesh generation process.



(a) A concave face, as it might appear in a Catmull-Clark mesh.

(b) The face point is outside of the original face shown in (a).

(c) The new vertex is inserted at the midpoint of one of the face's diagonals.

Figure 4.10.: Inserting a tetrahedral node into a concave face while converting the CC-solid model into a tetrahedral mesh. (a) shows the original CC-solid face. Inserting the tetrahedral node at the average of all face vertices leads to inverted triangles (shown in red), as the node (red dot) is inserted outside the face itself (b). Using the midpoint of one of the face's diagonals instead, divides the face into four correctly oriented triangles (c). The diagonals are shown as dashed red lines, while the newly created tetrahedral node is shown in green. An analog approach is used when inserting a new vertex into a concave CC-solid cell.

4.3.1. Inserting Tetrahedral Nodes into Concave CC-Solid Cells

As described above, for the mesh generation approach presented in this chapter, the quality of the tetrahedral mesh depends on the quality of the underlying CC-solid control mesh. However, even without self-intersections in the subdivision mesh, inconsistent simulation meshes can be created during the mesh generation process.

Inserting new vertices at the center of each face and the center of each cell becomes problematic for distorted hexahedral cells and produces inverted tetrahedral elements in some cases. This happens especially for concave geometries whenever the geometric center of the face or cell is not part of the face or cell itself. Figure 4.10 shows a 2D example of such a case. To avoid inversions, the methods described in Sections 4.1.2 and 4.1.3 are enhanced by an inside-outside test for the newly generated vertices.

If the geometric center of a face is outside of the face itself as shown in Figure 4.10(b), the midpoint of one of the face's diagonals is taken as new vertex position. The first diagonal with its midpoint inside the face is then used as shown in Figure 4.10(c), and the factors in the *tet generation matrix* T_n or $T_{n,s}$ are modified accordingly.

A similar approach is used when generating the new vertex at the center of each cell. If the center of the cell is outside of the cell itself, the cell's diagonals are computed as the lines between every two cell vertices that don't share a common face. The first diagonal with its midpoint inside the cell is used for the new vertex position. Thus, the creation of inverted tetrahedra from concave cells is avoided.

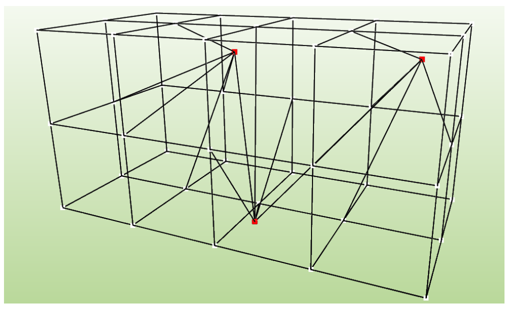
4.3.2. Laplacian Smoothing of Inner Points

To resolve small inversion of cells and to increase mesh uniformity, an iterative version of Laplacian smoothing is applied to the control points of the CC-solid model. For maintaining the limit surface of the designed 3D object, Laplacian smoothing is only applied to internal control points of the model. Please note, that this approach is only applicable to CC-solid models with constant material parameters. Varying material properties encoded at the control points as described in Section 3.1 are not preserved.

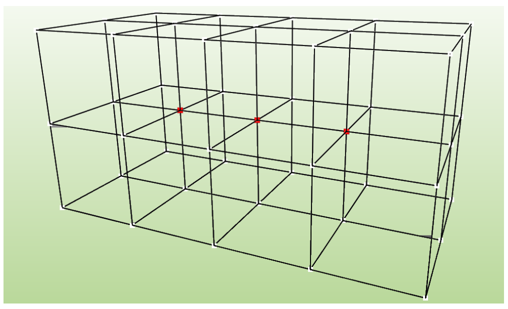
Within each iteration k, every inner control point c_i of the mesh is moved to the average position of all connected control points c_j as follows, with N being the number of neighboring control points of c_i:

$$c_{i,k+1} = \frac{1}{N}\sum_{j=1}^{N} c_{j,k} \tag{4.4}$$

Figure 4.11 shows a simple example of Laplacian smoothing on inner control points. However, perfect symmetry in the control mesh can worsen the convergence of the FEM solver due to numerical instabilities. My experiments have shown that in this case, applying small random displacements on inner control points helps to avoid these instabilities.



(a) A hexahedral control mesh with heavily distorted inner control points (marked in red)



(b) The same control mesh after 40 iterations of Laplacian smoothing

Figure 4.11.: A hexahedral control mesh before (a) and after (b) Laplacian smoothing of the inner control points. In (b), the points have converged into a regular constellation.

4.3.3. Quantifying Mesh Quality

In order to analyze the mesh quality of the tetrahedral meshes created from CC solids compared to the ones created with CGAL, I measured two quality indicators proposed by Shewchuk [She02] and Field [Fie00]: the volume and the circumradius of the tetrahedral elements. The results are shown in Table 4.3. In general, CGAL's meshing algorithm creates tetrahedral meshes with more uniform elements than CC-solid-based mesh generation does. This can be seen especially when comparing the min-max ratio for the volume and the circumradius. While Laplacian smoothing increases the mesh quality for the car and the coyote model, it decreases the quality for the other two models significantly. The optimal combination of tetrahedral partitioning scheme, interpolation method, and whether or not to apply Laplacian smoothing, depends on the actual CC-solid model to be converted.

Still, the CC-solid-based tetrahedral mesh generation approach creates meshes with more homogeneous connectivity compared to CGAL. Even if the minimum and maximum numbers of adjacent vertices are similar for both approaches, the median is much closer to the minimum value than it is for CGAL's meshes. This leads to a sparsification of the stiffness matrices, improving the solvability of the linear system and resulting in a speedup regarding simulation time.

Model	Method	Volume			Circumradius		
		min	max	factor	min	max	factor
Car	Type-6	2.27e-05	0.000935	41.3	0.0565	0.844	14.9
	Type-9 Sub.	7.88e-06	0.000459	58.4	0.0477	0.339	7.12
	Type-9 S. Lp	1.10e-05	0.000476	43.4	0.0494	0.275	5.56
	CGAL	3.05e-05	0.000294	9.65	0.0519	0.0940	1.81
Coyote	Type-6	0.000112	0.00846	75.5	0.0802	1.028	12.82
	Type-9 Sub.	2.93e-05	0.00530	181.2	0.0479	0.874	18.25
	Type-9 S. Lp	4.71e-05	0.00531	112.9	0.0517	0.717	13.87
	CGAL	2.79e-06	0.000145	52.0	0.0229	0.0836	3.66
Engine mount	Type-6	0.00511	2.82	552.0	0.336	27.2	81.1
	Type-9 Sub.	0.00222	1.54	692.7	0.276	33.2	120.2
	Type-9 S. Lp	0.000107	1.63	15235	0.274	21.7	79.0
	CGAL	2.71e-06	0.000850	313.6	0.0202	0.138	6.83
Wrench	Type-6	5.91e-06	0.000503	85.1	0.0413	0.591	14.3
	Type-9 Sub.	3.17e-06	0.000260	82.0	0.0318	0.412	13.0
	Type-9 S. Lp	1.64e-07	0.000282	1723.4	0.0330	2.27	68.7
	CGAL	4.19e-06	0.000423	100.8	0.0249	0.108	4.34

Table 4.3.: Evaluating the minimum and maximum values, as well as the min-max ratio of the volume and the circumradius per tetrahedron for tetrahedral meshes generated with my approach compared to meshes created by CGAL. The tetrahedral meshes were created using the Type-6 partitioning using linear interpolation, as well as the Type-9 partition using subdivision rules and Laplacian smoothing. Subdivision levels and CGAL cell sizes for the selected models were chosen such that the numbers of tetrahedral elements per mesh match closely.

For evaluating the effect of the lower mesh quality on the simulation, I performed a structural analysis on all four models, using a linear elastic material model with Eigen's conjugate gradient (CG) solver [JG08]. Figure 4.12 shows the computation time and the number of iterations required by the CG solver when simulating tetrahedral meshes created from CC solids and tetrahedral meshes created with CGAL. As can be seen, the higher quality of the meshes created with CGAL results in a lower runtime as well as in fewer iterations compared to my approach. The benefit of having a more uniform connectivity in the CC-solid mesh (see Table 4.4) is outweighed by the lower mesh quality. This applies especially to the wrench model, where every subdivision step creates more degenerate

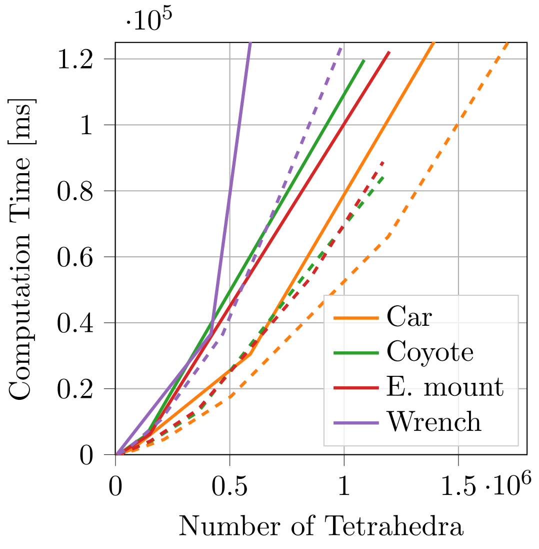
Model	Method	Adjacent Vertices min	max	median
Car	CC solids	5	26	9
	CGAL	5	21	13
Coyote	CC solids	5	26	9
	CGAL	4	23	14
Engine mount	CC solids	5	32	9
	CGAL	4	23	13
Wrench	CC solids	5	26	10
	CGAL	4	23	13

Table 4.4.: Analysis of the connectivity of every vertex to its neighboring vertices, showing the minimum, maximum and median value for each mesh. For my approach, the Type-9 partitioning was used. Again, subdivision levels and CGAL cell sizes for the selected models were chosen such that the number of tetrahedra match closely.

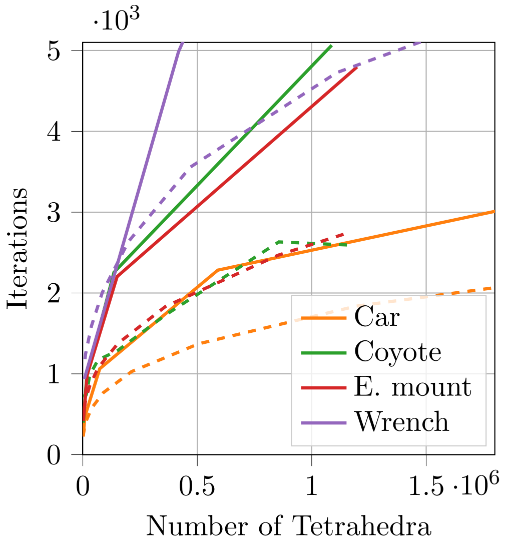
tetrahedra in the ring section of the wrench. However, as shown in Figure 4.12(c), the additional time required for solving the linear system is compensated for by the faster mesh generation of my approach compared to CGAL – except for higher-resolution meshes of the wrench.

Concerning the simulation results, the stress and displacement values for both approaches were compared to reference simulations using highly detailed tetrahedral meshes generated with a commercial meshing software. Figure 4.13 visualizes the *von Mises* stress for both approaches for the engine mount and the wrench model. As can be seen, the stress distribution is almost identical throughout the model, apart from artifacts close to fixations, resulting from the different discretizations.

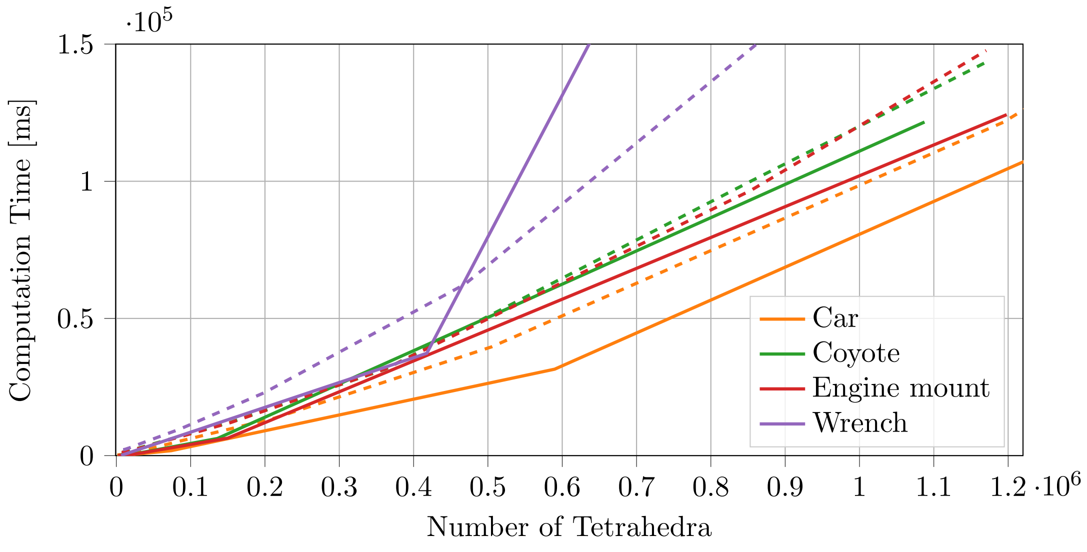
The simulation results were evaluated numerically in terms of comparing the maximum values for the *von Mises* stress and the displacement, as well as by comparing the overall distribution of stress and displacement throughout the entire model. As Figure 4.14 shows, the deviations in the *von Mises* stress behave similarly for both approaches – CGAL-based and CC-solid-based. The deviations in the displacement (Figures 4.14(b) and (d)) show bigger differences between the two approaches, especially for the car and the wrench model. However, for most cases, the relative errors for the maximum displacement are below 1%. Only meshes with a relatively low resolution produce larger errors.



(a) Runtime of Eigen's CG solver

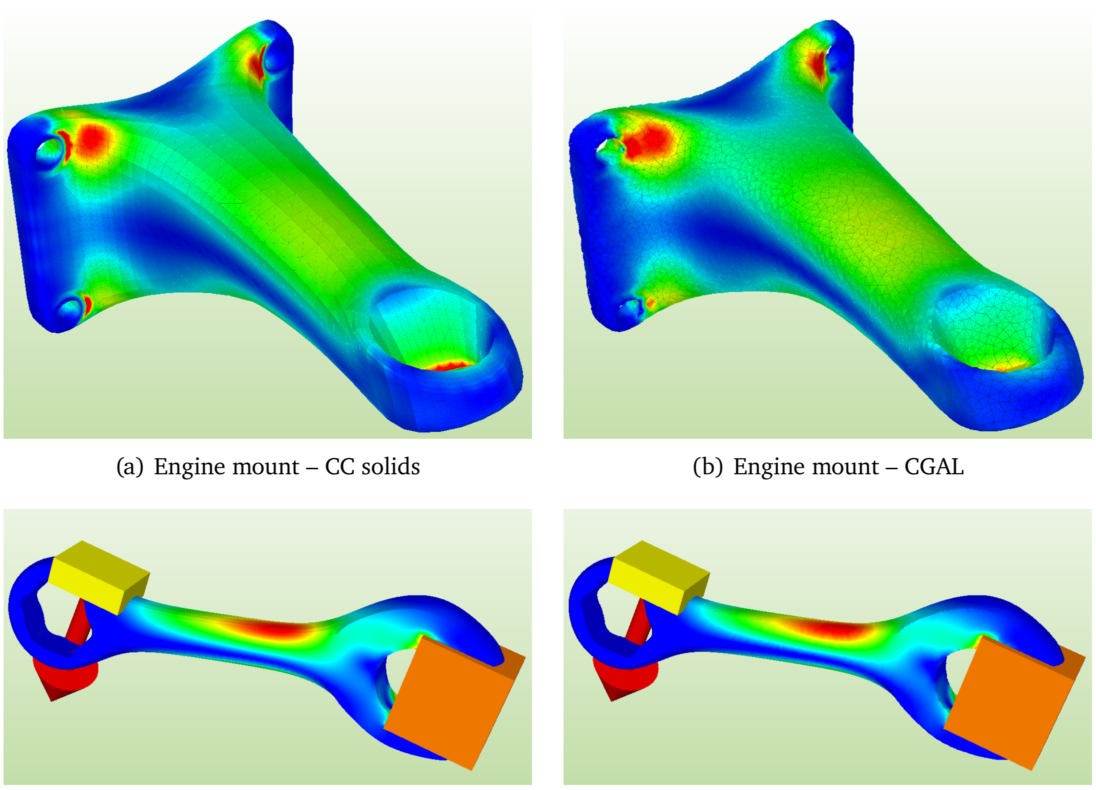


(b) Iterations required by the solver



(c) Combined runtime of meshing and solving

Figure 4.12.: Runtime (a), as well as the number of iterations (b) required by Eigen's conjugate gradient (CG) solver [JG08]. (c) shows the combined runtime for generating the tetrahedral mesh and solving the linear system. The finite element simulations were performed on tetrahedral meshes created with my approach (solid lines) as well as on meshes created by CGAL (dashed lines).



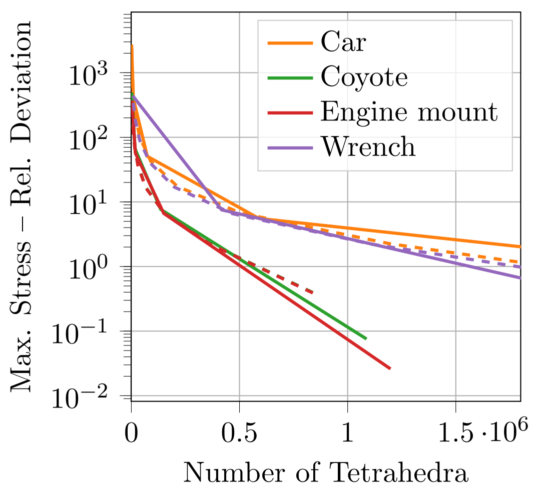
(a) Engine mount – CC solids

(b) Engine mount – CGAL

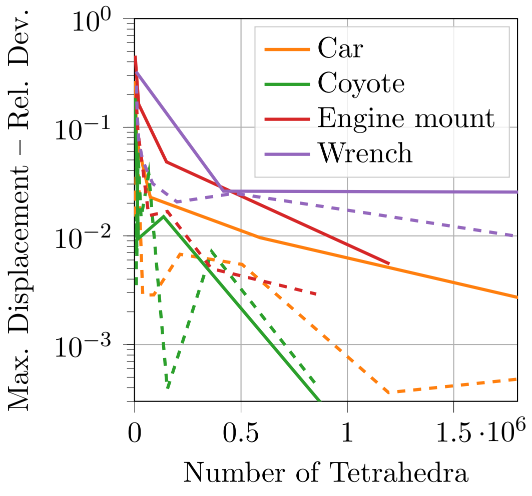
(c) Wrench – CC solids

(d) Wrench – CGAL

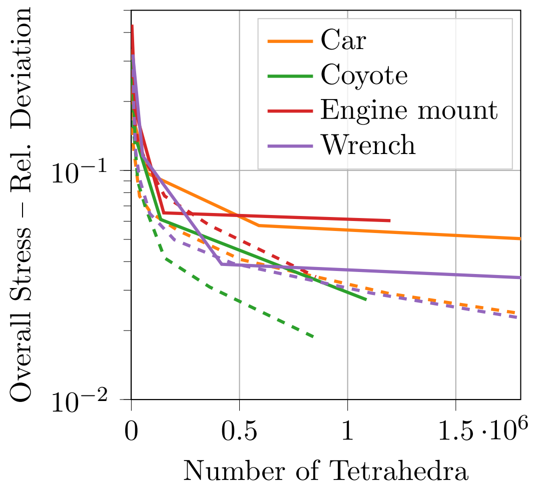
Figure 4.13.: Von Mises stress results for the engine mount (a, b) and the wrench model (c, d). The simulations were performed with Eigen's CG solver [JG08] on tetrahedral meshes created from the CC-solid models (a, c) as well as meshes created from the tessellated limit surface with CGAL [Jam+15] (b, d). For (c) and (d), the boundary conditions are also visualized, showing fixations in orange and forced areas in yellow with the direction of the force as a red arrow. The tetrahedral meshes for the engine mount consisted of approximately $150k$ tetrahedra while the meshes for the wrench model consisted of about $420k$ tetrahedra.



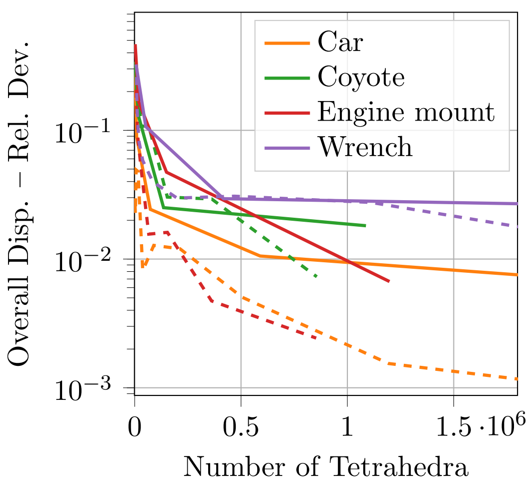
(a) Maximum von Mises stress



(b) Maximum displacement



(c) Overall von Mises stress



(d) Overall displacement

Figure 4.14.: Relative deviation from the reference solutions regarding the von Mises stress (a, c), as well as the displacement (b, d). The reference solutions were created by Eigen's CG solver [JG08] with highly detailed tetrahedral meshes generated by a commercial meshing software. (a) and (b) show the deviation in the maximum stress and displacement, while (c) and (d) show the overall relative deviation throughout the entire mesh. Again, the results obtained with my approach are shown as solid lines, while the results with meshes from CGAL are shown with dashed lines.

4.4. Summary

In this chapter, I investigated how volumetric subdivision schemes, i.e. Catmull-Clark solids can contribute to strengthening the integration of geometric modeling and simulation, while relying on the well-matured methods of tetrahedra-based FEA.

By exploiting the inherent volumetric structure of Catmull-Clark subdivision solids, I presented an efficient mesh generation approach that derives a tetrahedral mesh from a given CC-solid control mesh. The tetrahedral mesh can be used with many standard and state-of-the-art finite element solvers. The approach decreases the complexity of the meshing process and thereby also decreases the iteration times for simulation-based design and optimization loops by combining CC solids with traditional FEA.

The simulation mesh is obtained by applying the volumetric subdivision scheme to the initial CC-solid control mesh multiple times and then converting the result into a tetrahedral mesh or by directly evaluating the limit volume to calculate the positions of the tetrahedral mesh nodes. Since the CC-solid model and the tetrahedral mesh share some of their vertices, my approach creates a partial correspondence between the design mesh and the simulation mesh that allows the simulation results to be incorporated directly into the design.

Compared to creating a tetrahedral mesh from the CC limit surface using CGAL [Jam+15], my mesh generation approach achieves a speedup of up to $30\times$ for comparably large meshes. For a tetrahedral mesh of 3.3 million tetrahedra, the meshing time is reduced from 187 to 6.2 seconds. When only modifying the geometry of the design mesh, the *mesh generation matrix* can be precomputed, resulting in a speedup of up to $3100\times$.

In contrast to tetrahedral meshes generated from a surface model, the quality of the CC-solid-based simulation meshes depends directly on the configuration of the subdivision control mesh. Huge deviations in cell sizes as well as ill-shaped and distorted cells lead to simulation meshes of bad quality. Cohen et al. present several design guidelines for creating good simulation models in the context of IGA [Coh+10]. However, following these guidelines creates models that are inconvenient to work with using interactive modeling. In this chapter, I presented two methods for improving mesh quality during the mesh generation process:

a) To avoid local inversions for concave CC-solid elements, an extended set of rules is used when creating face points and cell points in such elements.
b) To improve mesh uniformity and to avoid self-intersections in the control mesh, Laplacian smoothing has been examined for inner control points.

Although these procedures improve the quality of the tetrahedral meshes, they only partly overcome the described drawback. State-of-the-art meshing algorithms such as CGAL still produces higher-quality meshes.

The lower mesh quality results in a higher number of iterations of the finite element solver and therefore a higher computational cost for the simulation itself. However, since the simulation results are comparable between both approaches, the increased runtime of the simulation is compensated by the speedup in the meshing process. Since the total time of meshing and simulating – as it appears in an iterative modeling and simulation scenario – is dominated by the time required for solving the linear system, using a faster solver would further increase the performance benefit of my approach. In Section 6.1, I present an integrated modeling and simulation framework that combines the efficient mesh generation approach with the fast GPU solver by Weber et al. [Web+13].

The next chapter shows how to perform physically based simulation directly on the CC-solid model, without an underlying tetrahedral mesh. Exploring the idea of IGA in the context of CC solids, I present new methods that improve the state of the art beyond the work of Burkhart et al. [BHU10b].

5. Isogeometric Analysis on Catmull-Clark Solids

The content of this chapter is based on the following paper:

Analyzing Mesh vs. Model Quality for Isogeometric Analysis on Catmull-Clark Solids
C. Altenhofen, T. Ewald, A. Stork, and D. W. Fellner

Submitted to *IEEE Computer Graphics and Applications*
Special Issue on Geometric Modeling and Processing
IEEE Computer Society

In Isogeometric Analysis (IGA), the same basis functions and parametrization are used for representing the geometry and solving the PDEs of the simulation. This concept enables an efficient transition between the two disciplines – geometric design and simulation – since the same representation is used for both stages of the engineering workflow. However, just as with finite elements, IGA based on trivariate splines only provides C^0 continuity between elements, unless additional effort is spent to algorithmically create higher continuity. IGA on subdivision geometry (surfaces, as well as volumes) inherently provides higher continuity across element borders and also allows for representing topologically complex objects as single entities – a single subdivision surface or volume, respectively.

Motivated by Burkhart et al.'s proof-of-concept paper [BHU10b], this chapter analyzes the underlying aspects of performing IGA on subdivision volumes, i.e. on Catmull-Clark solids. In the context of structural analysis, I present an improved numerical integration technique, reducing the error when computing external forces and element stiffnesses for irregular CC-solid cells (see Section 5.1). I introduce a parametrization quality metric for the limit volume based on a quality measure for conformal mappings (see Section 5.2). The metric reveals local distortions and degenerate areas that negatively affect local derivatives as well as the solvability of the linear system. Furthermore, I present a set of cell splitting operations that improve the parametrization quality and resolve singularities at the boundary. (see Section 5.2.3). Finally, I perform IGA simulations on CC solids incorporating the findings from Sections 5.1 and 5.2. I compare the simulation results to an analytical solution for a simple scenario as well as to finite element simulations with linear and quadratic tetrahedral elements (see Section 5.3).

Section 5.1 contributes to answering Research Question 4:

RQ4: How can numerical integration be improved for irregular CC-solid cells?

Section 5.2 contributes to answering Research Question 5:

RQ5: How can parameterization problems be identified, quantified, and avoided if possible?

5.1. Numerical Integration on Catmull-Clark Solids

For calculating integrals where an analytical solution is too time-consuming or even impossible to compute, numerical integration is used. Different numerical integration schemes exist for integrating different kinds of functions. Quadratures, such as the Gauss-Legendre quadrature or the Newton-Cotes quadrature [AS72], are based on evaluating the function to be integrated at pre-defined quadrature points and adding the evaluated function values in a weighted sum, using pre-defined weights.

Most quadratures allow for integrating polynomial functions exactly with a certain number of quadrature points, depending on the chosen quadrature and the degree of the function to be integrated. For the Gauss-Legendre quadrature, n quadrature points can exactly integrate polynomial functions of degree $d \leq 2n - 1$. For the Newton-Cotes quadrature, n quadrature points can exactly integrate polynomial functions of degree $d \leq n - 1$. Therefore, for exactly integrating polynomial functions employing the cubic basis functions of regular Catmull-Clark surface patches and CC-solid cells, 2 Gauss-Legendre points or 4 Newton-Cotes points are required in each dimension.

For non-polynomial functions, the quadrature schemes approximate the integral by fitting a polynomial function and calculating its integral instead. The approximation error can be reduced to some extent by using more quadrature points – i.e. fitting a polynomial with a higher degree – but will not improve beyond a certain point as shown by Barendrecht [Bar13]. In this section, I present a hierarchical quadrature method that improves the integration accuracy for irregular CC-solid cells, utilizing the concepts of the constant-time limit evaluation presented in Section 3.2.2.

5.1.1. Calculating the Area of Limit Patches and the Volume of Limit Cells

For a better understanding of the different aspects of numerical integration, this section briefly recaps the calculation of limit face areas and limit cell volumes for CC solids.

The surface area of a limit patch is calculated by integrating over the corresponding patch in Euclidean space (x, y, z). Similarly, the volume of a limit cell equals the integral over the corresponding cell. Both integrals can be converted to parameter space (u, v), or (u, v, w), respectively, using standard integral transformation with the determinant of the Jacobian matrix $\mathbf{J}$.

The integrals can be calculated using a suitable quadrature scheme as shown in Equations (5.1) and (5.2), where $w_{i,j}$ and $w_{i,j,k}$ resemble the weights of the quadrature scheme, and $\mathbf{J}$ is evaluated at the quadrature points (q_i, q_j) and (q_i, q_j, q_k), respectively.

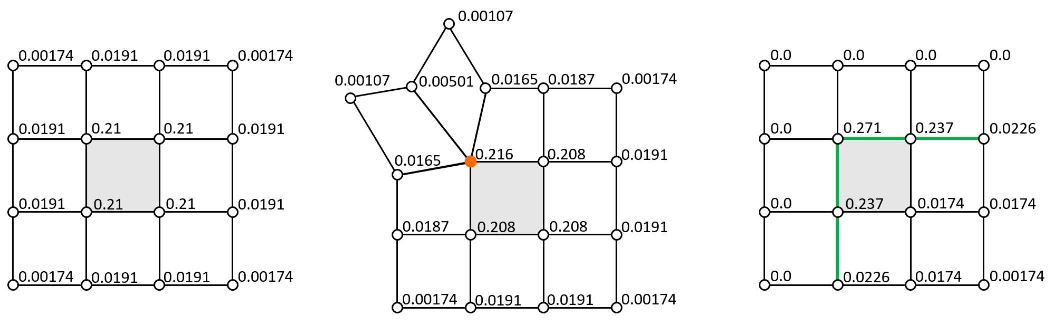
$$A = \sum_{i,j=1}^{n} w_{i,j} \sqrt{det(J_{i,j}^T J_{i,j})} \tag{5.1}$$

$$V = \sum_{i,j,k=1}^{n} w_{i,j,k} det(J_{i,j,k}) \tag{5.2}$$

Please note that the function for calculating the volume is polynomial for regular CC-solid cells, but the one for calculating the surface area is not due to the square root of $det(J_{i,j}^T J_{i,j})$.

5.1.2. Computing Weights for External Forces

When computing external forces for FEA and IGA simulations, the force assigned to a face has to be distributed over all nodes of that face, using the element's basis functions. For tetrahedral finite element meshes with linear basis function, the force per triangle is split equally on all three vertices of that triangle. For higher-order elements, the forces are distributed on the element's nodes using the corresponding higher-order basis functions.



(a) A regular patch

(b) An irregular patch with a valence of $\mathrm{N} = 5$

(c) A regular patch with crease edges

Figure 5.1.: Force weights for regular (a) and irregular surface patches (b), as well as for a patch with crease edges (c). The limit patch with a force assigned to it is highlighted in gray. The weights for the external forces are distributed over every control point that defines the given patch. All weights have to sum up to 1. The extraordinary vertex is marked in orange. Crease edges are marked in green.

As the basis functions of subdivision elements such as CC-solid cells span the element itself as well as its one-ring neighborhood, the force assigned to a Catmull-Clark surface patch with a valence of N has to be distributed onto all $2\mathrm{N} + 8$ control points that actually define the given patch (see Figure 5.1).

For Catmull-Clark surface patches, the weights W – a vector with one entry for every control point defining the corresponding patch – can be calculated by evaluating the subdivision basis functions $\varphi(u, v)$ at the quadrature points (q_i, q_j):

$$W = \sum_{i,j=1}^{n} w_{i,j}\varphi(q_i, q_j) \tag{5.3}$$

For regular patches, the regular B-spline basis functions $B(q_i, q_j)$ are used, while for patches containing extraordinary vertices or crease edges, the basis functions are obtained by applying Stam's limit evaluation method [Sta98a] as shown in Equation (2.13).

Figure 5.1 shows the weights for different surface patches. Independently from the topological configuration, the entries of W always have to sum up to 1. The actual force assigned to a given surface patch is multiplied with the corresponding weight for each control point to get the final force value for that control point.

As also described by Riffnaller-Schiefer, point forces that just affect a single control point are not supported by Catmull-Clark surfaces or solids [Rif19]. However, in most engineering use cases, forces are applied to individual or a set of surfaces that describe a functional region of the part to be simulated.

5.1.3. Computing Element Stiffnesses

In the context of physically based simulation – FEM as well as IGA – the stiffness matrix is a key part that defines the system of linear equations to be solved. The global stiffness matrix is constructed from the local element stiffnesses of all elements in the model, i.e. every CC-solid cell. The local element stiffness matrix $\mathbf{K}_{\mathbb{C}}$ for every CC-solid cell $\mathbb{C}$ is constructed by computing the integral as follows, with $\mathbf{D}$ being the *elasticity matrix* or *material matrix*:

$$\mathbf{K}_{\mathbb{C}} = \int_{\mathbb{C}} \mathbf{B}(\mathbf{x})^T \mathbf{D} \mathbf{B}(\mathbf{x}) \, \mathrm{d}\mathbb{C} \tag{5.4}$$

To clarify the difference between calculations performed in Euclidean space (x, y, z), and calculations performed in parameter space (u, v, w), I introduce the notations $\bar{x}$ and $\bar{u}$, respectively. As the stiffness matrix of a cell is computed in Euclidean space, the notations in Equation (5.4) can be changed to:

$$\mathbf{K}_{\mathbb{C}} = \int_{\bar{x}} \mathbf{B}_{\bar{x}}(\bar{x})^T \mathbf{D} \mathbf{B}_{\bar{x}}(\bar{x}) \, \mathrm{d}\bar{x} \tag{5.5}$$

The transformation of a parameter point (u, v, w) to its corresponding point (x, y, z) in Euclidean space is performed by the mapping function to the limit of the CC-solid model $\mathbf{f}(\bar{u})$. The integral itself can be converted from Euclidean space, to parameter space using standard integral transformation. This results in the formulation:

$$\begin{aligned} \mathbf{K}_{\mathbb{C}} &= \int_{\bar{x}} \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u}))^T \mathbf{D} \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u})) \, \mathrm{d}\bar{x} \\ &= \int_{\bar{u}} \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u}))^T \mathbf{D} \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u})) \, |det(\mathbf{J}(\mathbf{f}(\bar{u})))| \, \mathrm{d}\bar{u} \end{aligned} \tag{5.6}$$

The matrix $\mathbf{B}_{\bar{x}}$ contains partial derivatives of the basis functions N with respect to (x, y, z). However, $\mathbf{B}_{\bar{u}}$, containing the partial derivatives of the basis functions in (u, v, w) is much easier to compute.

$$\begin{aligned} \mathbf{J}_{\bar{x}} N(x, y, z) &\xrightarrow{map} \mathbf{B}_{\bar{x}} \\ \mathbf{J}_{\bar{u}} N(u, v, w) &\xrightarrow{map} \mathbf{B}_{\bar{u}} \end{aligned} \tag{5.7}$$

To calculate the derivatives in Euclidean space coordinates (x, y, z), a coordinate transformation has to be applied. With the help of this transformation, $\mathbf{B}_{\bar{x}}$ can be expressed by $\mathbf{B}_{\bar{u}}$ in the following way:

$$\mathbf{B}_{\bar{x}}(\bar{x}) = \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u})) = map\left(\frac{\partial \mathbf{f}(\bar{u})}{\partial \bar{x}}\right) \tag{5.8}$$

$$\begin{aligned} \frac{\partial \mathbf{f}(\bar{u})}{\partial \bar{x}} &= \frac{\partial \mathbf{f}(\bar{u})}{\partial \bar{u}} \frac{\partial \bar{u}}{\partial \bar{x}} \\ &= \frac{\partial \mathbf{f}(\bar{u})}{\partial \bar{u}} \left(\frac{\partial \bar{x}}{\partial \bar{u}}\right)^{-1} \\ &= \frac{\partial \mathbf{f}(\bar{u})}{\partial \bar{u}} \mathbf{J}(\mathbf{f}(\bar{u}))^{-1} \end{aligned} \tag{5.9}$$

$$\mathbf{B}_{\bar{x}}(\bar{x}) = \mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u})) = \mathbf{B}_{\bar{u}}(\bar{u})\, \mathbf{J}(\mathbf{f}(\bar{u}))^{-1} \tag{5.10}$$

To approximate the integral in equation (5.4), the integration is again performed numerically using a suitable quadrature. Applying the coordinate transformation and using numerical integration, the element stiffness matrices $\mathbf{K}_{\mathbb{C}}$ can be calculated as shown in Equation (5.11), with $\mathbf{B}_{\bar{x}}(\mathbf{f}(q_i, q_j, q_k))$ and $\mathbf{J}(\mathbf{f}(q_i, q_j, q_k))$ being evaluated at the quadrature points (q_i, q_j, q_k).

$$\mathbf{K}_{\mathbb{C}} = \sum_{i,j,k=1}^{n} w_{i,j,k} \mathbf{B}_{\bar{x}}(\mathbf{f}(q_i, q_j, q_k))^T \mathbf{D} \mathbf{B}_{\bar{x}}(\mathbf{f}(q_i, q_j, q_k))\ |det(\mathbf{J}(\mathbf{f}(q_i, q_j, q_k)))| \tag{5.11}$$

Please note that this calculation does not describe a polynomial function due to $\mathbf{J}(\mathbf{f}(\bar{u}))^{-1}$ being a part of $\mathbf{B}_{\bar{x}}(\mathbf{f}(\bar{u}))$ as shown in Equation (5.10).

For performing CSM on CC solids, the element stiffnesses define the system matrix of the linear system to be solved while the surface areas combined with the force weights define the right hand side. Therefore, accurate integration is especially important for these three components. In the following, I present a hierarchical quadrature scheme that improves the accuracy when integrating over irregular Catmull-Clark patches and cells, compared to using standard Gauss-Legendre quadrature as proposed by Burkhart et al. [BHU10b].

5.1.4. The Hierarchical Quadrature Scheme

Since the integration via quadrature points, e.g. using Gauss-Legendre or Newton-Cotes quadrature, is only accurate for polynomial functions, the integration scheme has to be adapted based on the function to be integrated and based on the topological configuration of the Catmull-Clark element – i.e. patch or cell – to be integrated over. While regular Catmull-Clark patches and cells provide polynomial basis functions, irregular ones do not. For irregular surface patches as well as for irregular cells, the corresponding basis functions do not have a single polynomial representation, but consist of an infinite series of polynomial functions.

The concepts used for evaluating the limit of irregular CC elements (see Sections 2.1.2 and 3.2.2) are utilized to create a hierarchical quadrature approach that reduces the approximation error compared to using standard Gauss-Legendre quadrature. The partitionings shown in Figures 2.7 and 3.12 are used to compute local integrals over regular sub-elements related to the parameters k and n. As the maximum number of hierarchical refinement steps n_{max} can be configured, the effect of different values for n_{max} is presented in Section 5.1.5.

In contrast to evaluating the limit, it is not sufficient that all quadrature points lie inside regular sub-elements. Instead, the integrals have to be computed using local quadrature points for each sub-element and are finally added to compute the integral of the entire patch or cell.

Surface Patches

For integrating over an irregular Catmull-Clark surface patch, it is partitioned into sub-patches $p_{n,k}$ by hierarchically bisecting its (u, v) space $[0, 1]^2$ as shown in Figure 2.7. For each sub-patch, the integral is calculated using standard Gauss-Legendre quadrature scaled to the given (u, v) interval. Since infinite refinement would be required to get accurate results, a small irregular part always remains that is approximated with standard Gauss-Legendre quadrature. The approximation error can be reduced by using a higher

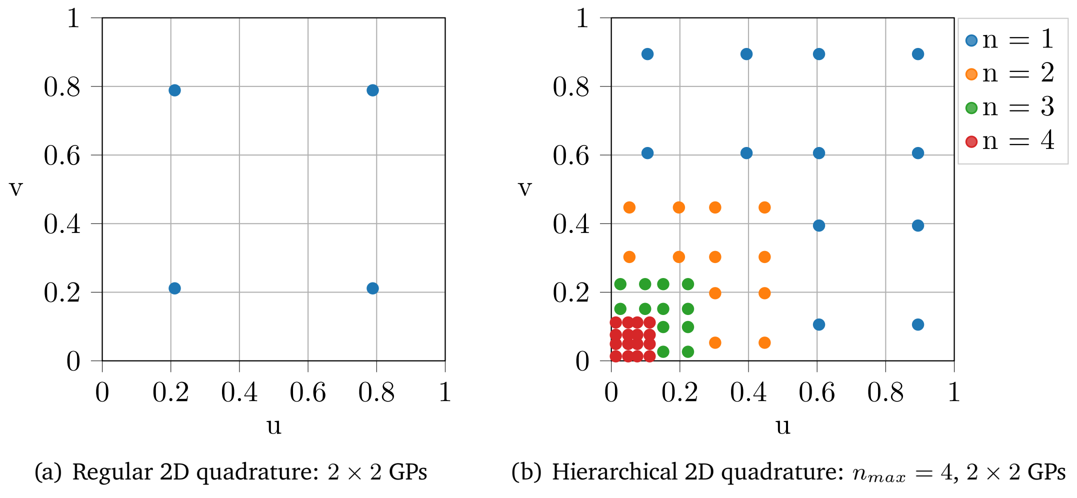


Figure 5.2.: Positions of the quadrature points in parameter space for regular (a) and irregular (b) CC-surface patches. The hierarchical quadrature follows Stam's partitioning of the (u, v) space [Sta98a] (see Figure 2.7), integrating over every sub-patch with a set of 2×2 Gaussian points (GPs).

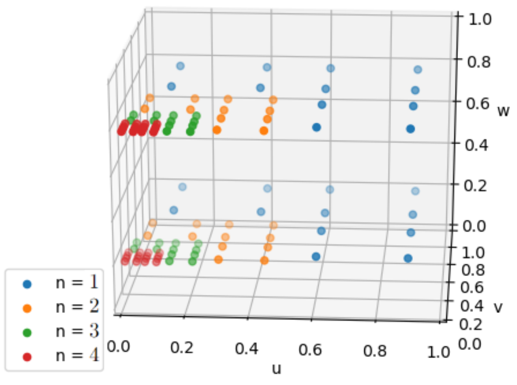
maximum number of refinement steps n_{max} (see Section 5.1.5). Finally, the integral over the initial surface patch p_0 is calculated as the sum over all sub-integrals:

$$\int p_0 = \sum_{n=1}^{n_{max}} \sum_{k=1}^{3} \int p_{n,k} + \int p_{n_{max},4} \tag{5.12}$$

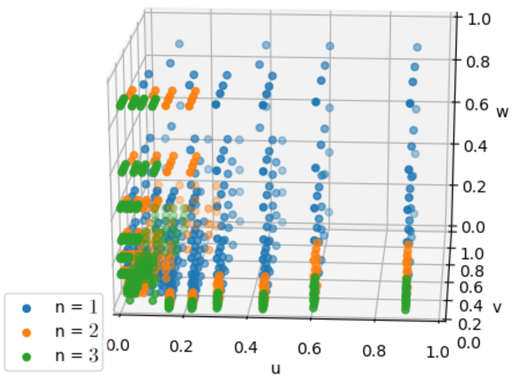
Figure 5.2 shows the distribution of quadrature points in the (u, v) space for regular as well as for irregular surface patches. For the latter, the EV is located at $(u, v) = (0, 0)$.

CC-Solid Cells

For cells with a layered topology – i.e. cells with one EE – the tensor product property of the CC-solid basis functions is utilized to combine a hierarchical 2D quadrature in u, v with a standard quadrature in w in the same way as the bivariate and univariate basis function are combined when evaluating the limit (see Section 3.2.2). The resulting distribution of quadrature points can be seen in Figure 5.3(a).



(a) Hierarchical layered quadrature



(b) Hierarchical irregular quadrature

Figure 5.3.: Distribution of quadrature points in the (u, v, w) space $[0, 1]^3$ for CC-solid cells with one EE (a) and three EEs (b). Both cells are shown with 2×2 Gaussian points per sub-cell and values of $n_{max} = 4$ (a) and $n_{max} = 3$ (b), respectively.

For integrals over CC-solid cells with two or three EEs, the integration domain $(u, v, w) \in [0, 1]^3$ is partitioned as shown in Figure 3.12. In analogy to the hierarchical surface integration, the integral for every sub-cell $c_{k,n}$ is calculated independently. Regular sub-cells are integrated with standard Gauss-Legendre quadrature. For layered sub-cells, the tensor product integration is used. The resulting distribution of quadrature points for a cell with three EEs is shown in Figure 5.3(b). Again, the maximum number of refinement steps n_{max} is limited and the integral of the remaining irregular sub-cell is approximated with standard Gauss-Legendre quadrature. This results in an approximation error that can be reduced with higher values of n_{max}. Finally, the integrals over all sub-cells are added:

$$\int c_0 = \sum_{n=1}^{n_{max}} \sum_{k=0}^{6} \int c_{n,k} + \int c_{n_{max},7} \tag{5.13}$$

The next section quantifies the improvements in accuracy of my hierarchical quadrature technique compared to using standard Gauss-Legendre quadrature for irregular Catmull-Clark patches and cells.

5.1.5. Accuracy Evaluation

In the context of computational structural mechanics (CSM) with CC solids, accurate numerical integration is possible for regular patches and cells when calculating the weights for external forces and when computing the volume of a limit cell. When calculating the surface area (see Equation (5.1)) or the element stiffness (see Equation (5.11)), quadratures designed for polynomial functions can only approximate the result apart from special cases when the Jacobian is constant throughout the patch or cell.

Surface Area

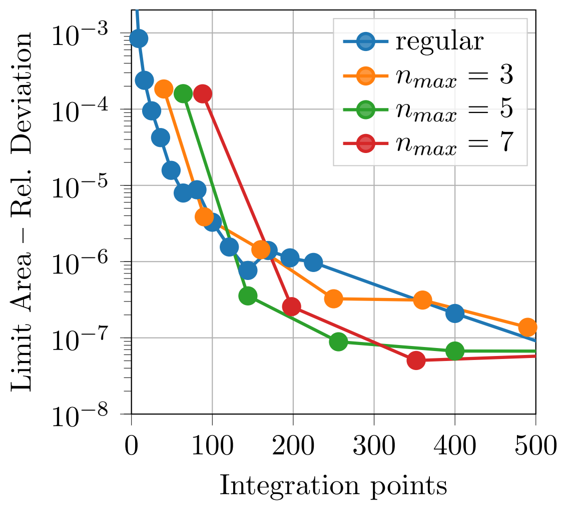
For evaluating the accuracy of the hierarchical quadrature technique for calculating the surface area, I applied standard Gauss-Legendre quadrature as well as my quadrature to irregular surface patches with valences of 3, 5, 6, and 7. I compare increasing numbers of Gauss-Legendre quadrature points (GPs) as well as increasing values for n_{max}. For all patches, a reference solution was created using a composite quadrature scheme, which is guaranteed to approximate the analytic solution up to a given threshold ϵ and up to rounding errors induced by using double precision floating point arithmetic [Han17]. At the same time, the composite quadrature required evaluating $10\times$ to $100\times$ more integration points than the hierarchical quadrature scheme.

The results are shown in Figure 5.4. The diagrams visualize the relative deviation d_{rel} of the surface area a from the reference solution a_{ref} for the different quadrature schemes. The relative deviation d_{rel} was calculated as follows:

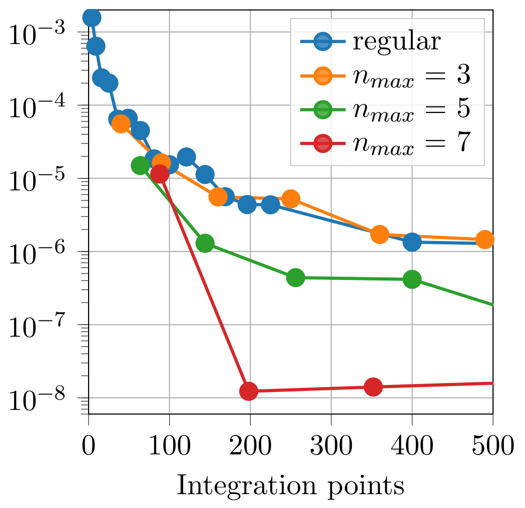
$$d_{rel} = \frac{|a - a_{ref}|}{a_{ref}} \tag{5.14}$$

As can be seen, performing more refinement steps – i.e. choosing higher values for n_{max} – increases the accuracy of the integration since the hierarchical quadrature can better represent the piecewise cubic basis function of the irregular surface patch. However, for surface patches with a valence of 3, performing more than 5 refinement steps does not provide additional benefit.

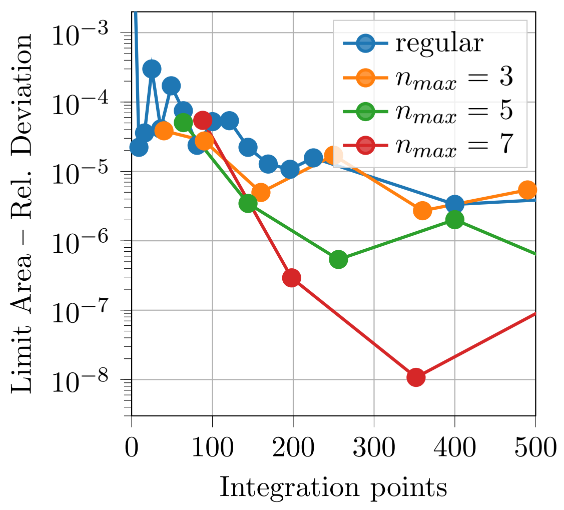
As, the surface area is defined by a non-polynomial function (see Equation (5.1)), the integrals have to be approximated even for regular sub-patches. While at first, increasing the number of Gaussian points per sub-patch improves accuracy, the results converge when using more than 4×4 GPs per sub-patch. For valences larger than 5, increasing the



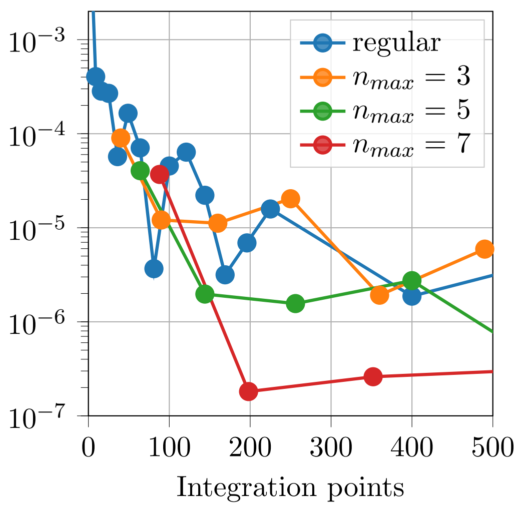
(a) Valence 3



(b) Valence 5



(c) Valence 6



(d) Valence 7

Figure 5.4.: Relative deviation from the reference solution for computing the limit surface area using regular Gauss-Legendre quadrature as well as the hierarchical quadrature with 2×2 to 30×30 GPs and increasing values of n_{max}. The integrals have been computed for irregular CC-surface patches with valences 3 (a), 5 (b), 6 (c), and 7 (d).

number of GPs even results in an oscillating behavior. This also matches the findings of Barendrecht who investigated the behavior of standard Gauss-Legendre integration for CC surfaces [Bar13].

Choosing $n_{max} = 5$ for patches with a valence of 3 and $n_{max} = 7$ for patches with valences larger than 4 improves the accuracy of the integrals by up to three orders of magnitude compared to standard Gauss-Legendre quadrature, while using the same total amount of integration points.

Element Stiffness

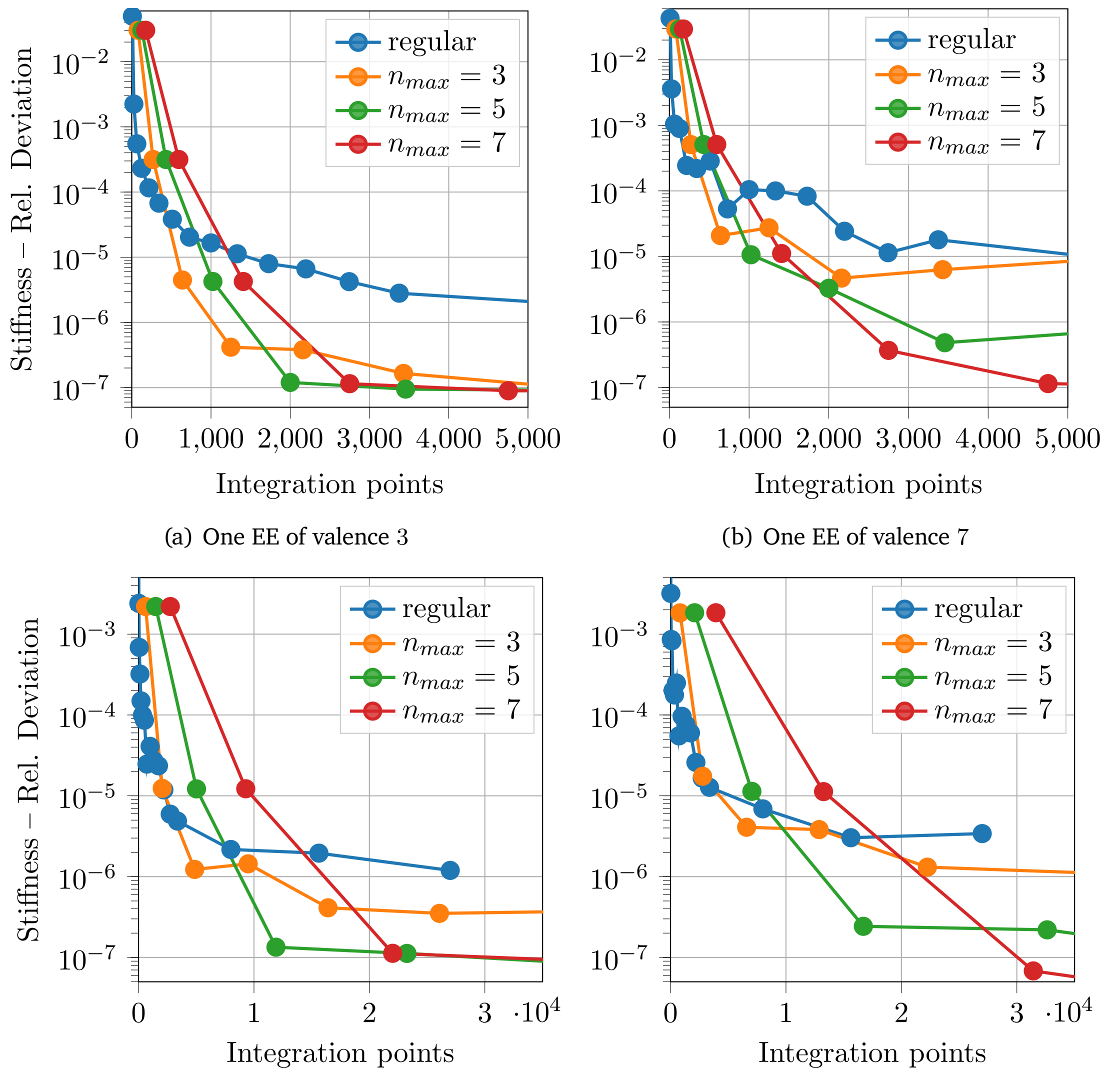
For evaluating the hierarchical quadrature for the element stiffnesses of irregular CC-solid cells, I used seven different topological configurations:

- Three different layered cells, each with one EE of valence 3, 5, and 7, respectively
- An irregular cell with two EEs of valences 3 and 5
- Three different irregular cells, each with three EES – the valences are $(3, 3, 3)$, $(5, 5, 5)$, and $(7, 7, 7)$, respectively

Again, increasing numbers of GPs, as well as increasing values for n_{max} were used for the evaluation. The material matrix $\mathbf{D}$ was computed from the material parameters of construction steel, i.e. a Young's modulus of $1.16e + 11$ and a Poisson's ratio of 0.3.

Figure 5.5 shows the evaluation results for computing the element stiffnesses of the limit cells. The diagrams visualize the relative deviation from the reference solution for the different quadrature schemes. As for the limit surface areas, the reference solution was created using a composite quadrature scheme. For the element stiffnesses, the composite quadrature required evaluating up to $1000\times$ more integration points than the hierarchical quadrature scheme. The deviation was computed using the *Frobenius* matrix norm [GC96]. Only the diagrams for four out of all seven configurations are shown in Figure 5.5. The other three configurations produced analogous results during the evaluation.

The diagrams show a similar behavior as the 2D integrals for computing the surface areas. Increasing n_{max} improves the accuracy of the integration for up until 5 refinement steps. Additional refinement steps are only beneficial if all EEs in the corresponding cell have valences higher than 5. Increasing the number of Gauss-Legendre points per sub-cell increases the accuracy for up until $4 \times 4 \times 4$ GPs for layered cells and up until $5 \times 5 \times 5$ GPs for cells with more than one EE. Similar to computing the surface areas, spending additional GPs results in oscillation or even divergence for higher valences.



(a) One EE of valence 3

(b) One EE of valence 7

(c) Two EEs of valences 3 and 5

(d) Three EEs, all with valences 5

Figure 5.5.: Relative deviation from the reference solution for computing the element stiffnesses using standard Gauss-Legendre quadrature as well as the hierarchical quadrature with $2 \times 2 \times 2$ to $30 \times 30 \times 30$ Gaussian points and increasing values of n_{max}. The integrals have been computed for layered CC-solid cells with valences 3 (a) and 7 (b), as well as for cells with two EEs (c) and three EEs (d).

Compared to using standard Gauss-Legendre quadrature as done by Burkhart et al. [BHU10b], the hierarchical quadrature scheme improves the accuracy for calculating the element stiffness by one to two orders of magnitude, while using the same total amount of points. Please note that although my method reduces the deviations to the reference solutions by several orders of magnitude for individual patches and cells, the deviations are already below 0.01% when calculating the integrals with standard Gauss-Legendre quadrature. Section 5.3.5 shows the influence of the different integration methods on the simulation results.

5.1.6. Performance

Since the integration points created by the hierarchical quadrature scheme are closer to the EEs and EVs than the ones used for standard Gauss-Legendre quadrature, Burkhart et al.'s subdivision-based limit evaluation approach [BHU10b] would require additional local subdivision steps. With the constant-time volumetric limit evaluation technique presented in Section 3.2, the required integration points can be evaluated very efficiently, also for complex topologies with multiple EEs. Therefore, the computational effort for calculating the integrals scales linearly with the number of integration points, independent from their location in the (u, v) or (u, v, w) space, respectively.

The goal is to find a good trade-off between accuracy and performance, i.e. the total number of integration points. As shown by the evaluation results, a high number of refinement steps is preferred over a high number of Gaussian points per sub-patch or sub-cell. Derived from the results presented in Section 5.1.5, the following combinations are the most beneficial ones:

- For surface patches with a valence of 3, $n_{max} = 5$ is chosen together with 4×4 GPs per sub-patch.
- All other surface configurations use $n_{max} = 7$ and 4×4 GPs.
- In the volumetric case, $n_{max} = 5$ is chosen whenever an EE of valence 3 is present, while all other topological configurations use $n_{max} = 7$.
- Layered cells are evaluated using $5 \times 5 \times 5$ GPs while cells with multiple EEs use $4 \times 4 \times 4$ per sub-cell.

These values are also used for the simulations shown in Section 5.3.5.

Apart from accurate integration, having good mesh quality is very important when performing physically based simulations. Badly shaped or degenerated elements significantly affect the quality of the simulation results in terms of convergence rate as well as correctness of the calculated solution. The next section presents my approach to quantify and improve the quality of CC-solid models for IGA.

5.2. Parametrization Quality of the Limit Volume

In order to rate *good* and *bad* elements in FEA, the concept of *mesh quality* has been introduced, usually calculated based on ratios of angles, lengths, areas and volumes. As shown in Section 4.3 for tetrahedral meshes, different geometric quality measures can be calculated for every element (see also the work by Field [Fie00] and by Shewchuk [She02]).

For subdivision solids, the *mesh quality* does not only depend on the geometric properties of the control mesh, or its individual cells, respectively, but also on the *quality* of the limit. Similarly as for trivariate B-splines and NURBS, a geometrically well-shaped CC-solid control mesh can still create degenerate limit cells if the transfer function $\mathbf{f}$ from parameter space (u, v, w) to Euclidean space (x, y, z) induces distortions or singularities. For CC solids, $\mathbf{f}$ resembles the transformation from the control mesh to the limit volume (see Section 3.1.3). Therefore, I introduce a metric to quantify the quality of the parametrization of the limit volume, based on quality measures for conformal mappings presented by Fu et al. [FLG15].

5.2.1. Quantifying Parametrization Quality

As proposed by multiple papers (e.g. by Massarwi and Elber [ME16], Rabinovich et al. [Rab+17], and Feng et al. [Fen+18]), the Jacobian matrix $\mathbf{J}$ can be used to define a quality metric for $\mathbf{f}$. Inspecting $\mathbf{J}$ and especially its determinant $det(\mathbf{J})$ gives a basic quality measure. Determinants close to 0 indicate degenerated regions, while negative determinants indicate inversions. Determinants $det(\mathbf{J})$ of exactly 0 reveal singularities where $\mathbf{f}$ loses its injectivity. Consequently, the concept of *mesh parametrization regularity* (MPR) is based on bounding $det(\mathbf{J})$ as far away from zero as possible.

For robust solvability of the system of linear equations, focusing on a good MPR is not sufficient. Inspecting the singular values of $\mathbf{J}$ reveals details about local distortions. However, the singular value decomposition is expensive to compute. The eigenvectors and eigenvalues of $\mathbf{J}^T\mathbf{J}$ contain the same information and can be analyzed instead. $\mathbf{J}^T\mathbf{J}$ results in a symmetric matrix having the squared derivatives w.r.t. u, v, and w on its main diagonal and the products of the derivatives in (u, v), in (u, w) and in (v, w) on its off-diagonals, respectively. While the values on the main diagonal represent the scaling along the corresponding u, v, or w axis, the products on the off-diagonals represent the change in angle between the two corresponding coordinate axes. The eigenvectors of $\mathbf{J}^T\mathbf{J}$ resemble the directions of the main distortions while the eigenvalues λ_i describe the scaling along theses directions.

For my parametrization quality metric, I inspect $\mathbf{f}_{\mathbb{C}}$ for each CC-solid limit cell $\mathbb{C}$ and measure the deviation from a globally scaled isometry. This is done using isometric deformation energies, a well-known quantity within the mesh parametrization community. I define $\mu(u, v, w)$ as the *mesh parametrization quality* (MPQ) for a single parameter point (u, v, w) inside $\mathbb{C}$ as follows, with V being the non-zero volume of $\mathbb{C}$:

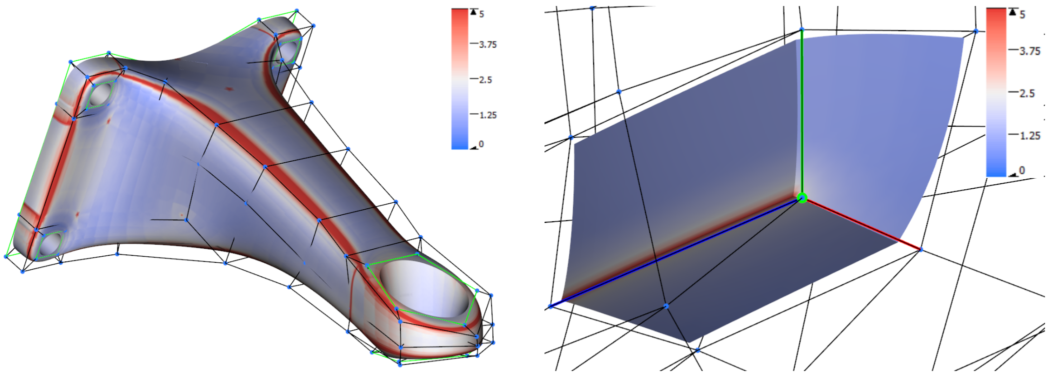
$$\mu(u, v, w) := \max \left\{ \frac{\sqrt{\lambda_{max}}}{\sqrt[3]{V}}, \frac{\sqrt[3]{V}}{\sqrt{\lambda_{min}}} \right\} \tag{5.15}$$

Please note, that $\sqrt{\lambda_i}$ is used instead of λ_i due to analyzing $\mathbf{J}^T\mathbf{J}$ instead of $\mathbf{J}$ itself. The point measure $\mu(u, v, w)$ is optimal – the optimal MPQ value is 1.0 – if and only if $\sqrt{\lambda_1} = \sqrt{\lambda_2} = \sqrt{\lambda_3} = \sqrt[3]{V}$, i.e. the geometry of the limit cell is a uniformly scaled unit cube. Higher values indicate distortions, while a values of $\mu(u, v, w) \to \infty$ indicate singularities.

I define $\mu_{\mathbb{C}}$ as the MPQ of a limit cell $\mathbb{C}$ as follows:

$$\mu_{\mathbb{C}} := \max_{(u,v,w)\in[0,1]^3} \mu(u, v, w) \tag{5.16}$$

Hence, better MPQ – i.e. lower values of $\mu_{\mathbb{C}}$ – results in better solvability of the system of linear equations that defines the given simulation problem. Figure 5.6 shows the point measure $\mu(u, v, w)$ applied to the engine mount model. Blue areas indicate a good MPQ while red areas indicate distortions and singularities. Such singularities occur at the boundary for cells with at least two connected boundary faces. As can be seen in Figure 5.6(b), extraordinary edges also locally distort the parametrization in their direct vicinity. This is due to the vanishing derivatives close to EEs and EVs. This is a well-known problem in the subdivision community and actually affects the quality of the simulation results.



(a) MPQ of the engine mount model (b) Cell with an EE inside the engine mount model

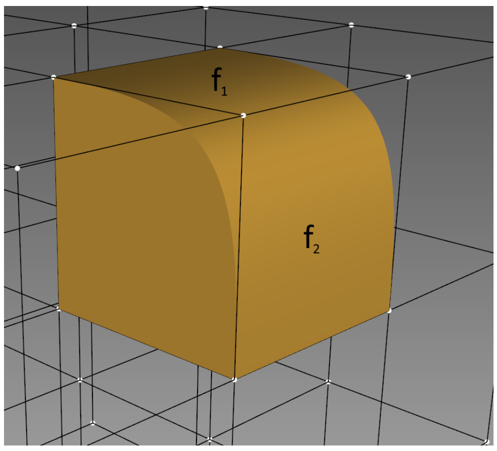
Figure 5.6.: Mesh parametrization quality (MPQ) of the engine mount model. Especially cells with more than one boundary face induce local distortions and singularities. A single cell out of the engine mount model is shown in (b), revealing parametric distortions located around an EE. The parametrization quality is color-coded from blue (good) to red (high distortions/singularities).

5.2.2. Critical Topological Configurations

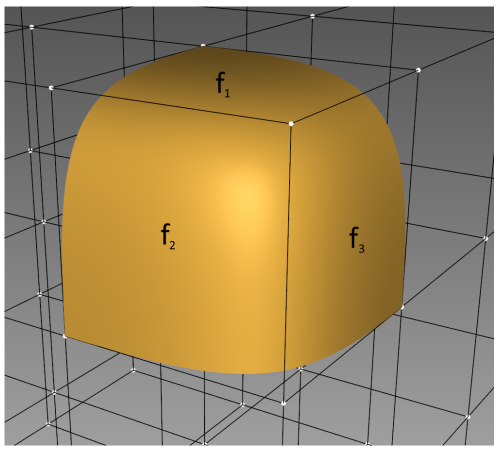
The parametrization quality metric presented in Section 5.2.1 allows measuring local distortions as well as revealing singularities for the CC-solid limit volume. Resulting from the subdivision rules, CC solids inherently induce singularities at the boundary, i.e. in cells with more than one connected boundary face.

Figure 5.7 shows two examples, where the control mesh is well-shaped in terms of geometric mesh quality but the transformation to the limit induces distortions and singularities as shown in Figures 5.7(c) and (d). The singularities appear at the limit points corresponding to the shared edges of two boundary faces, resulting in collapsing derivatives and a non-injective mapping $\mathbf{f}$.

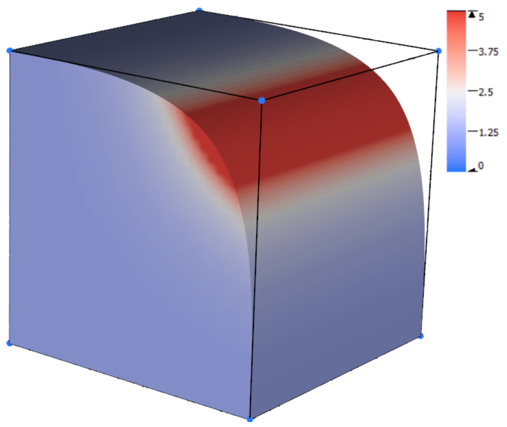
For having a good parametrization quality and thereby a high accuracy of the simulation results, a given CC-solid model should contain as few boundary cells with two or more connected boundary faces as possible. As shown by Cohen et al. [Coh+10], volumetric control meshes – for trivariate NURBS as well as for CC solids – can be designed in a way that those critical boundary cells are avoided as much as possible. Masswari and Elber adapted Cohen et al.'s concepts for parametrizing several primitives in their



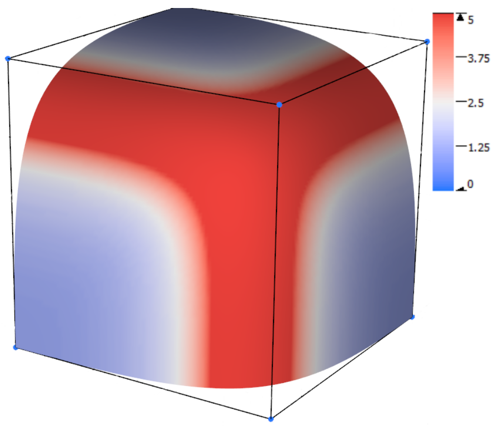
(a) Cell with two boundary faces f_1 and f_2



(b) Corner cell with three boundary faces f_1, f_2, and f_3



(c) MPQ of (a)



(d) MPQ of (b)

Figure 5.7.: Mesh parametrization quality (MPQ) of limit cells with two and three boundary faces. As can be seen, CC solids induce degenerate areas along their boundary wherever a cell contains two or more connected boundary faces. Although the control mesh is well-shaped in terms of geometric mesh quality (a, b), singularities appear along common edges of boundary faces, resulting in $det(\mathbf{J}) = 0$ and removing the injectivity of $\mathbf{f}$ (c, d). The MPQ is color-coded from blue (good) to red (high distortions/singularities).

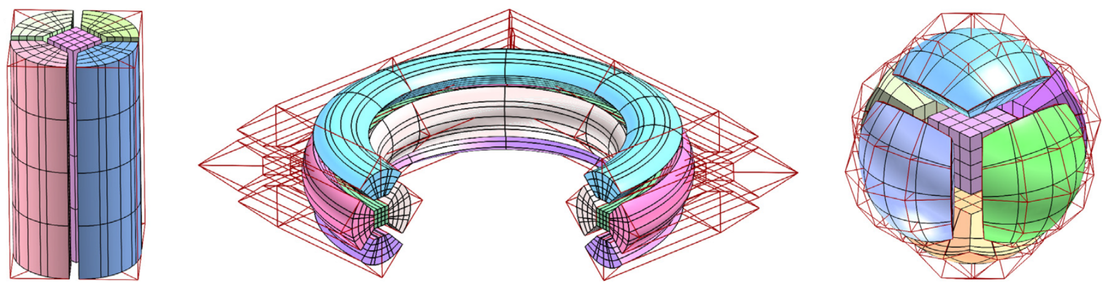

(a) Cylinder (b) Torus (c) Sphere

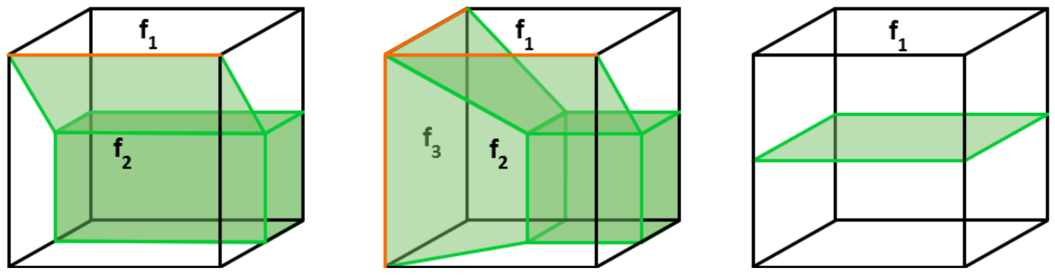
Figure 5.8.: Parametrization of different trivariate NURBS primitives following the recommendations presented by Cohen et al. [Coh+10]. The images were taken from the 2016 paper by Massarwi and Elber [ME16].

framework [ME16], which is based on Boolean operations (see Figure 5.8). However, those meshes are not well suited for interactive freeform modeling. As an alternative, cells with more than one connected boundary face can be split in a way that all resulting cells have just one boundary face, while at the same time maintaining the limit surface of the CC-solid model. This splitting approach is presented in the next section.

5.2.3. Splitting Cells to Improve the Parametrization Quality

CC-solid cells with multiple connected boundary faces can be split in different ways to create only cells with one boundary face. The individual splitting schemes differ by the number of EEs and EVs created, as well as by the number of neighboring cells that are affected as the splitting operations propagate throughout the model. The less EEs and EVs are created when splitting and the less propagation is introduced, the better. In terms of MPQ and therefore numerical stability of the simulation, propagation is always preferred over new EEs or even EVs due to the vanishing derivatives in their vicinity. However, the presence of EEs and EVs is not as problematic as the singularities at the boundary. As shown in Section 5.3.5, such singularities have a much bigger effect on the simulation results than internal EEs and EVs.

In 2005, Yong and Cheng presented a splitting scheme for quadrilaterals, splitting every quad face into three new quadrilaterals [YC05]. Although Yong's approach was designed for local refinement of Catmull-Clark surfaces, it inspired my method for splitting boundary cells in CC-solid control meshes.



(a) Cell with two boundary faces (b) Three boundary faces (c) Cell with one boundary face

Figure 5.9.: Splitting scheme for cells with different numbers of boundary faces. The boundary faces are located at the top and the front (a), at the top, front, and left (b), and at the top (c) of the corresponding cell. Edges that are shared by two boundary faces are highlighted in orange. The newly created edges and faces are shown in green.

Splitting Scheme

For boundary cells with two neighboring boundary faces, a 2.5D adaptation of Yong's scheme is used to split the CC-solid cell into three hexahedral cells as shown in Figure 5.9(a). Two cells have one boundary face each, while the third cell has no connection to the boundary. The splitting procedure introduces an extraordinary edge of valence 3 between the three new cells.

Boundary cells with three connected boundary faces – so-called *corner cells* – are split using a 3D interpretation of Yong's pattern. It splits the cell into four hexahedral cells, three of which have one boundary face each as shown in Figure 5.9(b). The fourth cell does not have any boundary faces. The splitting creates three EEs of valence 3 and an EV shared by those three EEs.

Splitting the cells in the described fashion propagates to all neighboring boundary cells with just one boundary face. Those cells have to be split into two cells as shown in Figure 5.9(c) in order to maintain a topologically valid control mesh. All splitting operations are performed by applying sequences of volumetric Euler operators (see Section 3.3.2).

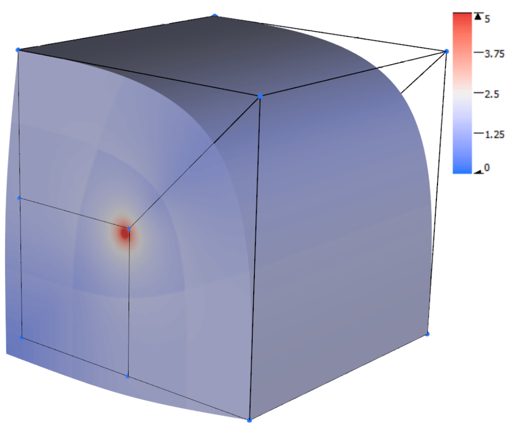
Motivated by Yong's refinement scheme preserving the limit for CC surfaces [YC05], my split operations were also designed to maintain the limit surface and therefore the overall shape of 3D object. The splitting scheme does not create new control points on the boundary.

MPQ Results

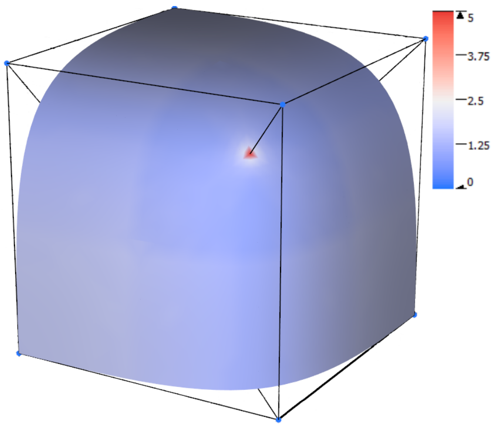
Table 5.1 shows the minimum, maximum, median and average values for the mesh parametrization quality of the cells shown in Figure 5.7 as well as for a cell with just one boundary face and a cell in one leg of the tripod model before and after being split. The cell with one boundary face had the shape of a scaled unit cube with an optimal MPQ of 1.0 before it was split into two cells. For each cell, the MPQ has been measured at regular sample points in $(u, v, w) \in [0, 1]^3$. As the MPQ cannot be evaluated directly at singularities, EEs, and EVs due to vanishing derivatives, it has been measured with a distance ϵ of 1^{-10}, instead. This ensured numerical stability and consistency for all measurements. While the median MPQ value has increased (worsened) by $1.25\times$ to $1.59\times$ after splitting, the average value has decreased (improved) to between $\frac{1}{25}$ and $\frac{1}{360}$ of the original value for cells with multiple boundary faces. This behavior resembles the trade-off between resolving the singularities at the boundary and introducing new EEs and EVs inside the cells. The drastic decrease in the maximum MPQ value for each cell shows that the distortions around the newly created EEs and EVs are much less severe than the singularities at the boundary. The impact on the simulation results is analyzed in Section 5.3.5.

Cell	**MPQ (lower = better)**			
	min.	**max.**	**median**	**avg.**
2 Boundary Faces	1.03	626282	1.03	149.4
2 BFs Split	1.34	13342	1.47	2.32
3 Boundary Faces	1.05	6969240	1.09	2783.7
3 BFs Split	1.01	632266	1.35	7.72
1 Boundary Face	1.0	1.0	1.0	1.0
1 BF Split	1.59	1.59	1.59	1.59
Tripod Cell	1.31	220645	1.68	73.25
Tripod Split	1.84	6779.8	2.41	2.87
Tripod Loop-Cut	1.15	205425	1.37	61.1
Tripod LC + Split	1.36	7362.5	1.81	2.33

Table 5.1.: Mesh parametrization quality (MPQ) for different cells with one or multiple boundary faces. The table shows the minimum, maximum, median and average MPQ values before and after performing the split. For the cell selected from the tripod model, an additional loop-cut is performed to improve the aspect ratios of the split cells. The results are visualized in Figures 5.10 and 5.11.



(a) MPQ after splitting a cell with two boundary faces

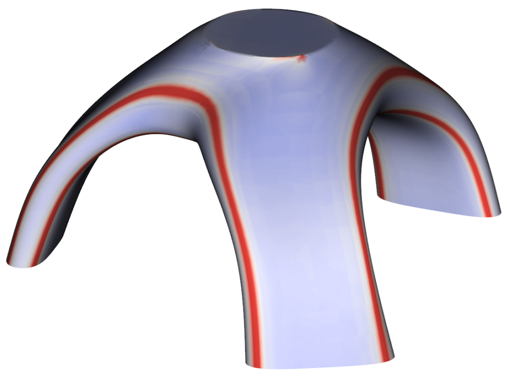


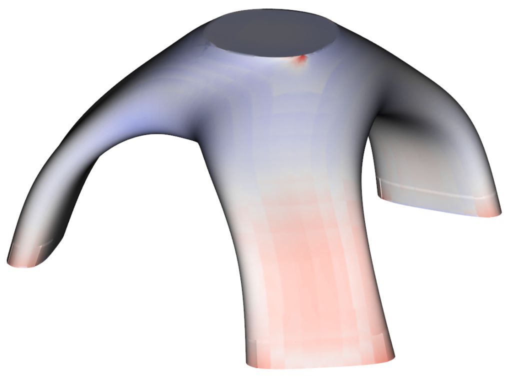
(b) MPQ after splitting a cell with three boundary faces

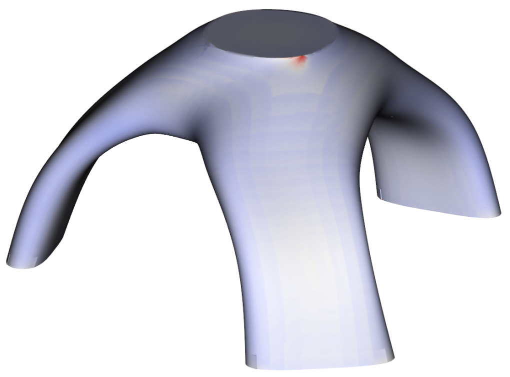
Figure 5.10.: MPQ after splitting the boundary cells shown in Figure 5.7. The MPQ is color-coded from blue (good) to red (high distortions). The same color scale was applied as in Figures 5.7(c) and (d). Although the parametrization quality is slightly worse throughout the cells and especially around the newly created EEs, the singularities at the boundary have been removed.

Figure 5.10 visualizes the MPQ after splitting boundary cells with two and three connected boundary faces. Compared to the MPQ of the original cells (see Figures 5.7(c) and (d)), the singularities at the boundary have been resolved. Away from the boundary the distortion has increased due to the modified aspect ratio of the individual cells. Around the newly created EEs, the effects of the vanishing derivatives can be seen. Note that although the limit surface is maintained when splitting boundary cells, the limit of the interior cells changes as can be seen in the lower left corner when comparing Figures 5.7(c) and 5.10(a).

Figure 5.11 shows the result of splitting the boundary cells of the tripod model. As for the individual cells shown in Figure 5.10, the splitting removes the singularities at the boundary, but increased the distortions throughout the rest of the model due to changing the aspect ratios of the cells. Performing several loop-cuts (see Section 3.3.3) restores the original aspect ratios and improves the MPQ away from the boundary, achieving almost the original quality values (see Figure 5.11(c)). However, as described earlier, the EEs and EVs created by splitting the boundary cells induce new distortions inside the model. Figure 5.11(d) shows a volumetric visualization of the tripod model, revealing the distortions around the EEs as red dots.


(a) MPQ of the original tripod model


(b) MPQ after splitting boundary cells


(c) MPQ after splitting and loop-cutting

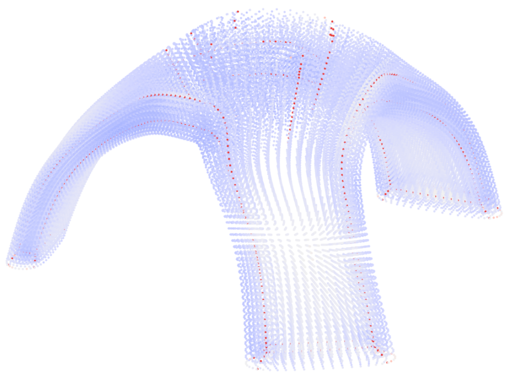

(d) Volumetric visualization

Figure 5.11.: MPQ of the tripod model before (a) and after splitting the boundary cells (b). The slight reduction of the parametrization quality away from the boundary can be counteracted by performing several loop-cuts (c). (d) shows a volumetric visualization of the tripod, revealing the distortions of the newly created EEs and EVs. The parametrization quality is color-coded from blue (good) to red (high distortions/singularities). All models use the same color mapping as in Figures 5.7 and 5.10.

5.3. Structural Analysis with Catmull-Clark Solids

This section describes my IGA approach based on CC solids in the context of computational structural mechanics (CSM). The method builds on the concepts presented by Burkhart et al. [BHU10b], but utilizes my constant-time volumetric limit evaluation method (see Section 3.2), the hierarchical quadrature technique (Section 5.1.4), as well as the findings from analyzing the mesh parametrization quality (Section 5.2).

The simulations solve time-independent linear elasticity problems for the four different simulation scenarios shown in Figure 5.12. Section 5.3.5 contains the evaluation of my method. It shows a comparison of IGA on CC solids with FEA on linear and higher-order tetrahedra, including an analytical solution for the first scenario. Furthermore, I analyze the impact of the hierarchical quadrature and of splitting boundary cells on the simulations.

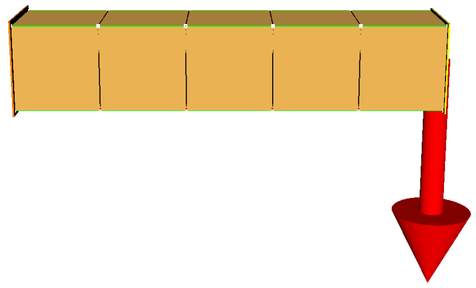
5.3.1. Initial Subdivision Steps

For the volumetric limit of the CC-solid mesh to be evaluatable, it has to fulfill the requirements presented in Section 3.2. If necessary, one or two initial subdivision steps are applied on the control mesh in a pre-processing step. A higher number of initial subdivision steps can be used to create additional degrees of freedom (DoFs) to accurately represent the simulation problem at hand.

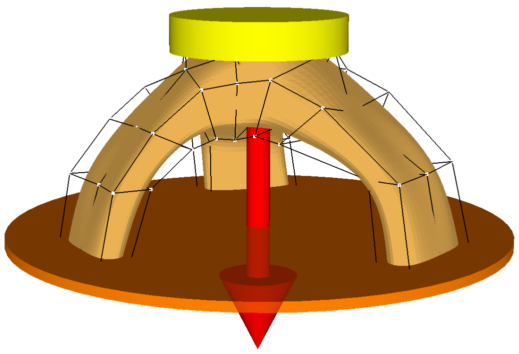
As long as the topology of the model remains constant, the subdivision matrices for these initial subdivision steps can be stored as shown in Section 3.1.2 in order to rapidly react on geometrical changes of the model.

5.3.2. Applying External Forces

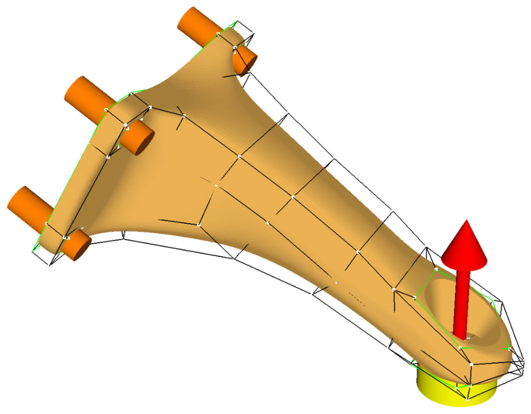
Area forces are applied as external forces to a set of patches of the CC-solid limit surface. Having defined an external force on the model, the force is distributed over every limit patch that is affected by that force. First, the force is split according to the surface area of every involved patch. The surface areas of the corresponding patches are calculated as described in Section 5.1.1. Second, the fraction of the force for every patch is split on all control points of that patch, using the subdivision basis functions as described in Section 5.1.2. The final force at a control point is the sum of the split forces calculated for every patch influenced by that control point.



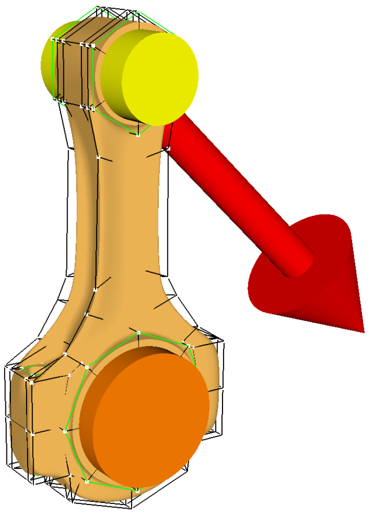
(a) Bar



(b) Tripod



(c) Engine mount



(d) Connecting rod

Figure 5.12.: The four simulation scenarios used for evaluation. The bar model (a) consists of five CC-solid cells with no extraordinary edges. The boundary edges of the bar are set crease. The tripod model (b) is constructed out of 136 cells with EEs of valences 3 and 5. The plateau on top of the tripod is enclosed by a ring of crease edges. The engine mount model (c) consists of 48 cells, again containing several EEs and crease edges. The connecting rod model (d) consists of 146 cells with multiple EEs, EVs, and crease edges. The volumetric Catmull-Clark control mesh is shown in black while the limit surface is shown in light brown. Fixations are visualized in orange, forced areas are visualized in yellow with force directions being shown as red arrows.

For irregular surface patches, the integration is performed with the hierarchical quadrature approach presented in Section 5.1.4. Measurements for the influence of improved integration on the accuracy of the surface areas and therefore also on the external forces can be found in Section 5.1.5.

5.3.3. Assembling the Stiffness Matrix

In CSM, the stiffness matrix resembles the central part of the system of linear equations (SLE) that has to be solved for the given simulation problem. The assembly of the global stiffness matrix $\mathbf{K}$ is performed by calculating the local element stiffness matrix $\mathbf{K}_{\mathbb{C}}$ for every CC-solid cell $\mathbb{C}$ as shown in Section 5.1.3, using Equation (5.11). The entries of $\mathbf{K}_{\mathbb{C}}$ are inserted into the global stiffness matrix $\mathbf{K}$ at the respective positions.

Again, the integration over irregular CC-solid cells is performed with my hierarchical quadrature scheme. Measurements for the influence of numerical integration on the accuracy of the stiffness matrix can be found in Section 5.1.5.

5.3.4. Solving the System

As the IGA approach with CC solids results in an SLE $Ax = b$ similar to those from using tetrahedral or hexahedral elements, any suitable technique for solving SLEs can be used, e.g. factorization, a Jacobi solver, a conjugate gradient (CG) solver, or a multigrid solver. However, the matrix A resulting from using CC solids is denser than the one for linear or higher-order tetrahedra. This has to be considered when choosing a suitable solver. In my implementation, I use the conjugate gradient solver provided by the C++ library *Eigen* [JG08].

Table 5.2 shows the amount of non-zero entries in A for Catmull-Clark solids in comparison with tetrahedra with linear and quadratic basis functions. For the four models used in the evaluation, the relative density of A is higher by $3.4\times$ to $28.8\times$ when using CC solids. The exact factor depends on the topologies of the CC-solid mesh and the tetrahedral mesh used for the simulation.

Model	E. Type	#DoFs	Non-Zeroes	Density
Bar	CC solids	1500	609408	27.08%
	Tet $d = 1$	870	27036	3.572%
	Tet $d = 2$	780	48060	7.899%
Tripod	CC solids	30402	22095666	2.391%
	Tet $d = 1$	30402	1234080	0.1335%
	Tet $d = 2$	31374	2367630	0.2405%
Engine mount	CC solids	11703	6778035	4.949%
	Tet $d = 1$	11799	448749	0.3223%
	Tet $d = 2$	12810	906930	0.5527%
Connecting rod	CC solids	31527	22445739	2.258%
	Tet $d = 1$	29595	1020249	0.1165%
	Tet $d = 2$	83808	5498928	0.0783%

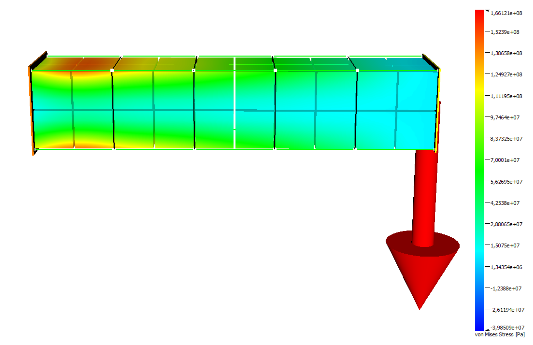
Table 5.2.: The number of non-zero entries and the corresponding relative density of the system matrix for the four models shown in Figure 5.12. Since the Catmull-Clark basis functions include the one-ring neighborhood of every cell, the matrices are less sparse for CC solids compared to linear or higher-order tetrahedra. Linear systems with sparse matrices are generally easier to solve.

5.3.5. Simulation Results

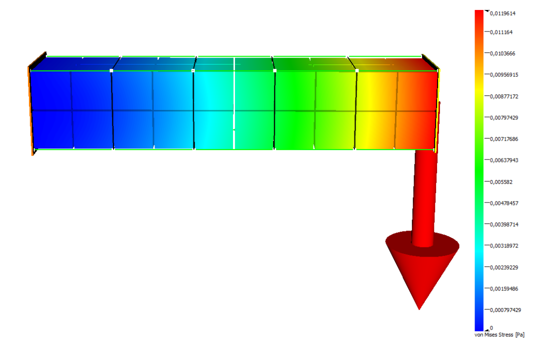
For the evaluation, I performed simulations of the following four scenarios:

1. A loaded horizontal bar fixed on the left side with a force applied to the right
2. A tripod fixed on the ground and loaded with a vertical force from the top
3. An engine mount fixed on its back and loaded with a vertical force in the front
4. A connecting rod fixed on its lower ring and loaded with a diagonal force on its upper ring

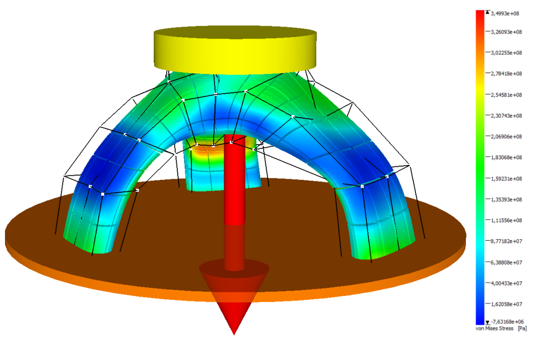
All four models and the corresponding simulation scenarios are visualized in Figure 5.12. The results for the IGA simulations on CC solids are visualized in Figures 5.13 and 5.14, showing the color-coded *von Mises* stress and displacement, as well as the boundaries of the CC-solid limit cells.



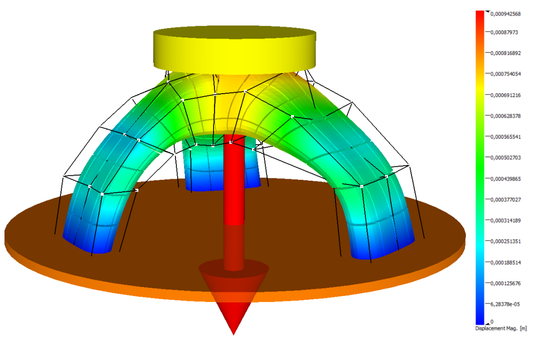
(a) Stress results for the bar model



(b) Displacement results for the bar model

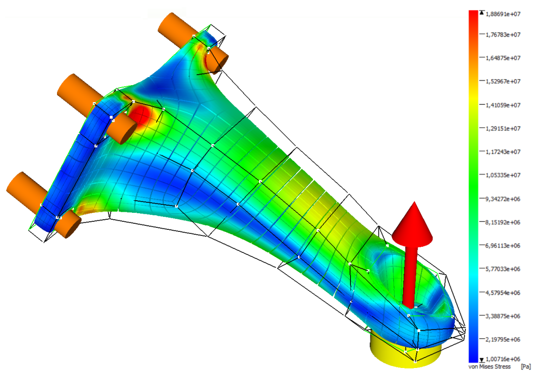


(c) Stress results for the tripod model

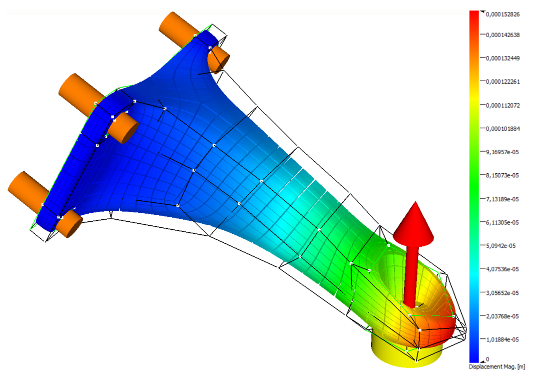


(d) Displacement results for the tripod model

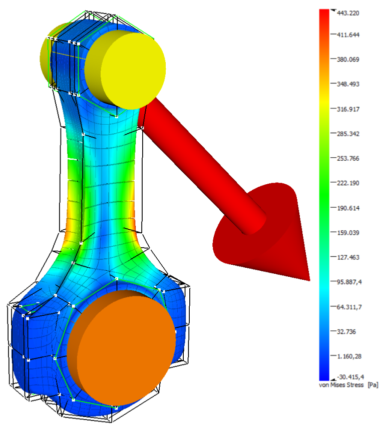
Figure 5.13.: Visualization for the von Mises stress (a, c), as well as the magnitude of the displacement (b, d). Blue color indicates low stress or displacement values, respectively, while red color indicates high values. The Catmull-Clark limit cells resulting from the initial subdivision steps are visualized using a slight scaling. The CC-solid control mesh is shown in black.



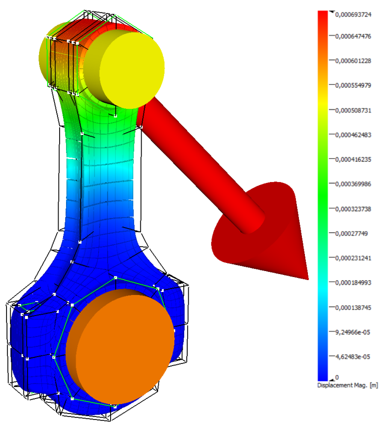
(a) Stress results for the engine mount model



(b) Displacement results for the engine mount



(c) Stress results for the connecting rod

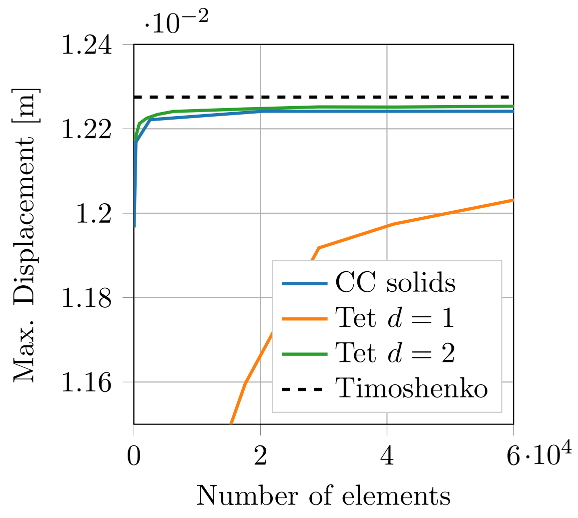


(d) Displacement results for the connecting rod

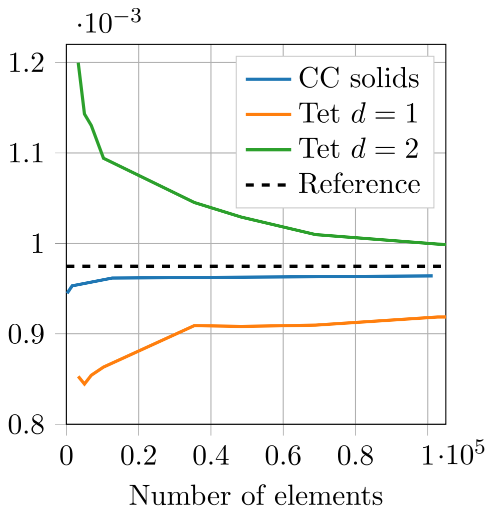
Figure 5.14.: Visualization of the von Mises stress (a, c), as well as the magnitude of the displacement (b, d). Again, blue color indicates low stress or displacement values, respectively, while red color indicates high values. The limit cells are visualized using a slight scaling and the control mesh is shown in black.

Comparison with Tetrahedral FEA

The simulations were performed with CC solids, as well as with linear and quadratic tetrahedral elements. The tetrahedral meshes were created from the tessellated CC-solid limit surfaces using CGAL's meshing algorithm [Jam+15]. For the bar model, I also compare the simulation results with an analytical solution. Applying Timoshenko's beam theory [BC09] gives a maximum displacement of 0.01228 meters, for a bar with a size of $5 \times 1 \times 1$ meters with a Young's modulus of $2.1 * 10^{11}$ Pascal and a vertical force of $5 * 10^{6}$ Newton. For the tripod model, I created a reference solution by meshing and simulating the model in a commercial FEA framework using 227188 quadratic tetrahedral elements with 338501 nodes. For the rest of this section, the term *element* will be used for tetrahedra as well as for CC-solid cells.



(a) Displacement results for the bar model

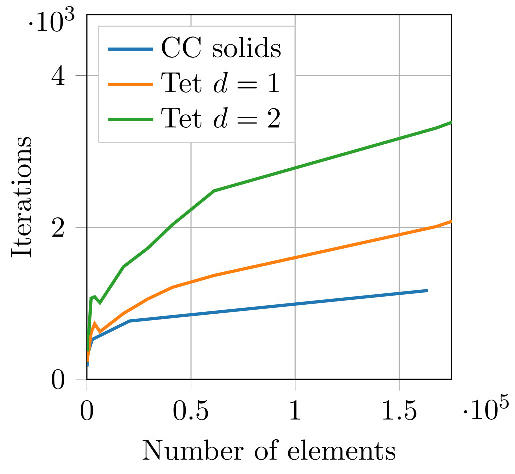


(b) Displacement results for the tripod model

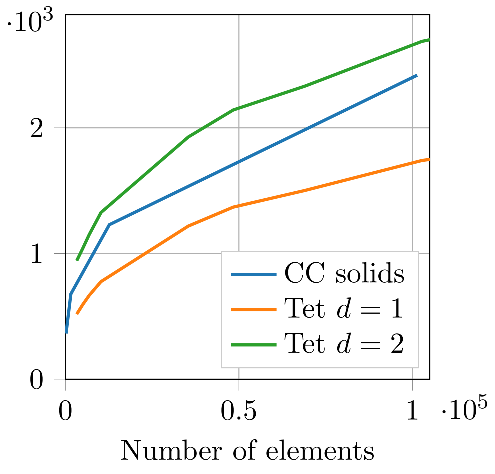
Figure 5.15.: Displacement results for the simulations on Catmull-Clark solids and on tetrahedra with linear and quadratic basis functions. For the bar model (a), quadratic tetrahedra as well as CC solids converge rapidly towards the analytic solution based on Timoshenko's beam theory [BC09]. For the more complex tripod scenario (b), CC solids converge with fewer elements, compared to linear and higher-order FEA. The reference solution in (b) was created with a commercial FEA framework.

Figure 5.15 shows the convergence of the maximum displacement over an increasing number of elements for the loaded bar as well as for the tripod model. As can be seen for the bar, all higher-order methods (including CC solids) converge rapidly towards the analytical solution. For the tripod, the results for CC solids converge with much fewer elements and much fewer DoFs than for linear or quadratic tetrahedra. The other two scenarios showed analog results.

Figure 5.16 shows the number of iterations needed by the CG solver to solve the system of linear equations for an increasing number of elements. For the bar model, CC solids require the lowest number of iterations to solve the SLE for any number of elements. For the tripod model, the SLEs obtained from linear tetrahedra require fewer iterations than the ones from CC solids for the same number of elements. However, CC solids still provide a more accurate solution to the simulation problem for that given number of elements as shown in Figure 5.15(b). Again, the other two simulation scenarios showed analog results to the tripod scenario, regarding the number of required iterations.



(a) Number of iterations required for the bar



(b) Number of iterations for the tripod model

Figure 5.16.: Number of iterations required by the conjugate gradient solver to solve the linear system. For the bar model, solving the system created from CC solids requires the fewest iterations for a given number of elements (a). For the tripod model, linear tetrahedra require fewer iterations than CC solids for the same number of elements (b). However, CC solids show more accurate results for the corresponding number of elements (see Figure 5.15(b)).

Influence of Improved Integration

For measuring the effects of the two different integration approaches – hierarchical quadrature and standard Gauss-Legendre quadrature – the CC-solid simulations have been performed with both methods.

When simulating the loaded bar, the results are identical for both quadrature schemes since the Jacobian **J** is constant throughout the entire model, i.e. all limit cells are uniformly scaled unit cubes. For the other three scenarios, the differences in the maximum displacement between the two methods are below 0.15%. Solving the linear systems created with the hierarchical quadrature required about 0.1% fewer iterations than solving the system created with standard Gauss-Legendre quadrature.

The effect of the hierarchical quadrature on the simulation results is confined by the portion of layered and irregular cells in the CC-solid model to be simulated. Table 5.3 shows the distributions of regular, layered (1 EE), and fully irregular cells (multiple EEs) for the tripod and the connecting rod model. Every subdivision step decreases the portion of non-regular cells at an exponential rate. The differences in the simulation results also decrease exponentially for higher subdivision levels.

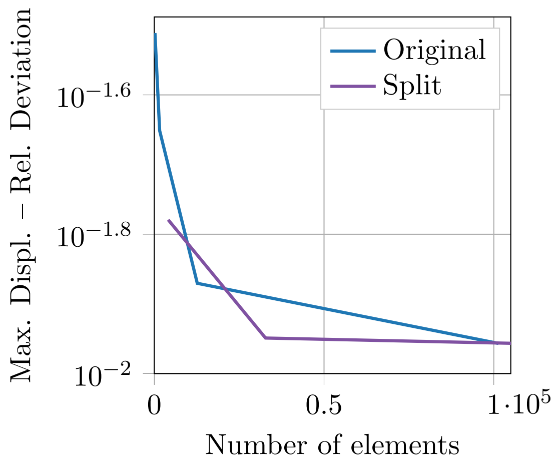
Model	**Subdiv. level**	**Cell Type**			
		Regular	**Layered**	**Fully irregular**	**Not evaluatable**
Tripod	0	0.0 %	0.0 %	0.0 %	100.0 %
	1	28.3 %	62.6 %	8.1 %	1.0 %
	2	79.4 %	19.6 %	1.0 %	0.0 %
	3	94.6 %	5.3 %	0.1 %	0.0 %
Connecting rod	0	5.5 %	2.7 %	8.2 %	83.6 %
	1	39.9 %	22.6 %	27.9 %	9.6 %
	2	80.3 %	11.7 %	8.0 %	0.0 %
	3	94.8 %	3.3 %	1.9 %	0.0 %

Table 5.3.: Distribution of cell types for the tripod and the connecting rod model. Both models can be evaluated after two subdivision steps. The percentage of regular cells increases with every subdivision steps. Each layered cell is subdivided in 2 layered and 6 regular cells. Each irregular cell is subdivided into 1 irregular cell, and depending on the number of EEs, 2 or 3 layered cells and 5 or 4 regular cells, respectively.

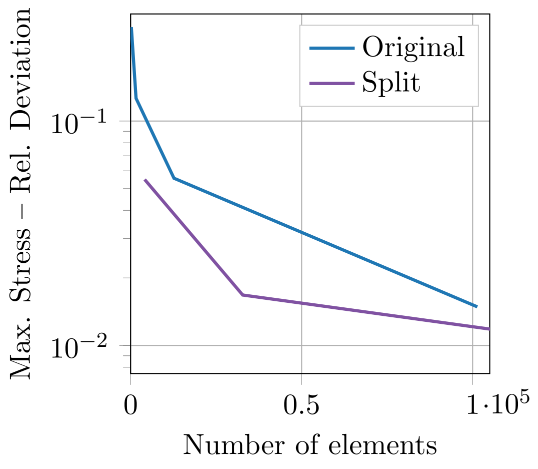
Influence of Improved Parametrization Quality

Figure 5.17 shows the relative deviation from the reference solution regarding the maximum displacement and maximum von Mises stress for the standard tripod model and the tripod with split boundary cells. For stress and displacement, the improved mesh parametrization quality reduces this deviation and improves the convergence of the simulation results – the results are stable for fewer elements and fewer DoFs. The relative errors of the stress and displacement values throughout the entire model were below 1.6% and 1.8% for both variants – with and without splitting – on all resolutions.

Figure 5.18 shows the number of iterations and the computation time required for solving the linear system. The additional EEs and EVs created by splitting the boundary cells increase the number of iterations compared to using the standard tripod model. However, the iterations can be performed with a reduced computational effort since the overall time to solve the SLE is similar for both model variants as shown in Figure 5.18(b).

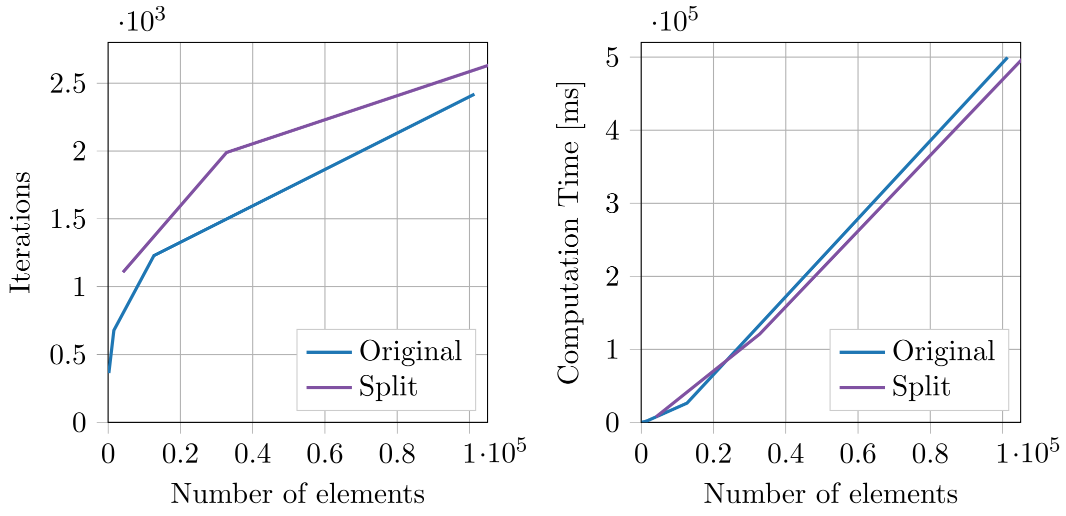


(a) Maximum displacement for the tripod model



(b) Maximum von Mises stress for the tripod model

Figure 5.17.: Influence of splitting boundary cells on the simulation results (IGA, tripod model). (a) shows the relative deviation from the reference solution regarding the maximum displacement, while (b) shows the relative deviation for the maximum von Mises stress. The reference solution was created from a high-resolution mesh using a commercial FEA framework.



(a) Number of iterations for solving the SLE

(b) Computation time required by the solver

Figure 5.18.: Influence of splitting boundary cells on the performance of the simulation (IGA, tripod model). (a) shows the number of iterations required for solving the linear system, while (b) shows the computation time required by the solver. As can be seen, the additional iterations required for the split model can be performed in less time as the total runtime of the solver is almost identical for both models.

As the results converge for fewer elements and fewer degrees of freedom, the computational costs and memory consumption for generating stable simulation results is reduced even further when using CC-solid IGA with split boundary cells compared to standard CC-solid models or even tetrahedral FEA.

5.4. Summary

This chapter discussed IGA on CC solids in the context of structural analysis. It focused on the two important aspects that were not investigated in Burkhart et al.'s proof-of-concept paper [BHU10b], i.e. numerical integration and mesh quality. Additionally, I compared my IGA method for CC solids with an analytical solution as well as with linear and higher-order tetrahedral FEA.

My integration approach adapts to the topology of the element to be integrated over, leading to a hierarchical quadrature scheme that better represents the piecewise cubic basis function of irregular Catmull-Clark surface patches as well as irregular CC-solid limit cells. This is especially important when calculating surface areas and element stiffnesses as those components mainly define the linear system of the simulation. My method reduces the deviations of these integrals from the corresponding reference solutions by up to three orders of magnitude compared to using standard Gauss-Legendre quadrature as proposed by Burkhart et al. [BHU10b]. Using my constant-time limit evaluation, the computation time scales linearly with the number of integration points, although they are much closer to the EEs and EVs with the hierarchical scheme compared to the standard approach. However, as shown in Sections 5.1.5 and 5.3.5, the original deviations of less than 0.01% seem to be *accurate enough* for the relatively low number of irregular cells in the CC-solid models. The difference in the simulation results for using the hierarchical quadrature instead of standard Gauss-Legendre quadrature is below 0.15%. Independently from CC solids or IGA in general, state-of-the-art hexahedral meshes such as the ones produced by Solomon et al. [SVB17] and Liu et al. [Liu+18], are also dominated by regular cells.

To assess local distortions and singularities in the limit volume, I introduced a mesh parametrization quality (MPQ) metric based on mesh parametrization regularity. The metric reveals that the limit parametrization of Catmull-Clark solids degenerates around EEs and EVs, but especially at the boundary where multiple boundary faces meet in a single cell. The latter case also creates singularities with $det(\mathbf{J}) = 0$. I presented a set of splitting operations for boundary cells in order to improve the parametrization quality in these areas. While splitting boundary cells worsens the median MPQ by $1.25\times$ to $1.59\times$ compared to the original boundary cells, the average MPQ value is improved by $25.5\times$ to $360\times$. The method resolves the singularities at the boundary and restores the injectivity of the transfer function $\mathbf{f}$ from the control mesh to the limit. The measurements in Section 5.3.5 have shown that splitting the boundary cells improves the accuracy and the converge of the simulation results while inducing no additional computational effort for the solver.

Comparing the simulation results of my IGA approach to those obtained using linear and higher-order tetrahedral meshes as well as to an analytical solution showed that my approach requires fewer elements and fewer degrees of freedom than linear or higher-order tetrahedra for the simulation results to converge to the reference solution. Additionally, the number of iterations required for solving the systems of linear equations is lower for CC solids. However, the systems are less sparse, which results in higher computational cost per iteration compared to tetrahedral FEA.

6. Applications

The content of this chapter is based on the following papers:

Volumetric Subdivision for consistent implicit Mesh Generation
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner

Published in *Computers & Graphics*
Volume 69, Supplement C, 2017
Elsevier

Integrating Interactive Design and Simulation for Mass Customized 3D-Printed Objects – A Cup Holder Example
C. Altenhofen, F. Loosmann, J. S. Mueller-Roemer, T. Grasser, T. H. Luu, and A. Stork

Published in *Solid Freeform Fabrication Symposium*
Volume 28, 2017
The Minerals, Metals & Materials Society TMS

Continuous Property Gradation for Multi-Material 3D-Printed Objects
C. Altenhofen, T. H. Luu, T. Grasser, M. Dennstädt, J. S. Mueller-Roemer, D. Weber, and A. Stork

Published in *Solid Freeform Fabrication Symposium*
Volume 29, 2018
The Minerals, Metals & Materials Society TMS

This chapter presents four applications of my research results. For every application, I implemented a prototype system, containing one or multiple of the concepts and methods presented in Chapters 3, 4, and 5.

The first application, presented in Section 6.1, showcases integrated 3D modeling and Finite Element Analysis. Employing the volumetric modeling concepts described in Section 3.3, I implemented an interactive modeling and simulation framework based on Catmull-Clark solids. The framework uses the efficient tetrahedral mesh generation approach described in Chapter 4 together with a GPU-based FEM solver based on the approaches by Weber et al. [Web+13], enabling a tight integration between geometric modeling and physically based simulation.

The second application (see Section 6.2) shows interactive customization of 3D-printed objects. The prototype utilizes parameters for CC solids as presented in Section 3.3.4 for interactively modifying the shape of a given 3D object. Combined with the efficient tetrahedral mesh generation (see Chapter 4) and the GPU-based FEM solver, the prototype enables an interactive exploration of the object's design space, providing instant feedback from the FE simulation and suggestions on how to improve the current design with respect to a given objective function.

The third application, presented in Section 6.3, deals with the design and manufacturing of multi-material AM parts. The application is based on CC-solid models, attaching material property values to the control points as described in Section 3.1. It allows for designing 3D objects with smoothly graded material distributions on the surface as well as volumetrically inside the object, which are tedious to model with conventional tools. The designed 3D object can be manufactured using a multi-material 3D printer. A recent study underlines that smoothly graded material transitions provide superior mechanical properties compared to sharp material boundaries [Aer18].

The fourth application (Section 6.4), uses CC solids for defining internal structures for additive manufacturing. First of all, inner structures reduce the weight, material costs and printing time of the manufactured parts. Inner structures can also improve the thermal situation during the printing itself, increasing the geometric precision in the manufacturing process [Tam+20]. Just based on their geometry, such structures can act as so-called metamaterials, emulating varying material properties in a single-material printing scenario and even inducing innovative physical behavior such as auxetic behavior or anisotropic, non-linear, or discontinuous stiffness. By approximating the limit surface as a B-spline B-Rep, the CC-solid-based structures can be exported back into the CAD system that was used to design the original part. This allows for further processing and also simulating the combined part, evaluating the physical behavior of the 3D object with its internal structures. Additionally, the approach can be used to not just define solid and empty regions inside a given 3D object, but also to create regions with smoothly graded material properties inside an otherwise uniform part using the techniques presented in Section 6.3.

6.1. Integrating Modeling and Simulation with CC Solids and Tetrahedral Meshes

This section describes a prototypical integrated modeling and simulation system, which consists of the following two components:

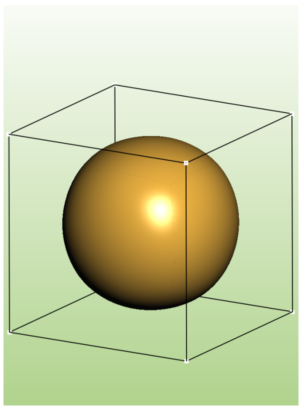
1. An interactive 3D modeling environment for Catmull-Clark solids
2. A GPU-based finite element solver for computational structural mechanics (CSM)

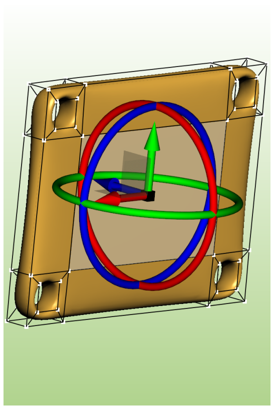
The interactive modeling environment allows for generating and manipulating CC-solid models similarly to existing modeling tools for subdivision surfaces, while at the same time maintaining the volumetric representation underneath. The CC-solid model is converted into a tetrahedral mesh using the efficient mesh generation method presented in Chapter 4. For the finite element simulation, a GPU-based FEM solver is used, based on the methods presented by Weber et al. [Web+13]. The effect of design changes can be seen immediately as the tetrahedral mesh and the finite element simulation quickly react to modifications of the CC-solid model. The simulation results are visualized directly on the model to identify and respond to critical points in the current design.

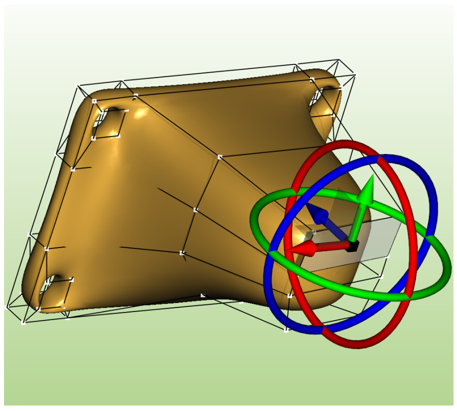
6.1.1. Interactive CC-Solid Modeling

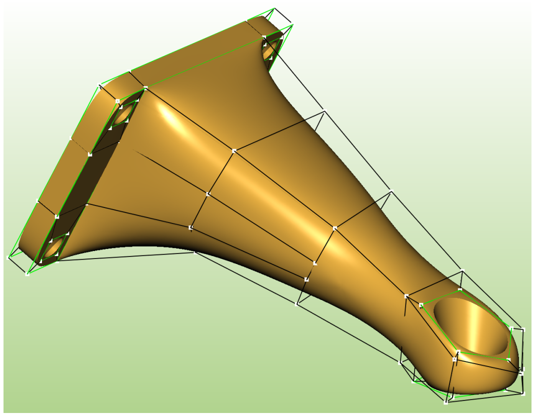
The modeling system offers functionality for generating and modifying CC-solid objects using the concepts of polyhedral modeling. It offers a set of volumetric modeling operations similar to the ones found in the polygonal modeling functionality offered by common tools such as Blender [Fou20] and Maya [Aut18]. Since the outer limit surface of a CC-solid model is identical to the limit surface of CC surfaces, there is no visual difference when designing 3D models with either Catmull-Clark surfaces or solids.

All modeling operations apply one or a sequence of multiple volumetric Euler Operators (see Section 3.3.2). To form the shape of the 3D object, one or multiple vertices, edges, faces or cells can be translated, rotated and scaled. To edit the topology of the object, new cells can be added to any outer face. To add more detail, additional vertices can be inserted into edges, faces or cells, splitting these into two or more edges/faces/cells. To create loops or holes, it is also possible to connect two boundary faces with one another by either merging them or by connecting them with a new cell. Analogously to Catmull-Clark surfaces, sharp features can be added by defining crease edges on the volumetric control mesh. The control mesh can be globally subdivided (with and without smoothing)


(a) The initial hexahedral control mesh and its smooth limit surface


(b) The back plate of the engine mount model without crease edges


(c) Intermediate version of the model after adding additional cells towards the front end


(d) The final engine mount model including crease edges

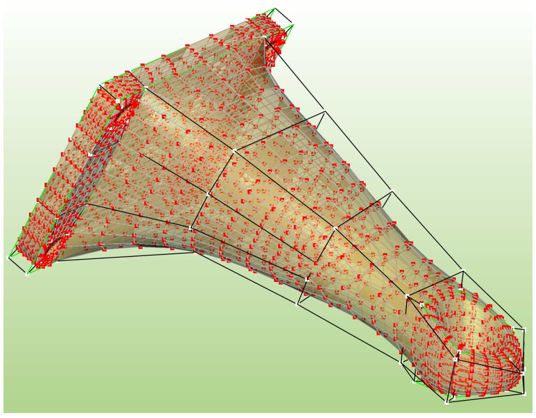
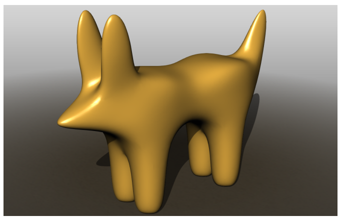
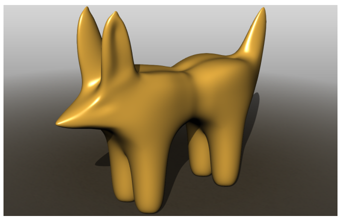

(e) The volumetric structure of the model after two subdivision steps

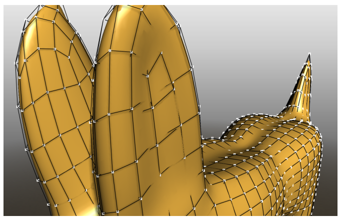
Figure 6.1.: Creating a CC-solid model with the volumetric modeling system. (a) to (d) show the progress from an initial hexahedral control mesh to the final design. The interaction widget shown in (b) and (c) is used to manipulate the control mesh. Selected faces are highlighted in gray. (e) shows the volumetric structure that is maintained during the entire modeling process and which is also the basis for the mesh generation approach.

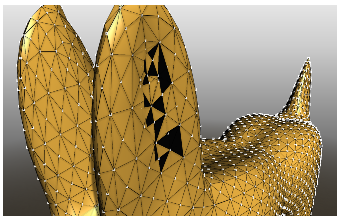
to iteratively increase the mesh resolution and to add details on lower levels. As the modeling operations are not restricted to the surface of the model, the inner structure can be modified as well. Figure 6.1 shows an example of the modeling process from the initial geometry to a complete model.

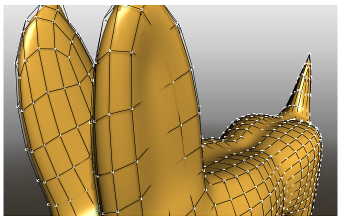
Using consistent modeling operations based on combinations of Euler operators, the created volumetric control meshes are manifold by definition. There is no way to break them except for self-intersection. However, self-intersection is a major problem. Allowing the user to arbitrarily manipulate the geometry of the object might lead to situations, where vertices are moved in a way that faces intersect other faces of the mesh. As this might happen with a surface mesh as well, it is much more likely to happen – and much less visible to the user – when modeling volumetric meshes. If inner vertices are not


(a) The original coyote model


(b) The modified coyote model with additional details


(c) The coyote's ear showing intersections of outer and inner control points


(d) The tetrahedral mesh created from (c), showing inverted elements in black


(e) The coyote's ear after applying Laplacian smoothing to the inner control points

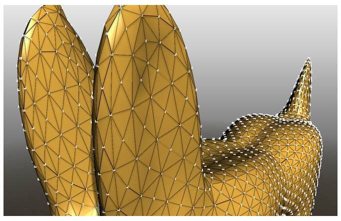

(f) The tetrahedral mesh created from (e) without element inversion

Figure 6.2.: When modeling the outer surface of a volumetric model (a, b), self-intersections with the unchanged inner control points might occur (c),leading to invalid tetrahedral meshes (d). Applying Laplacian smoothing to the inner control points resolves the intersections (e) and creates a consistent tetrahedral mesh for simulation (f).

moved accordingly when manipulating the outer surface, they will at some point intersect with the mesh boundary while the user keeps modeling. For approximating subdivision schemes such as Catmull-Clark surfaces and solids, self-intersections of the control mesh do not necessarily lead to self-intersections of the subdivided mesh, but they can be used as an approximation as it is a necessary but not sufficient criterion. However, surface modeling tools usually do not provide any assistance for preventing self-intersections.

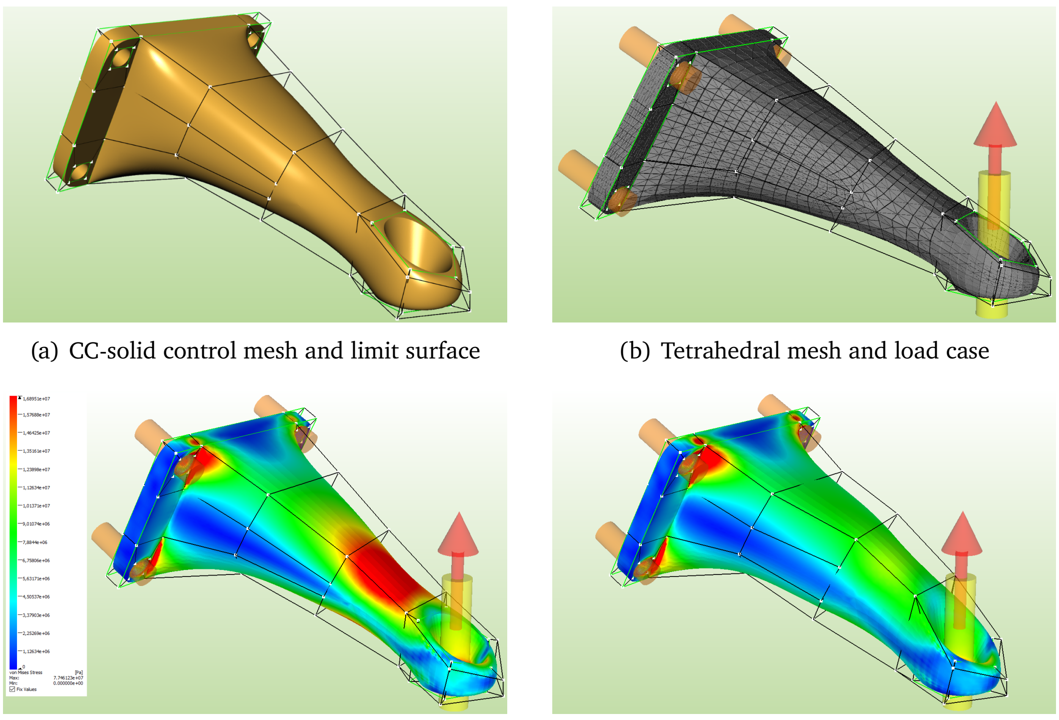
To avoid self-intersections while modeling CC-solid models, the implemented modeling system detects and rejects self-intersections for the outer surface of the control mesh. However, this may still lead to intersections inside the control mesh, e.g. when adding small details to a subdivided mesh after having defined its overall shape on a coarse level. In these cases, the mass-spring-based approach presented in Section 3.3.5 as well as Laplacian smoothing of inner control points (see Section 4.3) can be applied to resolve the intersections. As an example shown in Figure 6.2(b), the coyote model has been subdivided twice and enriched with details on its ears, shoulders, back and tail, resulting in self-intersections that are resolved using Laplacian smoothing.

6.1.2. Finite Element Simulation

As a smooth workflow is essential for a tight integration between modeling and simulation, the system uses a fast GPU-based finite element solver based on the methods by Weber et al. [Web+13] to simulate the structural behavior of the 3D object. The solver requires a tetrahedral mesh as input. It is able to run the simulation with tetrahedral elements with linear or higher-order basis functions, as presented by Weber et al. [Web+15]. For converting the CC-solid model into a tetrahedral mesh, subdivision-based tetrahedral mesh generation is performed as described in Chapter 4. The simulation itself can be run either as a static or a dynamic simulation. While static simulations try to find an equilibrium between internal elastic forces and external forces (boundary conditions), dynamic simulations are time dependent and can deal with dynamically changing boundary conditions as well. To control the simulation parameters, material properties can be defined in terms of Young's modulus and Poisson's ratio for the entire model or per tetrahedron. Boundary conditions such as fixation and external forces can be defined per node of the simulation mesh. The solver provides the deformation for each node and the computed stress tensors for each element as its results.

6.1.3. Interactive Visual Feedback

The relation between the subdivision mesh and the tetrahedral mesh allows for a volumetric visualization of the simulation result (e.g. the *von Mises* stress or the absolute displacement) directly on the CC-solid model. The displacement is shown geometrically on the nodes whereas the stress is visualized using an interactive color mapping as shown in Figures 6.3(c) and (d). Although the visualization is just an approximation of the exact values, it helps to identify the weak points of the 3D object and to improve the design.



(a) CC-solid control mesh and limit surface

(b) Tetrahedral mesh and load case

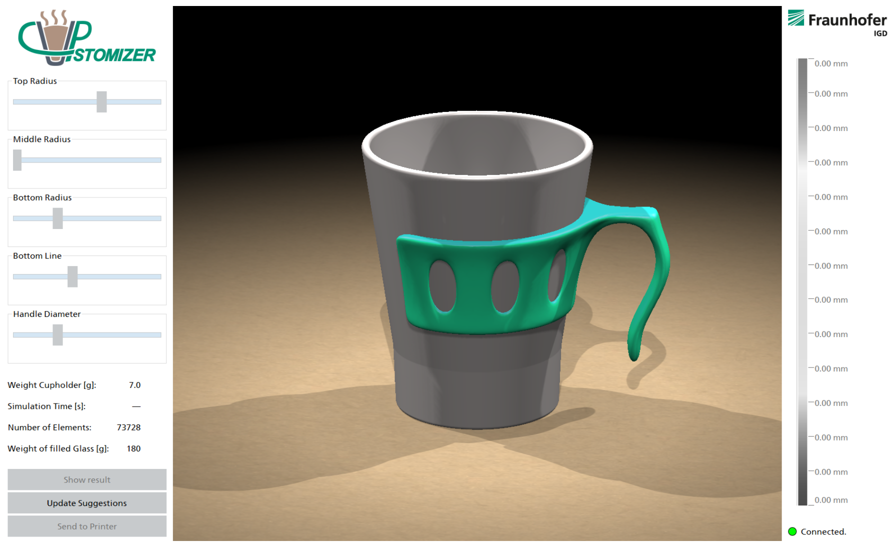
(c) Simulation results for the model shown in (a)

(d) Simulation results after modifiying the model

Figure 6.3.: The engine mount model (a), which has been created from scratch with the interactive modeling environment, is converted into a tetrahedral mesh (b) and simulated (c). To strengthen the front of the object, several control points are modified interactively. Simulating the modified engine mount shows a reduction of the von Mises stress in that area (d). The same color mapping is used in (c) and (d).

Figure 6.3 shows how the system can be used. For this specific example, the CC-solid control mesh consists of 120 control points and 43 hexahedral cells. It is converted into a tetrahedral mesh with 133632 tetrahedra using the Type-9 partitioning with two initial subdivision steps (see Section 4.1.1). While the generation of the tetrahedral mesh and the corresponding matrices took 1419 milliseconds on an Intel Core i5 4570 CPU, updating the tetrahedral mesh after modifying the control points just took 2 ms. On an NVIDIA GeForce GTX 970 GPU, performing the simulation took 139 ms.

6.2. Interactive Design Space Exploration for Mass-Customized 3D-Printed Objects



Weight Cupholder [g]: 7.0
Simulation Time [s]: —
Number of Elements: 73728
Weight of filled Glass [g]: 180

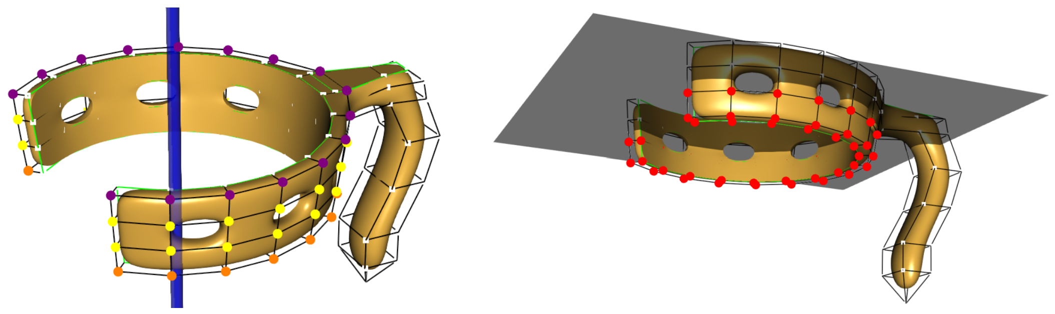
Figure 6.4.: Interactive exploration of a given design space using a CC-solid model with five parameters, linked with a finite element simulation based on tetrahedra.

Mass customization and 3D printing services on the internet are a recent trend. 3D printing allows for custom one-off product manufacturing at a reasonable price. Typically, 3D printing online shops and service providers such as Thingiverse [Thi20], Shapeways [SHA19], or Nervous System [Ner19] only check whether a product can be printed by analyzing the geometry of the model. While these services allow for geometry variations and are performing geometry-based checks before printing, they do not cover physically based requirements resulting from real-world use.

The application presented in this section is a show case for integrating interactive design and simulation for customizing parametrized 3D models. The application enables a user to vary design parameters and guides him/her towards parameters that satisfy a given objective function. In a manufacturing scenario, such an objective function could be to maintain stability of the part while adding as little material as possible. As in Section 6.1, CC solids are used together with the efficient tetrahedral mesh generation method presented in Chapter 4 and the GPU-based FEM solver. Instead of manipulating the CC-solid model on control point level, a set of parameters is defined using the concepts presented in Section 3.3.4. Starting from the initial geometric configuration, the user manipulates the shape of the 3D object by adjusting the parameter values. The smooth transition between design and simulation, as well as the speed of the GPU-solver, render the system interactive, requiring no precomputation.

The efficient mesh generation and the speed of the GPU-based solver allow for not only performing stability analysis but also sensitivity analysis. Additional simulations are being run with varied parameter values to give suggestions on which parameter to tune first, in case the chosen configuration does not yield a stable model. These suggestions can be taken-up to optimize the design and find a best possible compromise between the desired custom shape and the stability requirements to be fulfilled.

As a consistent example throughout this section, I use a customizable cup holder with five parameters, i.e. three different radii, the overall height of the cup holder and the thickness of its handle (see Figure 6.5). The load case for the simulation (see Figure 6.6(a)) resembles the cup holder being held at its handle, carrying the weight of a filled cup.


(a) Three axis parameters to modify the thickness of the cup holder

(b) Plane parameter to modifiy the height of the cup holder

Figure 6.5.: The cup holder model and its parameters. (a) shows three groups of control points marked in purple, yellow, and orange. Each group is scaled relative to the z axis (shown in blue) to modify the outer radius of the holder at its top, middle section and bottom, respectively. (b) shows the group of control points (red) that is scaled relative to a plane (shown in dark gray) to modify the overall height of the cup holder.

6.2.1. Customizing Parametrized CC-Solid Models

Constrained parametric design is one of the most common object representations suited for customization. It allows capturing the design intent using features and constraints, enabling users to easily perform changes. For example, organizations often turn to parametric modeling when making families of products that include slight variations of a core design.

The cup holder example defines the following five parameters:

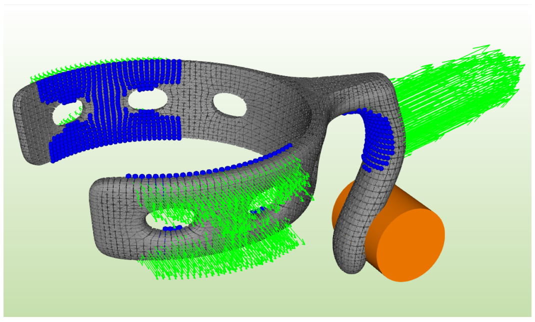
a) The outer radius at the top of the cup holder
b) The outer radius of the middle section
c) The outer radius at the bottom
d) The overall height of the cup holder
e) The thickness of the handle

Following the concepts presented in Section 3.3.4, each parameter applies one or multiple parametric operations to the CC-solid control mesh. While the first three parameters use scaling operations around an axis (see Figure 6.5(a)), the fourth parameter is realized with scaling relative to a plane (see Figure 6.5(b)). The fifth parameter combines multiple axis operations and plane operations in order to follow the complex geometry of the handle. In the application's user interface, each parameter is represented by one slider that allows changing the parameter value interactively between the parameter's minimum and maximum allowed value (see Figure 6.4). The influence of the parameters on the control points results in an interactive geometry modification without geometry caching.

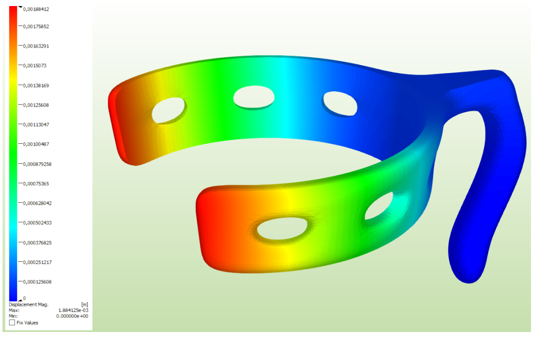
6.2.2. Finite Element Simulation

In order to evaluate the validity of the customized model by its physical properties, the system performs a structural analysis whenever a parameter value is changed by the user. The load case represents the gravitational force of the filled cup and the forces applied by the human hand to the handle (see Figure 6.6(a)). For the example of the "c"-shaped cup holder, the holder will more likely bend and release the cup than that it will break. Therefore, the validity of the custom design depends on the maximum displacement computed by the simulation.

The transition between modeling and simulation as well as the performance of the simulation is crucial to the user experience. Therefore, the tetrahedral mesh is generated from the CC-solid model using the efficient mesh generation approach presented in Chapter 4.



(a) Tetrahedral mesh and load case



(b) Simulation results (displacement)

Figure 6.6.: The tetrahedral mesh generated from the CC-solid model (a) and the color-coded simulation results (b). (a) also shows the boundary conditions of the load case, i.e. the fixation (orange cylinder) and external forces (green arrows). The tetrahedral mesh nodes which are affected by the external forces are highlighted in blue.

The simulation is performed on tetrahedral elements with linear basis functions, using the same GPU-based solver as in Section 6.1.2. The material parameters are adjusted to the material used in the 3D printing process (e.g. polylactic acid – PLA).

Figure 6.6 shows the tetrahedral mesh with its load case, as well as the color-coded visualization of the displacement. The mesh consists of 146432 tetrahedra. The simulation was performed in 823 milliseconds on an NVIDIA GeForce GTX 970 GPU.

6.2.3. Guided Exploration of the Design Space

In order to provide hints to the user on which parameter he or she should modify in order to improve the current design, an objective function is defined that aims for minimal weight while ensuring enough stability for holding the filled cup. The system calculates the partial derivative $|\Delta_i f(\mathbf{p})|$ of each parameter i w.r.t the objective function f using one-sided finite differences. The parameter with the largest $|\Delta_i f(\mathbf{p})|$ has the highest impact on improving f and therefore the current design. The sign of $\Delta_i f(\mathbf{p})$ indicates the direction in which the parameter shall be changed.

The used techniques provide smooth transitions between the individual stages – modeling, simulation, and adaptation – in an interactive user experience. Figure 6.7 shows five differently customized cup holders created with the application. Before having been printed, all five successfully passed the validation via the FE simulation. The printed objects have proven to be able to hold a full cup.

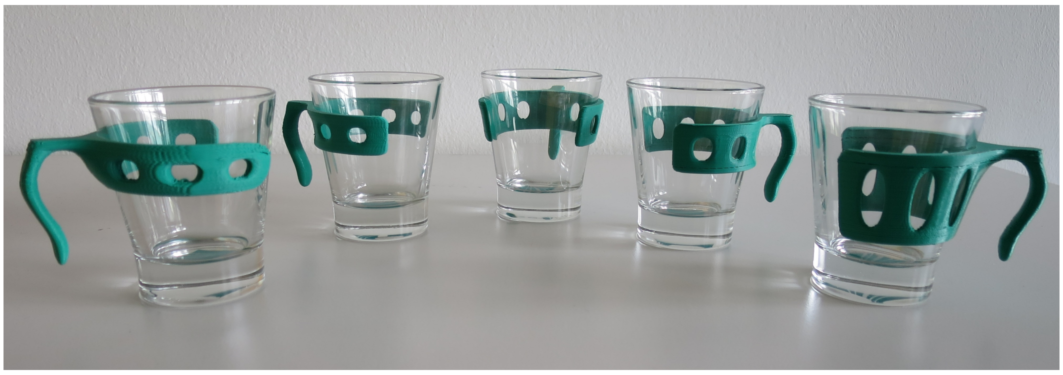


Figure 6.7.: Print-outs of five different cup holders, customized and simulated with the application presented here. All five passed the validation.

6.3. Modeling of Smoothly Graded Volumetric Material Properties for Multi-Material Additive Manufacturing

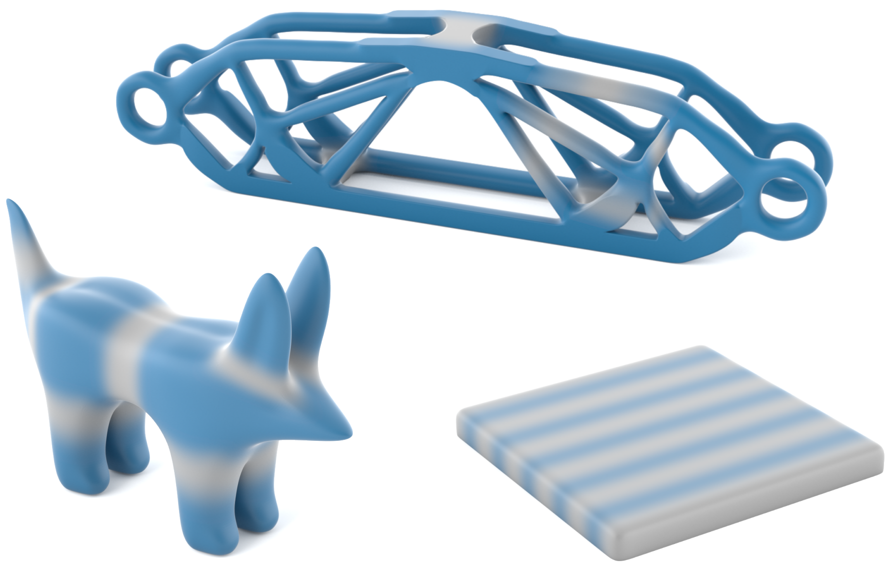


Figure 6.8.: Multi-material 3D printable models with smooth material gradation generated with the techniques presented in this section. Areas with a high stiffness are shown in blue, while flexible regions are shown in white.

Modern 3D printers can vary mechanical properties of objects locally, e.g. by depositing different base materials or varying process parameters – the latter is less common, still. Traditionally, CAD systems based on B-Reps only support assigning one material for each solid or topological lump. However, printers, such as the Stratasys J750, allow for material variation on a per-voxel basis. Obviously, defining a lump per voxel would be an impractical approach considering the size of the individual elements being printed.

The application presented in this section uses Catmull-Clark solids to represent the shape and the material property distribution for 3D-printed objects. The approach offers great flexibility in geometric modeling and property definition with a relatively low number of degrees of freedom. Applying Luu et al.'s direct slicing approach [Luu+19] to the CC-solid models allows for accurate voxelization and printing of the multi-material objects.

6.3.1. Material Definition on Volumetric Subdivision Models

For modeling continuous volumetric material properties, the application presented here implements the concept for encoding additional properties on CC-solid control points as described in Section 3.1. This way, models with e.g. volumetrically varying stiffness can be created and very interesting physical behaviors can be achieved.

The application extends the interactive modeling system presented in Section 6.1.1 with functionality for applying material properties on the individual CC-solid control points. This allows for defining smooth variations of material properties. However, even though CC-solid subdivision supports crease edges and sharp features for the geometry, discrete boundaries and discontinuous transitions are not yet supported for material properties.

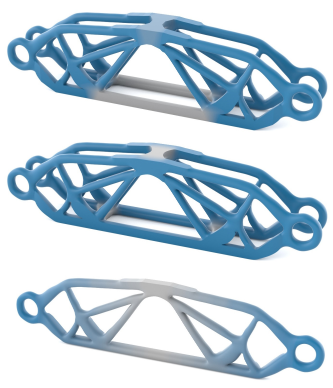
6.3.2. Slicing and 3D Printing of Volumetric Multi-Material Models

To print the 3D objects with a 3D printer that supports multiple materials, the objects have to be converted into the respective input format for that particular printer. In case of the Stratasys J750 printer that was used for the experiments shown in this section, the input format is a stack of 2D images that determines, which material is placed at which position in a three-dimensional voxel grid. Every image represents one layer that is finally 3D printed. Creating the layer information from the input 3D object is called “slicing”.

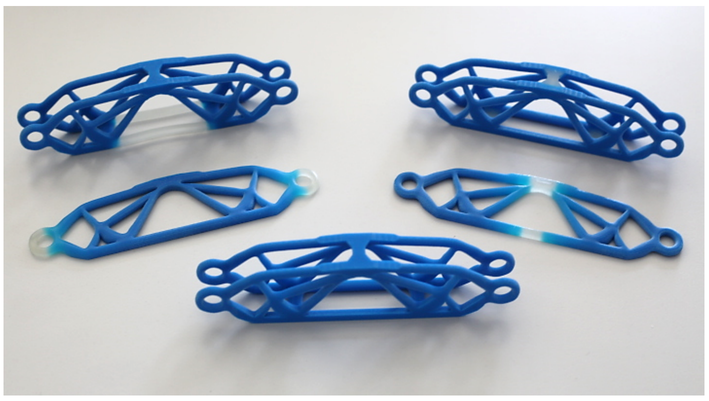
The multi-material CC-solid models are sliced at printer resolution using the direct and efficient slicing approach presented by Luu et al. [Luu+19], avoiding any prior approximation error. The calculated property values – each resembling a mixture of materials – are then halftoned via a 3D blue noise mask, which results in a concrete assignment of one material per voxel. This halftoning is performed by the software Cuttlefish [IGD19].

For printing the example models, two different photopolymer materials were used on a Stratasys J750 printer. A cyan *Vero* photopolymer was chosen as a stiff material. *Agilus30* was chosen as a flexible material. Figure 6.9 shows some of the printed multi-material parts alongside their digital counterparts.

However, the described pipeline of assigning material property values and printing multi-material objects is missing a calibration step for correctly converting the modeled stiffness values into a material distribution. One possible approach to obtain a calibration is to print samples with different material mixtures and measure their stiffness. Interpolating between the samples will give material distributions for stiffnesses that can be achieved with the available materials.



(a) Digital models



(b) 3D printed parts

Figure 6.9.: Multiple example models created with the approach (a), as well as the corresponding printouts with the Stratasys J750 (b). Blue areas indicate a high stiffness, while white areas indicate a low one.

6.4. CC-Solid-Based Internal Structures for AM Parts

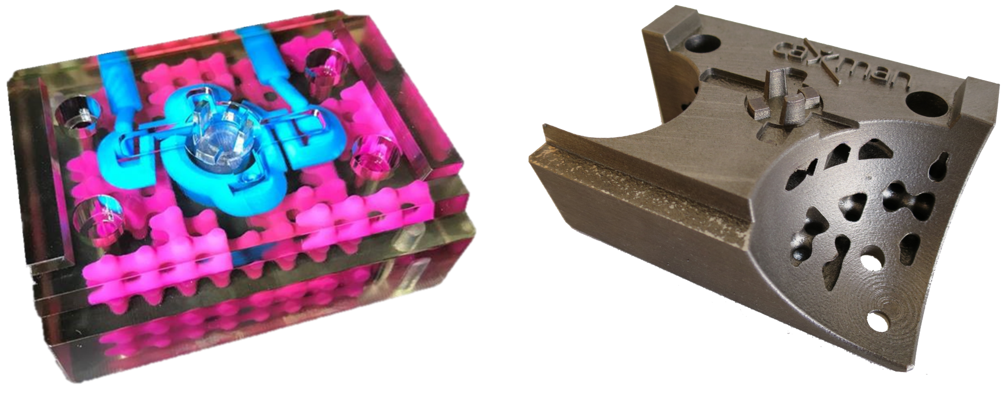
Additive manufacturing enables the production of parts with internal structures that cannot be created with other manufacturing technologies, such as turning, milling, and casting. The most obvious motivation of integrating inner structures into a part design is to save material and manufacturing costs by reducing the weight, as well as the printing time of such parts, while still maintaining the required stability. Modern 3D printing software such as Cura [BV20] or Simplify3D [Inc20c] offers hollowing the object to be printed with so-called *infill* – repeated inner structures oriented in printing direction that are added in the slicing process. When working with powder-based or resin-based printing techniques, unused material remains inside the hollowed object, unless outlets are designed explicitly or holes are drilled into the physical part afterwards.

However, internal structures provide more advantages than just reducing the amount of material. Based on the chosen AM technology, repeated heating and cooling of every layer induces thermal stresses resulting in distortions in the printed part. By decreasing the total area per layer, inner structures decrease heat-induced warping and bending as examined by Tamellini et al. [Tam+20]. This leads to a lower amount of support structures required for dissipating heat and for counteracting thermal stress, as well as to a higher geometrical precision of the printed part. Finally, as multi-material 3D printing is almost non-existing for metal printing or for powder-bed techniques in general, inner structures can be used to change the overall physical behavior of the part by emulating material properties different from those of the actual printing material. Such properties include anisotropic, non-linear, and discontinuous stiffnesses, auxetic behavior and more.

In contrast to infill defined in the slicer, explicitly defined structures can be integrated into the original 3D object for further processing in a CAD system – in most cases the same CAD system that was used to create the 3D model in the first place. The structures can also be integrated when analyzing the object in a given simulation environment. Only this way, the mechanical behavior of the printed part can be predicted properly and the advantages of inner structures can be fully exploited. However, defining internal structures without following the printing direction might require additional support material. Support material around such structures will be difficult or even impossible to remove.

The techniques presented in this section use Catmull-Clark solids as the geometric representation for defining internal structures for AM parts. Inherently, CC solids create smooth, organic shapes with a watertight surface. The subdivision basis functions transform sharp angles in the control mesh into smooth transitions, similar to inserting fillets in a CAD program. Depending on the 3D printing technology to be used, this reduces the amount of required support material and increases the surface quality, as it turns downfacing surfaces into self-supporting arches [Ltd19]. Instead of modeling the actual geometry of the structures with CC solids, the subdivision geometry can also be used to model empty regions inside the original 3D object, as well as for modeling regions with different material properties – constant or volumetrically graded – than the rest of the part (see Section 3.1). Applying the techniques presented in Section 3.3.4, the structures can be controlled by a set of geometric parameters, e.g. for shape optimization. This versatility enables a variety of applications for CC solids in AM part design.

The volumetric representation is needed only when defining regions with volumetrically graded material properties. For all other cases, the limit surface is sufficient. However, CC solids inherently create watertight limit surfaces. Figure 6.10 shows CC-solid structures inserted into a part of an injection mold. In this specific case, the CC-solid geometry was used to represent the empty regions of the part. The final model was created by subtracting the CC-solid limit surface from the B-Rep model of the injection mold with a Boolean operation.



(a) Multi-color print-out (b) Print-out in metal

Figure 6.10.: One half of an injection mold, enhanced with subdivision-based internal structures. (a) shows a multi-material print-out with the mold's cooling system colored in blue and the CC-solid structures colored in pink. In the production part, the cooling channels as well as the subdivision structures will be void – no material will be deposited there. (b) shows a metal-printed version of the same mold with two cut-outs for better visibility of the inner structures.

In the following, I present two different approaches for creating CC-solid geometry inside a given B-Rep model:

1. Interactive modeling
2. Procedural generation

Both approaches can be used for explicitly creating structures, as well as for modeling void or multi-material regions. For both approaches, an offset surface of the original 3D model can be created for ensuring a minimum wall thickness in the hollowed part. Interactive, as well as automatic placement of outlets for removing unused material allows both approaches to adapt to the AM process of choice.

6.4.1. Interactive Modeling of Inner Structures

The first approach for creating internal structures extends the interactive CC-solid modeling system presented in Section 6.1.1. The user interactively creates CC-solid geometry inside a given 3D object, e.g. a CAD model in B-Rep. For better orientation inside the object,

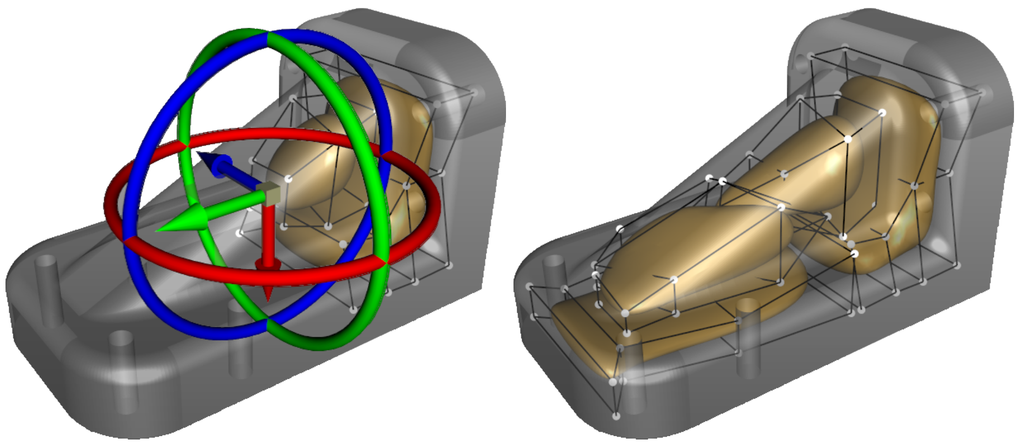

(a) Interactive modeling (b) Resulting cavities

Figure 6.11.: Interactive modeling of cavities inside a CAD model of a bracket. (a) pictures the interactive modeling process showing the 3D widget to manipulate vertices, edges, faces, and cells of the cavities. (b) shows the final model with four independent cavities inside.

its outer shell is rendered transparently. Additionally, intersections between the CC-solid limit surface and the outer surface of the original object are detected and visualized. Figure 6.11 shows an example of interactively designing several cavities inside a bracket.

Depending on the additive manufacturing process used to print the part, additional design constraints must be met. When printing a hollowed part with powder-based or resin-based methods, outlets have to be created to prevent unused material being trapped inside the object. The interactive modeling system offers a two-step process. First, an automatic algorithm connects all individual cavities via internal channels to topologically create a single cavity inside the object as shown in Figure 6.12(a). In the second step, the user interactively creates outlets for material removal by selecting the desired positions on the part's surface. The system inserts additional CC-solid cells that intersect the outer surface at that position. The outlet is oriented according to the surface normal at that point. The system automatically creates a channel from the outlet to the closest cavity inside the object as shown in Figure 6.12(b). The diameter of each channel is user-defined and can be adapted to the requirements of the printing process.

Figure 6.13 shows a demo print-out of the bracket including the cavities designed with the interactive modeling approach as described above. As a showcase, the bracket has been printed in two parts. Note that no channels are required, since the part was printed with *fused filament fabrication* (FFF). The cavities have also been printed for demonstration purposes, using a *stereolithography* printer.

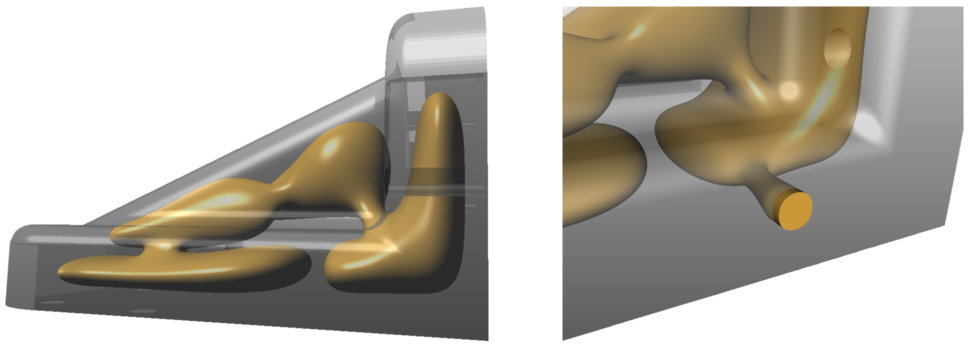

(a) Cavities connected by channels (b) Interactively placed outlet

Figure 6.12.: Automatic connection of individual cavities with internal channels to allow flow of unused material (a) and interactive creation of material outlets by intersecting the outer surface of the part with subdivision geometry (b).

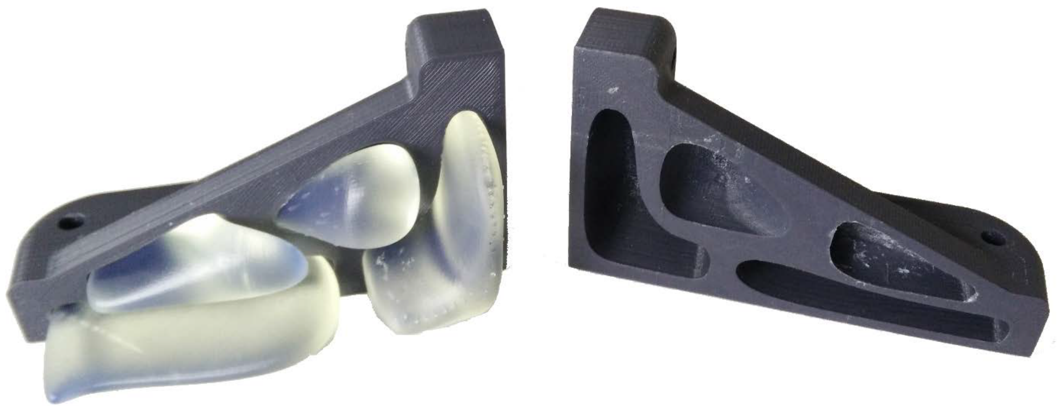


Figure 6.13.: Demo print-out of the bracket including four interactively designed subdivision-based cavities.

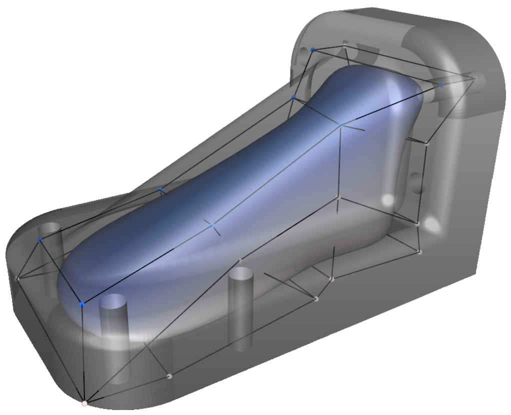


Figure 6.14.: Bracket with a region of smoothly graded density inside, defined by CC-solid geometry. Different density values are assigned to the CC-solid control points using the techniques presented in Section 3.1 Blue color represents a high density while regions with low density are shown in white.

Assigning varying material properties such as stiffness or density/porosity at the CC-solid control points as shown in Section 3.1 allows for creating regions with varying material properties inside given B-Rep models. Figure 6.14 shows CC-solid geometry with graded density inside the bracket providing high density in the diagonal part while reducing the density and thereby saving material at the bottom and at the back of the bracket.

6.4.2. Procedural Inner Structures

The second approach for creating inner structures is procedural generation. Procedural structures can be created by programmatically repeating patterns of CC-solid geometry in a grid inside the object. The individual patterns can be defined either by explicitly modeling a reference instance or by a rule set that creates CC-solid cells for individual grid cells based on their three-dimensional index in the grid. Different patterns hold different physical properties in terms of stability, heat dissipation, and material reduction. Figure 6.15 shows five example patterns that are repeated in a regular grid inside a cube. Figure 6.10 shows procedural structures inside the mold, created by repeating one of these patterns in a regular grid. The individual arches of the structures span a distance of about $2.8mm$ and could therefore be printed without additional support.

Alternating patterns can be applied to different regions of the given part to better react on different stress levels or load cases. For the same reason, the grids can be distorted interactively or algorithmically, e.g. by applying the parameters presented in Section 3.3.4. Figure 6.16 shows the creation of procedural structures inside the bracket, based on a warped grid and the pattern shown in the bottom left of Figure 6.15.

Instead of repeating patterns, procedural internal structures can also be generated from mathematical functions describing the shape of the structures. Parameters of these functions can be altered continuously or discretely, resulting in (smoothly) varying structures throughout the model. Figure 6.17 shows different structures with the shape of sine curves. The structures show the results of varying amplitudes, frequency and phase offsets of the sine functions. Such geometry can be used for modifying the stiffness inside the part and for inducing a dampening behavior.

As for the interactive modeling of structures, the procedurally generated subdivision geometry can either represent the structures themselves, or the regions inside the 3D object that are not be filled with material (void) or that are filled with a different material, respectively.

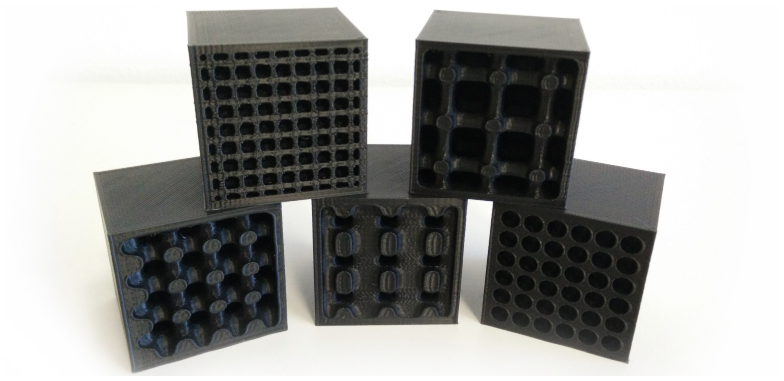
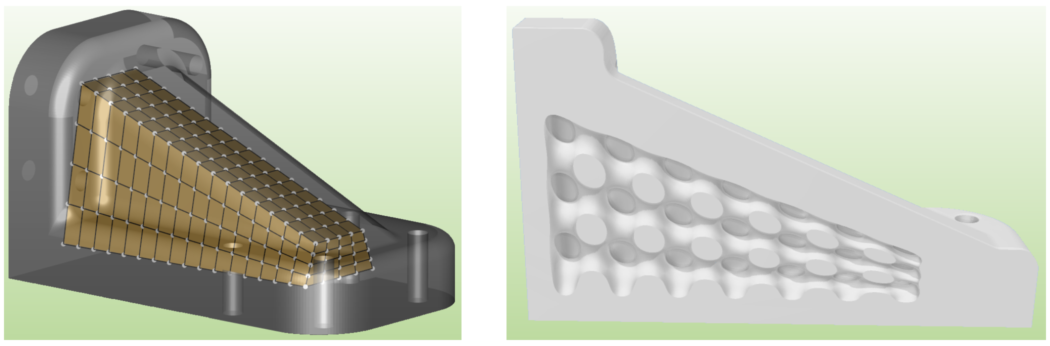


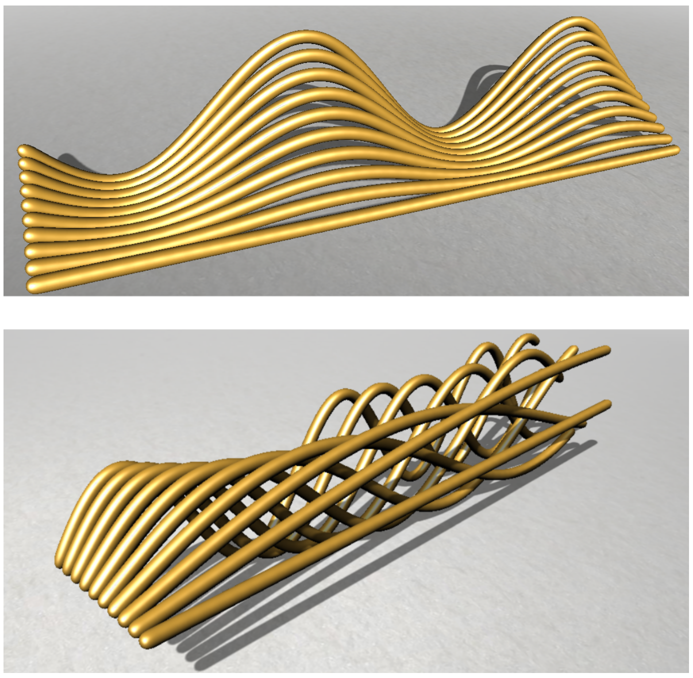
Figure 6.15.: Different CC-solid structures inserted into several cubes. The structures were created by repeating different patterns of CC-solid cells in a regular grid. The parts were printed with an FFF printer.



(a) Warped grid inside the bracket model

(b) Resulting inner structures (cut)

Figure 6.16.: Creating internal structures procedurally from a warped grid (a) and a set of rules that define in which grid cells to create the CC-solid geometry. (b) shows a cut through the bracket model, revealing the resulting structures.


(a) Sine curves with increasing amplitude and frequency

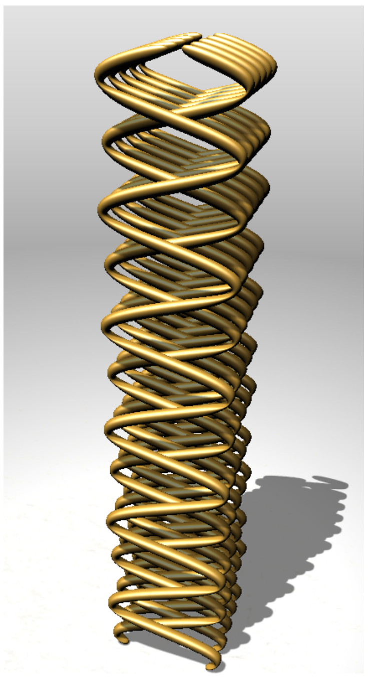

(b) Interleaved sine curves with alternating phase offsets

Figure 6.17.: Procedural CC-solid structures that can be embedded into a given 3D object. (a) shows structures created from multiple sine curves with increasing amplitude (top) as well as from curves with increasing frequency (bottom). The structure shown in (b) is created from sine curves with alternating phase offsets and can provide a spring-like or dampening effect.

6.4.3. Creating the Combined Model

For integrating the CC-solid structures into the original CAD model, the subdivision geometry has to be converted into a representation that is compatible with common CAD tools. As the limit surface of CC solids is equivalent to a standard Catmull-Clark limit surface, it can be converted into a set of bicubic B-spline patches e.g. using the algorithm presented by Loop and Schaefer [LS08]. Please note that only regular Catmull-Clark patches can be converted exactly, while irregular patches introduce an approximation error.

The resulting continuous B-Rep can be exported as STEP AP214 [ISO10]. In this form, it can be imported into any CAD system that supports the STEP AP214 standard, e.g. CATIA [Das20b], SolidWorks [Das20a], TopSolid [Sof20], or Rhino [MA18]. The combination of the inner structures with the original B-Rep model is performed inside the CAD system with a single Boolean operation. If the internal structures do not intersect with the surface of the original CAD model, the combination boils down to adding the structures as additional shells. An example using the CAD software *TopSolid* [Sof20] is shown in Figure 6.18.

Please note, that this approach is only applicable, if the volumetric information of the CC-solid structures is not needed, i.e. if the CC solids represent structures with a uniform material/material property or void.

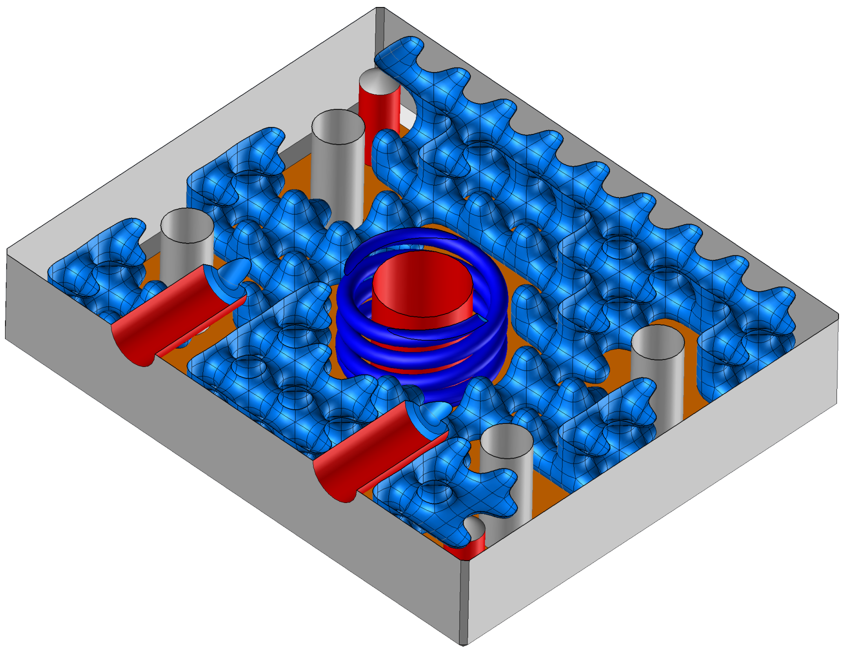


Figure 6.18.: Combination of inner structures with the B-Rep model of the mold. The limit surface of the CC-solid structures was converted into a set of B-spline patches and imported into TopSolid [Sof20]. The combination with the original CAD model of the mold was performed with a Boolean operation inside the CAD system. A cut through the model reveals the structures (shown in light blue).

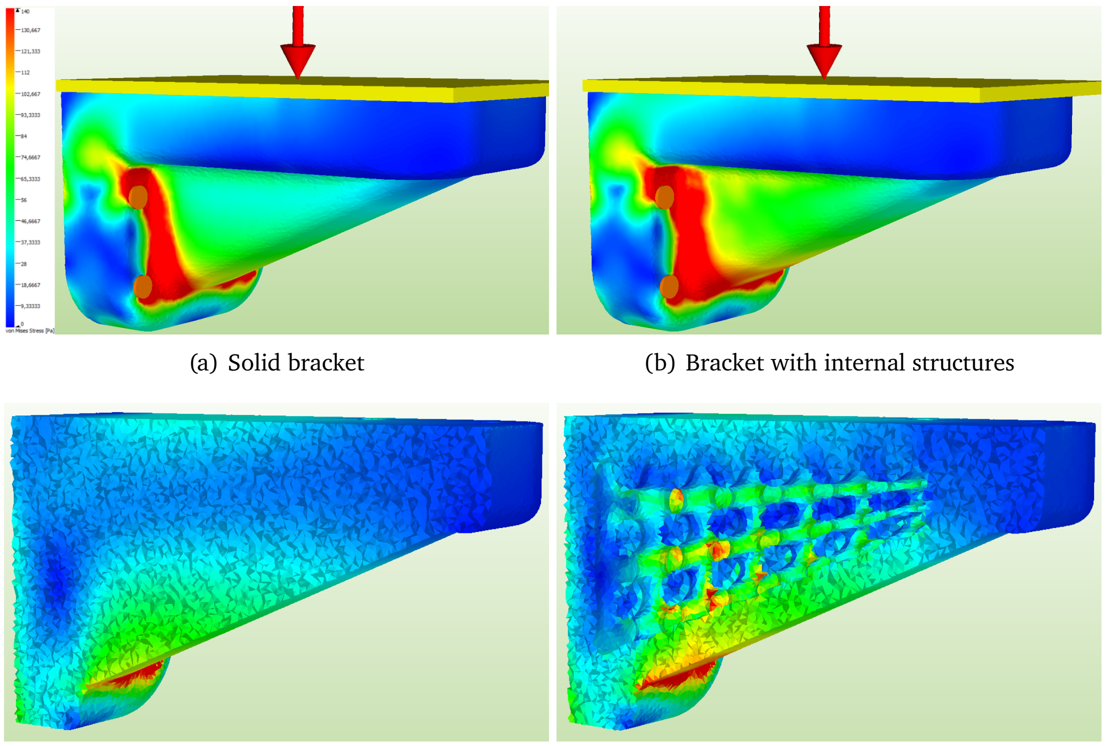
6.4.4. Simulating Parts with Internal Structures

As stated earlier, the structures have to be taken into account when analyzing the part in terms of physically based simulation. This is not possible for infill defined during slicing, as no explicit description of the geometry exists.

After combining the original CAD model with the CC-solid-based inner structures as described above, the resulting B-Rep model can be converted into a tetrahedral mesh using state-of-the-art meshing tools such as CGAL [Jam+15], TetGen [Si15], or Gmsh [GR09]. However, depending on the size of the individual structures, a large number of elements is required for properly representing fine geometric details in the tetrahedral mesh. The resulting mesh can be simulated in the FEA framework of choice.

Figure 6.19 shows a structural analysis of the bracket model using the same FE environment as in Sections 4.3, 6.1, and 6.2. In this example, both meshes consist of around 700k tetrahedra. The simulations solve the linear elasticity problem for the solid bracket as well as for the hollowed model shown in Figure 6.16(b). The bracket is fixed at the four holes on its backside and a vertical force is applied as shown in Figures 6.19(a) and (b). Inserting the structures reduces the volume of the model and therefore the amount of material required for manufacturing it by 13.4%. However, at the same time, the maximum displacement is increased by 15.0% compared to the solid bracket. While the maximum *von Mises* stress decreases by 1.66%, the average stress throughout the model is increased by 28.6%. Inserting the structures results in a more uniform stress distribution as can be seen when comparing Figures 6.19(c) and (d). Hollowing the part without inserting any structures saves 26.2% of the material but increases the maximum displacement by 26.9% and also increases the maximum stress by 11.2%.

Tamellini et al. incorporated the procedural structures into optimizing the thermal situation during the printing process for reduced warping and improved geometric accuracy [Tam+20]. By employing the fast GPU solver used throughout this chapter, my CC-solid-based structures can also be envisaged in an efficient shape optimization scenario based on structural analysis.



(a) Solid bracket

(b) Bracket with internal structures

(c) Cut through (a)

(d) Cut through (b)

Figure 6.19.: Von Mises stress distribution for the solid bracket (a, c) and the bracket with internal structures (b, d). (a) and (b) also show the load case with fixations shown in orange, forced areas shown in yellow, and the force directions being indicated by a red arrow. The stress is color-coded from blue (low) to red (high). All four images use the same color mapping.

7. Conclusion and Future Work

Nowadays, there is still a large gap in the engineering workflow between geometric design and physically based simulation. This is due to the incompatibility of the different representations commonly used for the two steps – continuous B-Rep models vs. discrete volumetric meshes. Following the idea of Isogeometric Analysis, I investigated subdivision volumes, i.e. Catmull-Clark solids, as a continuous volumetric representation for efficient integration of geometric modeling and simulation.

Catmull-Clark solids combine the beneficial modeling characteristics known from subdivision surfaces:

- creating smooth geometry, defined by few control points,
- allowing for control meshes with almost arbitrary topology, and
- providing C^2 continuity of the limit in regular regions,

with the volumetric mesh properties needed for physically based simulation:

- having cells and nodes in the interior of the model to represent the physical behavior throughout the solid object, as well as
- providing higher-order basis functions suitable for solving PDEs.

While subdivision surfaces caught a lot of attention in the context of IGA for thin-shell objects, subdivision volumes remained rather unexplored except for the work of Burkhart et al. [BHU10b; Bur11]. For IGA on volumetric representations, trivariate NURBS are still the de facto standard. However, complex topologies are difficult to represent due to the tensor-product structure of trivariate splines. To model complex objects, either trimming or tiling is required, both reducing the continuity to C^0. Subdivision volumes such as Catmull-Clark solids are able to handle extraordinary edges, as well as extraordinary vertices, and can represent complex objects as a single entity – a single subdivision volume. However, these irregularities reduce the continuity to less than C^2.

For finite elements, hexahedral and especially tetrahedral meshes are able to approximate almost any given surface geometry. While linear finite elements can be elevated to higher-order basis functions by introducing additional degrees of freedom, the basis functions of Catmull-Clark solids are inherently cubic. As for trivariate NURBS, basis functions for tetrahedral or hexahedral finite elements are only defined inside the element, resulting in C^0 continuity across element borders.

The cubic basis functions and the C^2 continuity of CC solids can not only be used for geometric modeling and simulation, but also to encode smooth volumetric scalar, vector, or tensor fields, such as varying (anisotropic) material properties. Although tetrahedral and hexahedral meshes as well as trivariate NURBS can carry additional volumetric information, using volumetric Catmull-Clark subdivision models inherently allows representing continuous volumetric gradation, also for complex topologies. As shown in a recent report on multi-material additive manufacturing, graded material transitions greatly improve the stability of the printed parts compared to using discontinuous changes in material [Aer18].

Apart from the benefits listed above, using CC solids for integrating geometric modeling and simulation introduces the following challenges that I tackled in this dissertation:

a) Volumetric limit evaluation of CC-solid models is considerably more complex than evaluating the limit surface of a CC-surface model due to the drastically increased number of irregular topological configurations. At the same time, an efficient limit evaluation technique is crucial for the performance of the simulation, as well as for applications such as slicing for multi-material additive manufacturing.
b) Modeling volumetric control meshes is much less explored than modeling surface meshes. While polygonal control meshes for CC surfaces can be modeled with several commonly used 3D modeling tools such as Blender [Fou20] or Maya [Aut18], to the best of my knowledge, no volumetric modeling tools for CC-solid control meshes have been presented prior to this dissertation. Additionally, maintaining the consistency of the volumetric mesh during interactive modeling is much more difficult due to the higher topological complexity compared to surface meshes.
c) Simulations on CC solids require specialized algorithms to set up and solve the simulation problem and to visualize the simulation results. The FE community provides a wide range of well-developed and mature tools for pre-processing, performing and post-processing FE simulations. However, those tools are not compatible with the CC-solid representation.
d) For performing physically based simulation, numerical integration is required for assembling the system of linear equations, i.e. for the computation of element stiffnesses and external forces. While efficient and exact integration is crucial for

the performance of the simulation and the accuracy of its results, integrating over CC solids – especially over irregular elements – with standard quadrature methods as proposed by Burkhart et al. [BHU10b] produces erroneous results.

e) For FEA as well as for IGA, the quality of the simulation mesh has great impact on the solvability of the linear system and on the accuracy of the simulation results. For tetrahedral finite element meshes, geometric measures such as the ones presented by Shewchuk [She02] and by Field [Fie00] are suitable for determining the quality of the mesh. However, geometrically well-shaped CC-solid control meshes lead to ill-conditioned or unsolvable linear systems if the mapping from the control mesh to the limit volume contains distortions, singularities, or inversions. This issue has not been investigated by Burkhart et al. [BHU10b].

According to the five research questions **RQ1** to **RQ5**, derived from these challenges, I presented the following five contributions **C1** to **C5** in this dissertation:

RQ1: How can limit evaluation for CC solids be performed more efficiently?

Based on the original CC-solid subdivision rules by Joy and MacCracken [JM99], I developed a modified vertex rule. Combining it with Burkhart et al.'s modified edge rule [BHU10b], I achieved a tensor-product property for the CC-solid basis functions also around extraordinary edges and vertices. Exploiting this tensor-product property, I presented the first constant-time limit evaluation method for CC solids (**C1**), including irregular configurations with one or more EEs, as well as boundary cells (see Section 3.2). This approach is faster than Burkhart et al.'s [BHU10b] subdivision-based evaluation approach for every topology and every parameter point (u, v, w) that would otherwise require more than two local subdivision steps. Constant-time limit evaluation increases the efficiency of IGA based on CC solids, but is also applied highly beneficially in the field of multi-material AM as shown by Luu et al. [Luu+19].

RQ2: How can volumetric control meshes be created and modified?

Based on a half-face data structure similar to OpenVolumeMesh (OVM) [KBK13], I presented volumetric modeling operations for CC-solid models (**C2**, see Section 3.3). The data structure is between $1.7\times$ and $2.8\times$ faster on average when querying topological relations frequently needed for volumetric modeling and CC-solid subdivision, but requires about $12.7\times$ as much memory as OMV. The volumetric modeling operations are built from a set of volumetric Euler operators allowing the creation and modification of volumetric control meshes while ensuring consistent topologies. Furthermore, I presented an approach for enhancing volumetric control meshes with parameters that can be used for shape optimization or interactive exploration of design spaces. Finally, I showed how to algorithmically reduce self-intersections in the volumetric control mesh when modifying just the outer surface – a problem that arises for subdivision solids and does not exist for surface models.

RQ3: How can the volumetric nature of CC solids be exploited when generating tetrahedral meshes?

In order to create a link to the traditional FEA community, I presented an efficient tetrahedral mesh generation approach based on the inherent volumetric structure of CC solids (**C3**, see Chapter 4). Given a CC-solid model, my mesh generation approach is up to $30\times$ faster than creating the tetrahedral mesh just from the outer surface, e.g. by using the *CGAL* library [Jam+15]. When modifying the geometry of the control mesh while keeping the topology unchanged, the method adapts the tetrahedral mesh almost instantly (e.g. in 20 milliseconds for 1.57 million tetrahedra). However, the quality of the resulting tetrahedral mesh depends on the quality of the CC-solid control mesh. As the quality of the meshes produced by tetrahedral meshing algorithms that start from a surface model does not depend on the quality of the input model, these approaches generally produce higher-quality meshes. The lower mesh quality of my approach increases the computation time for solving the PDEs, which is compensated by the short runtime of my fast mesh generation method. Nonetheless, the tetrahedral meshes generate comparable simulation results.

RQ4: How can numerical integration be improved for irregular CC-solid cells?

I presented a hierarchical quadrature for accurately integrating over irregular Catmull-Clark surface patches as well as over irregular CC-solid cells (**C4**, see Section 5.1). The quadrature scheme follows the same concept of local refinement as my constant-time limit evaluation and utilizes the eigenstructures of the local subdivision matrices to evaluate the quadrature points in constant time. For computing the limit surface area and the element stiffnesses, the integration reduces the deviations to corresponding reference solutions by up to three orders of magnitude in comparison to using standard Gauss-Legendre quadrature as proposed by Burkhart et al. [BHU10b]. However, in my experiments, the difference in the simulation results between the two quadrature schemes is only around 0.15% due to the relatively small number of irregular cells in common CC-solid meshes.

RQ5: How can parameterization problems be identified, quantified, and avoided if possible?

I presented a parametrization quality metric – called *mesh parametrization quality* (MPQ) – that reveals areas with a distorted parametrization as well as vanishing derivatives and singularities (**C5**, see Section 5.2). All three phenomena lead to numerical instabilities and a bad conditioning of the system matrix when performing IGA on CC solids. I presented split operations for improving the parametrization quality at the boundary of the CC-solid models. Splitting boundary cells resolves the singularities that are inherently induced by the subdivision rules and thus improves the average MPQ by $25.5\times$ to $360\times$. The median MPQ value is worsened by $1.25\times$ to $1.59\times$ compared to the original boundary cells, meaning that the parametrization quality away from the singularities has decreased slightly. In most cases, those regions already have good MPQ values. Even though the splitting operations introduce EEs and EVs, the simulation results show better convergence due to the improved MPQ.

Four applications demonstrate the practical value of my research results in different domains and scenarios (see Chapter 6):

1. Interactive integrated modeling and simulation
2. Interactive simulation-guided design space exploration
3. Modeling of volumetrically graded material properties for multi-material AM parts
4. Generation of internal structures and cavities for additively manufactured objects

In summary, my dissertation presents decisive contributions that exploit and extend Catmull-Clark solids for a tighter integration of geometric design and physically based simulation, narrowing the gap between the two disciplines in the engineering workflow. On the one hand, I show how to efficiently couple geometric concepts of CC solids with the well-researched and mature methods of FEA. On the other hand, I present techniques for improving IGA on CC solids beyond Burkhart et al.'s pioneering work in 2010. Providing volumetric modeling methods for the design of CC-solid objects as well as presenting two different ways for performing analysis on these models (FEA and IGA), my research clearly demonstrates the potentials of using volumetric subdivision for efficiently integrating modeling and simulation.

However, this process of integration is far from complete. While subdivision surfaces are the state-of-the-art representation for continuous models in the computer graphics and animation community, subdivision methods lack acceptance in engineering. Besides historical reasons, a major drawback is that CC surfaces and solids have to approximate analytic shapes, such as spheres, cylinders or cones, which can be represented exactly using NURBS. Yet, in recent years, subdivision has gained increasing attention. More and more CAD systems are adding support for subdivision surfaces – e.g. under the name of *polyNURBS*. This is often motivated by the organic shapes that result from topology optimization for additive manufacturing. The fact that most subdivision schemes can only approximate analytic shapes is secondary in this scenario, as for AM parts, screw holes and threads – functional surfaces in general – have to be machined anyhow.

The increased interest in subdivision technology might also help subdivision solids to gain momentum in the engineering community. With additive manufacturing requiring novel ways of efficiently representing, modeling, and simulating highly complex multi-material objects, there might be a paradigm shift right ahead of us. If so, my research provides crucial building blocks for serving these demands.

Future Work

A major issue regarding the acceptance of (volumetric) subdivision methods in engineering is, that existing CAD model cannot be re-used in a subdivision-based modeling and simulation environment. Up to now, existing B-Rep NURBS models cannot be straightforwardly transformed into CC solids. CC-solid models have to be created from scratch. For IGA on trivariate NURBS, several ways have been discussed for converting a NURBS surface model into a trivariate NURBS volume, e.g. by Al et al. [Al +16] and by Dokken et al. [DSB19]. For CC solids, these methods are not applicable. One possibility for converting B-Rep models into CC solids is to utilize hexahedral meshing algorithms. A hexahedral mesh created from the given surface geometry could serve as a basis for a CC-solid control mesh with some modifications. Since the Catmull-Clark subdivision scheme – for surfaces as well as for solids – is an approximating scheme, the control points resulting from the hexahedral mesh have to be modified in order for the limit surface to approximate the input B-Rep model. This can be done by inverting the limit stencils for every control point and transforming it accordingly. However, modifying just the surface control points will suffice for single-material objects, as B-Rep models don't carry volumetric information. Although hexahedral meshing is a hard-to-solve problem, promising results have been presented in the last years by Liu et al. [Liu+18], Corman and Crane [CC19], and Gao et al. [GSP19]. As tetrahedra as well as prisms/wedges are also valid CC-solid control cells, hex-dominant meshing might be sufficient instead of pure hexahedral meshing. However, pyramids have to be avoided as they are not supported in CC-solid control meshes (see Section 3.1).

Another drawback with volumetric subdivision is that analytic shapes, such as spheres, cylinders and cones, have to be approximated. NURBS – bivariate and trivariate – can represent those shapes exactly. Cashman overcame this issue for subdivision surfaces with his NURBS-compatible subdivision scheme [Cas+09]. Riffnaller-Schiefer [Rif19] showed how to perform IGA on these surfaces, combining the benefits of NURBS and subdivision surfaces for shell simulation. Since IGA on surface models is prone to approximation errors in the geometry, exact representation of the shape is a huge benefit. Although the approaches presented in this dissertation are based on Catmull-Clark solids, my methods can be adapted to other subdivision schemes. Although IGA on solid models – trivariate NURBS as well as CC solids – is more robust towards geometric approximation errors, extending Cashman et al.'s NURBS-compatible subdivision scheme and Riffnaller-Schiefer's techniques from surfaces to solids would allow for an exact geometry representation.

As described in Section 3.1, polyhedra that do not subdivide into hexahedra cannot be used in CC-solid control meshes. Developing a pre-processing step that splits such polyhedra into polyhedra compatible with CC-solid subdivision will increase the versatility in geometric modeling. For pyramids, this problem is called *Schneider's pyramid* [Sch96a] and is also addressed by Verhetsel et al. [VPR19] as part of the EC-funded research project *HEXTREME* [Cou19]. As the solutions presented there introduce many EEs and EVs, they are not directly applicable in the context of CC-solid subdivision, but might serve as the baseline for a more feasible splitting scheme.

The efficient mesh generation method shown in Chapter 4 provides a link between CC-solid modeling and standard tetrahedral FEA. However, approximating an inherently cubic CC-solid model with linear tetrahedral elements will result in low mesh quality in some cases – sometimes even creating inverted tetrahedra. Curved tetrahedra with quadratic or cubic basis functions – also called *Tet10* and *Tet20* elements – are better suited for approximating the (u, v, w) parametrization, providing better mesh quality and avoiding inversions and self-intersections as long as the CC-solid control mesh is inversion-free. While requiring increased computational effort for handling the additional degrees of freedom, creating Tet10 or Tet20 meshes will result in increased numerical stability of the simulation and better solvability of the system.

Although splitting boundary cells as shown in Section 5.2.3 resolves the singularities in the parametrization of CC-solid boundary cells, my IGA approach on Catmull-Clark solids still suffers from parametrization problems around extraordinary edges and vertices. Wawrzinek and Polthier investigated this problem for CC surfaces, proposing an alternative parametrization which is stable around irregularities but time-consuming to compute [WP16]. Improving the parametrization for CC solids would increase the solvability of the linear system when simulating CC-solid models in general, but especially for models with split boundary cells as they contain additional EEs and EVs.

Efficient GPU implementations of my methods were out of scope for this dissertation. Suitable efficient GPU-based solvers (e.g. the one presented by Weber et al. [Web+13]) as well as efficient GPU data structures for volumetric meshes (e.g. the one presented by Mueller-Roemer et al. [MAS17]) already exist in the computer graphics community. While serious performance benefits are expected from the massively parallel execution on the GPU, my algorithms have to be adapted in order to fulfil the conceptual requirements of modern GPU architectures. Especially the constant-time limit evaluation is not directly compatible and would require modifications e.g. in the computation of the eigenstructures.

References

[Aer18] Aerosint. How to make cheap, scalable multi-material 3D printing a reality. Ed. by Medium.com. [Online; posted 30-April-2018]. Apr. 2018. URL: https://medium.com/@aerosint/how-to-make-cheap-scalable-multi-material-3d-printing-a-reality-d298fa1f76da (visited on 04/11/2020).

[Aig+09] Martin Aigner, Christoph Heinrich, Bert Jüttler, Elisabeth Pilgerstorfer, Bernd Simeon, and A-V Vuong. "Swept volume parameterization for isogeometric analysis". In: IMA international conference on mathematics of surfaces. Springer. 2009, pp. 19–44.

[Al +16] H. Al Akhras, T. Elguedj, A. Gravouil, and M. Rochette. "Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models". In: Computer Methods in Applied Mechanics and Engineering 307 (2016), pp. 256–274.

[All+05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. "Variational tetrahedral meshing". In: ACM Transactions on Graphics (TOG). Vol. 24. ACM. 2005, pp. 617–625. DOI: 10.1145/1186822.1073238.

[All+16] Pierre Alliez, Clément Jamin, Laurent Rineau, Stéphane Tayeb, Jane Tournois, and Mariette Yvinec. "3D Mesh Generation". In: CGAL User and Reference Manual. 4.9. CGAL Editorial Board, 2016.

[Arm13] Mark Anthony Armstrong. Basic topology. Springer Science & Business Media, 2013.

[AS72] Milton Abramowitz and Irene A. Stegun. "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. National Bureau of Standards Applied Mathematics Series 55. Tenth Printing." In: (1972).

[Aut18] Autodesk. Autodesk Maya. June 2018. URL: http://www.autodesk.de/products/maya/overview (visited on 04/11/2020).

[Aut20] Autodesk. Fusion360. Apr. 2020. URL: `https://www.autodesk.de/products/fusion-360/overview` (visited on 04/11/2020).

[Baj+02] Chandrajit Bajaj, Scott Schaefer, Joe Warren, and Guoliang Xu. “A Subdivision Scheme for Hexahedral Meshes”. In: The visual computer 18.5-6 (2002), pp. 343–356. DOI: `10.1007/s003710100150`.

[Bar13] Pieter J. Barendrecht. “Isogeometric Analysis for Subdivision Surfaces”. MA thesis. Eindhoven, NL: Eindhoven University of Technology, Aug. 2013.

[Baz+10] Yuri Bazilevs et al. “Isogeometric analysis using T-splines”. In: Computer Methods in Applied Mechanics and Engineering 199.5-8 (2010), pp. 229–263.

[BBK18] Pieter J. Barendrecht, Michael Bartoň, and Jiři Kosinka. “Efficient quadrature rules for subdivision surfaces in isogeometric analysis”. In: Computer Methods in Applied Mechanics and Engineering 340 (2018), pp. 1–23.

[BC09] Olivier Andre Bauchau and James I Craig. Structural analysis: with applications to aerospace structures. Vol. 163. Springer Science & Business Media, 2009.

[Bei+09] Matthias Bein, Sven Havemann, André Stork, and Dieter W. Fellner. “Sketching subdivision surfaces”. In: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling. ACM. 2009, pp. 61–68. DOI: `10.1145/1572741.1572753`.

[Ben89] Martin P. Bendsøe. “Optimal shape design as a material distribution problem”. In: Structural optimization 1.4 (1989), pp. 193–202.

[BHU10a] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. “Adaptive and Feature-Preserving Subdivision for High-Quality Tetrahedral Meshes”. In: Computer Graphics Forum. Vol. 29. Wiley Online Library. 2010, pp. 117–127. DOI: `10.1111/j.1467-8659.2009.01581.x`.

[BHU10b] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. “Iso-Geometric Finite Element Analysis Based on Catmull-Clark Subdivision Solids”. In: Computer Graphics Forum 29 (5) (2010), pp. 1575–1584. DOI: `10.1111/j.1467-8659.2010.01766.x`.

[BLZ00] Henning Biermann, Adi Levin, and Denis Zorin. “Piecewise Smooth Subdivision Surfaces With Normal Control”. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 113–120. DOI: `10.1145/344779.344841`.

[BS02] Jeffrey Bolz and Peter Schröder. “Rapid Evaluation of Catmull-Clark Subdivision Surfaces”. In: Proceedings of the seventh international conference on 3D Web technology. ACM. 2002, pp. 11–17. DOI: 10.1145/504502.504505.

[Bur11] Daniel Burkhart. “Subdivision for Volumetric Finite Elements”. Dissertation. Technische Universität Kaiserslautern, 2011. ISBN: 978-3-8439-0128-4.

[BV20] Ultimaker BV. Ultimaker Cura. Mar. 2020. URL: https://ultimaker.com/software/ultimaker-cura (visited on 04/11/2020).

[Cas+09] Thomas J. Cashman, Ursula H. Augsdörfer, Neil A. Dodgson, and Malcolm A. Sabin. “NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes”. In: ACM Transactions on Graphics (TOG). Vol. 28. ACM. 2009, p. 46. DOI: 10.1145/1576246.1531352.

[CC19] Etienne Corman and Keenan Crane. “Symmetric moving frames”. In: ACM Transactions on Graphics (TOG) 38.4 (2019), pp. 1–16.

[CC78] Edwin Catmull and James Clark. “Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes”. In: Computer-aided design 10.6 (1978), pp. 350–355. DOI: 10.1016/0010-4485(78)90110-0.

[Cha74] George Merrill Chaikin. “An Algorithm for High-Speed Curve Generation”. In: Computer graphics and image processing 3.4 (1974), pp. 346–349. DOI: 10.1016/0146-664x(74)90028-8.

[CHR07] J.A. Cottrell, T.J.R. Hughes, and A. Reali. “Studies of refinement and continuity in isogeometric structural analysis”. In: Computer methods in applied mechanics and engineering 196.41-44 (2007), pp. 4160–4183.

[Cir+02] Fehmi Cirak, Michael J. Scott, Erik K. Antonsson, Michael Ortiz, and Peter Schröder. “Integrated Modeling, Finite-Element Analysis, and Engineering Design for Thin-Shell Structures Using Subdivision”. In: Computer-Aided Design 34.2 (2002), pp. 137–148. DOI: 10.1016/s0010-4485(01)00061-6.

[CMQ03] Yu-Sung Chang, Kevin T. McDonnell, and Hong Qin. “An Interpolatory Subdivision for Volumetric Models Over Simplicial Complexes”. In: Shape Modeling International, 2003. Seoul: IEEE, 2003, pp. 143–152.

[Coh+10] E. Cohen, T. Martin, R.M. Kirby, Tom Lyche, and R.F. Riesenfeld. “Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis”. In: Computer Methods in Applied Mechanics and Engineering 199.5-8 (2010), pp. 334–356.

[Cou19] European Research Council. HEXTREME. Aug. 2019. URL: `https://www.hextreme.eu/` (visited on 04/11/2020).

[CQ03] Yu-Sung Chang and Hong Qin. “Mass: Multiresolutional Adaptive Solid Subdivision”. In: (May 2003).

[Das20a] Dassault Systems. SolidWorks. Apr. 2020. URL: `https://www.solidworks.com` (visited on 04/11/2020).

[Das20b] Dassault Systems. CATIA. Jan. 2020. URL: `https://www.3ds.com/de/produkte-und-services/catia/` (visited on 04/11/2020).

[Das20c] Dassault Systems. Abaqus. Apr. 2020. URL: `https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/` (visited on 04/11/2020).

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. “Subdivision Surfaces in Character Animation”. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM. 1998, pp. 85–94. DOI: `10.1145/280814.280826`.

[DLG90] Nira Dyn, David Levine, and John A Gregory. “A butterfly subdivision scheme for surface interpolation with tension control”. In: ACM transactions on Graphics (TOG) 9.2 (1990), pp. 160–169. DOI: `10.1145/78956.78958`.

[DLP13] Tor Dokken, Tom Lyche, and Kjell Fredrik Pettersen. “Polynomial splines over locally refined box-partitions”. In: Computer Aided Geometric Design 30.3 (2013), pp. 331–356.

[DLR15] Catterina Dagnino, Paola Lamberti, and Sara Remogna. “Near-best C^2 quartic spline quasi-interpolants on type-6 tetrahedral partitions of bounded domains”. In: Calcolo 52.4 (2015), pp. 475–494.

[Dok+09] Tor Dokken, Vibeke Skytt, Jochen Haenisch, and Kjell Bengtsson. “Isogeometric representation and analysis: bridging the gap between CAD and analysis”. In: 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition. 2009, p. 1172.

[DS78] Daniel Doo and Malcolm Sabin. “Behaviour of Recursive Division Surfaces Near Extraordinary Points”. In: Computer-Aided Design 10.6 (1978), pp. 356–360. DOI: `10.1016/0010-4485(78)90111-2`.

[DSB19] Tor Dokken, Vibeke Skytt, and Oliver Barrowclough. “Trivariate spline representations for computer aided design and additive manufacturing”. In: Computers & Mathematics with Applications 78.7 (2019), pp. 2168–2182.

[DSO12] Engin Dikici, Sten Roar Snare, and Fredrik Orderud. “Isoparametric finite element analysis for Doo-Sabin subdivision models”. In: Proceedings of Graphics Interface 2012. Canadian Information Processing Society. 2012, pp. 19–26.

[DW19] Guido Dhondt and Klaus Wittig. CalculiX: A Free Software Three-Dimensional Structural Finite Element Program. Nov. 2019. URL: `http://www.calculix.de/` (visited on 04/11/2020).

[EDE17] Ben Ezair, Daniel Dikovsky, and Gershon Elber. “Fabricating Functionally Graded Material Objects Using Trimmed Trivariate Volumetric Representations”. In: Proceedings of SMI’2017 Fabrication and Sculpting Event (FASE), Berkeley, CA, USA. 2017.

[Eri13] Jeff Erickson. Theoretical Advances in Hexahedral Mesh Generation. Workshop on mesh generation, 29th Annual Symposium on Computational Geometry, Rio de Janiero, Brazil. 2013.

[EW79] C. Eastman and K. Weiler. Geometric Modeling Using the Euler Operators. Institute of Physical Planning. 1979.

[Fen+18] Leman Feng, Pierre Alliez, Laurent Busé, Hervé Delingette, and Mathieu Desbrun. “Curved Optimal Delaunay Triangulation”. In: ACM Trans. Graph. 37.4 (July 2018), 61:1–61:16. ISSN: 0730-0301. DOI: `10.1145/3197517.3201358`.

[Fie00] David A. Field. “Qualitative measures for initial meshes”. In: International Journal for Numerical Methods in Engineering 47.4 (2000), pp. 887–906. DOI: `10.1002/(sici)1097-0207(20000210)47:4<887::aid-nme804>3.3.co;2-8`.

[FLG15] Xiao-Ming Fu, Yang Liu, and Baining Guo. “Computing locally injective mappings by advanced MIPS”. In: ACM Transactions on Graphics (TOG) 34.4 (2015), p. 71. DOI: `10.1145/2766938`.

[Fou20] Blender Foundation. Blender. Apr. 2020. URL: `https://www.blender.org/` (visited on 04/11/2020).

[Fre14] Walter Frei. Parameterizing the Dimensions of Imported CAD Files. Jan. 2014. URL: `https://www.comsol.com/blogs/parameterizing-dimensions-imported-cad-files/` (visited on 04/11/2020).

[Gav15] Edwin Alan Gavidia. Adjusting Models with the SOLIDWORKS Freeform Feature. Aug. 2015. URL: `https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/10535/Adjusting-Models-with-the-SOLIDWORKS-Freeform-Feature.aspx` (visited on 04/11/2020).

[GC96] G.H. Golub and F. Charles. "Matrix computations". In: Johns Hopkins Universtiy Press, 3rd edtion (1996).

[GKS02] Eitan Grinspun, Petr Krysl, and Peter Schröder. "CHARMS: a simple framework for adaptive simulation". In: ACM transactions on graphics (TOG) 21.3 (2002), pp. 281–290. DOI: `10.1145/566570.566578`.

[GR09] Christophe Geuzaine and Jean-François Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities". In: International journal for numerical methods in engineering 79.11 (2009), pp. 1309–1331.

[Gri+99] Eitan Grinspun, Fehmi Cirak, Peter Schröder, and Michael Ortiz. Non-Linear Mechanics and Collisions for Subdivision Surfaces. Tech. rep. California Institute of Technology, 1999.

[GSP19] Xifeng Gao, Hanxiao Shen, and Daniele Panozzo. "Feature Preserving Octree-Based Hexahedral Meshing". In: Computer Graphics Forum. Vol. 38. 5. Wiley Online Library. 2019, pp. 135–149.

[GT04] Seth Green and George Turkiyyah. "Second-Order Accurate Constraint Formulation for Subdivision Finite Element Simulation of Thin Shells". In: International Journal for Numerical Methods in Engineering 61.3 (2004), pp. 380–405. DOI: `10.1002/nme.1070`.

[Han17] Eskil Hansen. Numerical Analysis - Chapter 3: Quadrature Formulas. Centre for Mathematical Sciences, Lund University, Sweden. 2017.

[Hav05] Sven Havemann. "Generative Mesh Modeling". PhD thesis. Braunschweig University of Technology, Germany, Nov. 2005.

[HCB05] Thomas J.R. Hughes, John A. Cottrell, and Yuri Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement". In: Computer methods in applied mechanics and engineering 194.39 (2005), pp. 4135–4195. DOI: `10.1016/j.cma.2004.10.008`.

[Hon+16] Min Hong, Jae-Hong Jeon, Hyo-Sub Yum, and Seung-Hyun Lee. “Plausible mass-spring system using parallel computing on mobile devices”. In: Human-centric Computing and Information Sciences 6.1 (2016), p. 23. DOI: 10.1186/s13673-016-0079-9.

[Hop+94] Hugues Hoppe et al. “Piecewise Smooth Surface Reconstruction”. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques. ACM. 1994, pp. 295–302. DOI: 10.1145/192161.192233.

[HS10] Jinghao Huang and Peter Schröder. “$\sqrt{3}$-Based 1-Form Subdivision”. In: International Conference on Curves and Surfaces. Springer. 2010, pp. 351–368.

[Hu+18] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. “Tetrahedral meshing in the wild”. In: ACM Transactions on Graphics (TOG) 37.4 (2018), p. 60. DOI: 10.1145/3197517.3201353.

[IGD19] Fraunhofer IGD. Cuttlefish. 2019. URL: https://www.cuttlefish.de/ (visited on 03/19/2019).

[Inc20a] ANSYS Inc. Ansys Mechanical Enterprise. Apr. 2020. URL: https://www.ansys.com/products/structures/ansys-mechanical-enterprise (visited on 04/11/2020).

[Inc20b] Siemens Industry Software Inc. Siemens NX. Jan. 2020. URL: https://www.plm.automation.siemens.com/global/en/products/nx/nx-for-design.html (visited on 04/11/2020).

[Inc20c] Simplify3D Inc. Simplify3D. Mar. 2020. URL: https://www.simplify3d.com/ (visited on 04/11/2020).

[ISO10] International Standards Organization ISO. Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes, ISO/TC 184/SC 4. Standard. 2010.

[Jam+15] Clément Jamin, Pierre Alliez, Mariette Yvinec, and Jean-Daniel Boissonnat. “CGALmesh: a generic framework for delaunay mesh generation”. In: ACM Transactions on Mathematical Software (TOMS) 41.4 (2015), pp. 1–24.

[JG08] Benoît Jacob and Gaël Guennebaud. Eigen: a C++ template library for linear algebra. 2008. URL: http://eigen.tuxfamily.org/ (visited on 03/19/2019).

[JKD14] Kjetil André Johannessen, Trond Kvamsdal, and Tor Dokken. “Isogeometric analysis using LR B-splines”. In: Computer Methods in Applied Mechanics and Engineering 269 (2014), pp. 471–514.

[JLW10] Zhongping Ji, Ligang Liu, and Yigang Wang. “B-Mesh: a modeling system for base meshes of 3D articulated shapes”. In: 29.7 (2010), pp. 2169–2177. DOI: `10.1111/j.1467-8659.2010.01805.x`.

[JM99] Kenneth I. Joy and Ron MacCracken. “The refinement rules for Catmull-Clark solids”. In: Technical. Report. Citeseer, 1999.

[KBK13] Michael Kremer, David Bommes, and Leif Kobbelt. “OpenVolumeMesh: A Versatile Index-Based Data Structure for 3D Polytopal Complexes”. In: Proceedings of the 21st International Meshing Roundtable. Ed. by Xiangmin Jiao and Jean-Christophe Weill. 2013, pp. 531–548. DOI: `10.1007/978-3-642-33573-0_31`.

[KSD14] Jiří Kosinka, Malcolm Sabin, and Neil Dodgson. “Creases and Boundary Conditions for Subdivision Curves”. In: Graphical Models 76.5 (2014), pp. 240–251. DOI: `10.1016/j.gmod.2014.03.004`.

[KSY09] Hyun-Jung Kim, Yu-Deok Seo, and Sung-Kie Youn. “Isogeometric analysis for trimmed CAD surfaces”. In: Computer Methods in Applied Mechanics and Engineering 198.37-40 (2009), pp. 2982–2995.

[KWA] Michel Krämer, Hendrik Martin Würz, and Christian Altenhofen. “Executing Cyclic Scientific Workflows in the Cloud”. In: Journal of Cloud Computing: Advances, Systems and Applications (JoCCASA). Currently under review.

[Lab17] Sandia National Laboratories. HTet - Cubit 15.2 User Documentation. 2017. URL: `https://cubit.sandia.gov/public/15.4/help_manual/WebHelp/mesh_generation/meshing_schemes/conversion/htet.htm` (visited on 04/11/2020).

[LB07] Dylan Lacewell and Brent Burley. “Exact Evaluation of Catmull-Clark Subdivision Surfaces near B-spline Boundaries”. In: Journal of Graphics Tools 12.3 (2007), pp. 7–15. DOI: `10.1080/2151237x.2007.10129243`.

[LBK16] Max Lyon, David Bommes, and Leif Kobbelt. “HexEx: Robust Hexahedral Mesh Extraction”. In: ACM Transactions on Graphics 35.4 (2016).

[LFS16] Xin Li, G. Thomas Finnigan, and Thomas W. Sederberg. “G1 Non-uniform Catmull-Clark Surfaces”. In: ACM Trans. Graph. 35.4 (July 2016), 135:1–135:8. ISSN: 0730-0301. DOI: `10.1145/2897824.2925924`.

[Lin+15] Hongwei Lin, Sinan Jin, Qianqian Hu, and Zhenbao Liu. “Constructing B-spline solids from tetrahedral meshes for isogeometric analysis”. In: Computer Aided Geometric Design 35 (2015), pp. 109–120.

[Liu+13] Tiantian Liu, Adam W. Bargteil, James F. O’Brien, and Ladislav Kavan. “Fast simulation of mass-spring systems”. In: ACM Transactions on Graphics (TOG) 32.6 (2013), p. 214. DOI: `10.1145/2508363.2508406`.

[Liu+18] Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. “Singularity-constrained octahedral fields for hexahedral meshing”. In: ACM Transactions on Graphics (TOG) 37.4 (2018), p. 93. DOI: `10.1145/3197517.3201344`.

[Loo87] Charles T. Loop. “Smooth Subdivision Surfaces Based on Triangles”. PhD thesis. Department of Mathematics, The University of Utah, Masters Thesis, Jan. 1987.

[LS07] Ming-Jun Lai and Larry L Schumaker. Spline functions on triangulations. 110. Cambridge University Press, 2007.

[LS08] Charles Loop and Scott Schaefer. “Approximating Catmull-Clark subdivision surfaces with bicubic patches”. In: ACM Transactions on Graphics (TOG) 27.1 (2008), pp. 1–11.

[Ltd19] Autonomous Manufacturing Ltd. Metal 3D Printing: A Definitive Guide. [Online; posted 26-June-2019]. June 2019. URL: `https://amfg.ai/2019/06/26/metal-3d-printing-a-definitive-guide/?cn-reloaded=1#Designing_for_Metal_3D_Printing` (visited on 04/11/2020).

[Luu+19] Thu Huong Luu, Christian Altenhofen, Tobias Ewald, André Stork, and Dieter Fellner. “Efficient slicing of Catmull–Clark solids for 3D printed objects with functionally graded material”. In: Computers & graphics 82 (2019), pp. 295–303.

[MA18] Robert McNeel and Associates. Rhino. June 2018. URL: `https://www.rhino3d.com/` (visited on 04/11/2020).

[Mae12] George Maestri. Displaying subdivision surfaces. June 2012. URL: `https://www.lynda.com/Maya-tutorials/Displaying-subdivision-surfaces/96715/107387-4.html` (visited on 04/11/2020).

[Mae17] George Maestri. Working with subdivision surfaces. Aug. 2017. URL: `https://www.lynda.com/Blender-tutorials/Working-subdivision-surfaces/87088/95375-4.html` (visited on 04/11/2020).

[MAE19] Fady Massarwi, Pablo Antolin, and Gershon Elber. “Volumetric untrimming: Precise decomposition of trimmed trivariates into tensor products”. In: Computer Aided Geometric Design 71 (2019), pp. 1–15.

[Mar19] Steven Marjieh. Create the highest quality surfaces in CATIA. Dec. 2019. URL: https://www.lynda.com/CATIA-tutorials/Create-highest-quality-surfaces-CATIA/2825343/2938491-4.html (visited on 04/11/2020).

[MAS17] Johannes Sebastian Mueller-Roemer, Christian Altenhofen, and André Stork. “Ternary Sparse Matrix Representation for Volumetric Mesh Subdivision and Processing on GPUs”. In: Computer Graphics Forum 36.5 (2017), pp. 59–69. DOI: 10.1111/cgf.13245.

[MCQ04] Kevin T. McDonnell, Yu-Sung Chang, and Hong Qin. “Interpolatory, Solid Subdivision of Unstructured Hexahedral Meshes”. In: The Visual Computer 20.6 (2004), pp. 418–436. DOI: 10.1007/s00371-004-0246-2.

[ME16] Fady Massarwi and Gershon Elber. “A B-spline based framework for volumetric object modeling”. In: Computer-Aided Design 78 (2016), pp. 36–47. DOI: 10.1016/j.cad.2016.05.003.

[Moe18] Kevin M. Moerman. “GIBBON: The Geometry and Image-Based Bioengineering add-On.” In: J. Open Source Software 3.22 (2018), p. 506.

[MS06] Johannes Mezger and Wolfgang Straßer. “Interactive soft object simulation with quadratic finite elements”. In: International Conference on Articulated Motion and Deformable Objects. Springer. 2006, pp. 434–443. DOI: 10.1007/11789239_45.

[MT03] Matthias Mueller and Matthias Teschner. “Volumetric meshes for real-time medical simulations”. In: Bildverarbeitung für die Medizin 2003. Springer, 2003, pp. 279–283. DOI: 10.1007/978-3-642-18993-7_57.

[Mül+02] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. “Stable real-time deformations”. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation. ACM. 2002, pp. 49–54. DOI: 10.1145/545265.545269.

[Nea+06] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. “Physically based deformable models in computer graphics”. In: Computer graphics forum. Vol. 25. Wiley Online Library. 2006, pp. 809–836. DOI: 10.1111/j.1467-8659.2006.01000.x.

[Ner19] Nervous System. Nervous System. Sept. 2019. URL: `https://n-e-r-v-o-u-s.com/` (visited on 04/11/2020).

[NKP14] Thien Nguyen, Keçstutis Karčiauskas, and Jörg Peters. “A comparative study of several classical, discrete differential and isogeometric methods for solving Poisson’s equation on the disk”. In: Axioms 3.2 (2014), pp. 280–299.

[Pan+15] Qing Pan, Guoliang Xu, Gang Xu, and Yongjie Zhang. “Isogeometric analysis based on extended Loop’s subdivision”. In: Journal of Computational Physics 299 (2015), pp. 731–746. ISSN: 0021-9991. DOI: `https://doi.org/10.1016/j.jcp.2015.06.044`. URL: `http://www.sciencedirect.com/science/article/pii/S002199911500443X`.

[Pro+95] Xavier Provot et al. “Deformation constraints in a mass-spring model to describe rigid cloth behaviour”. In: Graphics interface. Canadian Information Processing Society. 1995, pp. 147–147.

[PT12] Les Piegl and Wayne Tiller. The NURBS book. Springer Science & Business Media, 2012.

[Rab+17] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. “Scalable locally injective mappings”. In: ACM Transactions on Graphics (TOG) 36.2 (2017), p. 16. DOI: `10.1145/3072959.3126782`.

[RAF15] Andreas Riffnaller-Schiefer, Ursula H. Augsdörfer, and Dieter W. Fellner. “Isogeometric Analysis for Modelling and Design”. In: EG 2015 - Short Papers. Ed. by B. Bickel and T. Ritschel. The Eurographics Association, 2015.

[RAF16] Andreas Riffnaller-Schiefer, Ursula H. Augsdörfer, and Dieter W. Fellner. “Isogeometric shell analysis with NURBS compatible subdivision surfaces”. In: Applied Mathematics and Computation 272, Part 1 (2016). Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability, pp. 139–147. ISSN: 0096-3003. DOI: `10.1016/j.amc.2015.06.113`.

[Rif19] Andreas Riffnaller-Schiefer. “A subdivision approach to isogeometric analysis: Analysis, design and simulation”. English. Dissertation. Graz University of Technology, 2019.

[Sch96a] Robert Schneiders. “A grid-based algorithm for the generation of hexahedral element meshes”. In: Engineering with computers 12.3-4 (1996), pp. 168–177.

[Sch96b] Jean E. Schweitzer. “Analysis and Application of Subdivision Surfaces”. PhD thesis. University of Washington, 1996.

[Sed+03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. "T-splines and T-NURCCs". In: ACM transactions on graphics (TOG) 22.3 (2003), pp. 477–484.

[Set+04] Volker Settgast, Kerstin Müller, Christoph Fünfzig, and Dieter Fellner. "Adaptive tesselation of subdivision surfaces". In: Computers & Graphics 28.1 (2004), pp. 73–78. DOI: 10.1016/j.cag.2003.10.006.

[SG98] Ian M. Smith and D. V. Griffiths. Programming the finite element method. 3. ed. Chichester: Wiley, 1998. ISBN: 0-471-96543-X. URL: http://www.loc.gov/catdir/description/wiley035/96054905.html.

[SHA19] SHAPEWAYS. Shapeways. Nov. 2019. URL: https://www.shapeways.com/ (visited on 04/11/2020).

[She02] Jonathan Shewchuk. "What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)". In: University of California at Berkeley 73 (2002), p. 12.

[SHW04] Scott Schaefer, Jan Hakenberg, and J. Warren. "Smooth subdivision of tetrahedral meshes". In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. ACM. 2004, pp. 147–154. DOI: 10.1145/1057432.1057452.

[Si15] Hang Si. "TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator". In: ACM Trans. Math. Softw. 41.2 (Feb. 2015), 11:1–11:36. ISSN: 0098-3500. DOI: 10.1145/2629697.

[Sof20] Missler Software. TopSolid. Jan. 2020. URL: https://www.topsolid.com/ (visited on 04/11/2020).

[Sta98a] Jos Stam. "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values". In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM. New York, NY, USA: ACM, 1998, pp. 395–404. DOI: 10.1145/280814.280945.

[Sta98b] Jos Stam. "Fast evaluation of loop triangular subdivision surfaces at arbitrary parameter values". In: Computer Graphics (SIGGRAPH'98 Proceedings, CD-ROM Supplement). 1998.

[Stu20] Pixar Animation Studios. Pixar. Apr. 2020. URL: https://www.pixar.com/ (visited on 04/11/2020).

[SVB17] Justin Solomon, Amir Vaxman, and David Bommes. “Boundary Element Octahedral Fields in Volumes”. In: ACM Trans. Graph. 36.3 (May 2017), 28:1–28:16. ISSN: 0730-0301. DOI: 10.1145/3065254. URL: http://doi.acm.org/10.1145/3065254.

[SWB12] Robert Schmidt, Roland Wüchner, and Kai-Uwe Bletzinger. “Isogeometric analysis of trimmed NURBS geometries”. In: Computer Methods in Applied Mechanics and Engineering 241-244 (2012), pp. 93–111. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2012.05.021. URL: http://www.sciencedirect.com/science/article/pii/S0045782512001843.

[SZ07] Tatyana Sorokina and Frank Zeilfelder. “Local quasi-interpolation by cubic C^1 splines on type-6 tetrahedral partitions”. In: IMA journal of numerical analysis 27.1 (2007), pp. 74–101.

[Tam+20] Lorenzo Tamellini et al. “Parametric Shape Optimization for Combined Additive–Subtractive Manufacturing”. In: JOM 72.1 (Jan. 2020), pp. 448–457. ISSN: 1543-1851. DOI: 10.1007/s11837-019-03886-x. URL: https://doi.org/10.1007/s11837-019-03886-x.

[Thi20] Thingiverse. Thingiverse. Feb. 2020. URL: https://www.thingiverse.com/customizable (visited on 04/11/2020).

[VPR19] Kilian Verhetsel, Jeanne Pellerin, and Jean-François Remacle. “A 44-element mesh of Schneiders’ pyramid: Bounding the difficulty of hex-meshing problems”. In: Computer-Aided Design (2019), p. 102735.

[Wan+13] Wenyan Wang, Yongjie Zhang, Lei Liu, and Thomas J.R. Hughes. “Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology”. In: Computer-Aided Design 45.2 (2013), pp. 351–360.

[Web+11] Daniel Weber, Thomas Kalbe, André Stork, Dieter W. Fellner, and Michael Goesele. “Interactive Deformable Models With Quadratic Bases in Bernstein-Bézier-Form”. In: The Visual Computer 27.6 (2011), p. 473. ISSN: 1432-2315. DOI: 10.1007/s00371-011-0579-6.

[Web+13] Daniel Weber, Jan Bender, Markus Schnoes, André Stork, and Dieter Fellner. “Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications”. In: Computer Graphics Forum. Vol. 32. Wiley Online Library. 2013, pp. 16–26.

[Web+15] Daniel Weber, Johannes Mueller-Roemer, Christian Altenhofen, André Stork, and Dieter Fellner. “Deformation Simulation Using Cubic Finite Elements and Efficient p-Multigrid Methods”. In: Computers & Graphics 53 (2015), pp. 185–195. DOI: `10.1016/j.cag.2015.06.010`.

[Web15] Daniel Weber. “Interactive Physically Based Simulation - Efficient Higher-Order Elements, Multigrid Approaches and Massively Parallel Data Structures”. Dissertation. Darmstadt: Technische Universität Darmstadt, 2015.

[WHP11] Anna Wawrzinek, Klaus Hildebrandt, and Konrad Polthier. “Koiter’s Thin Shells on Catmull-Clark Limit Surfaces”. In: Vision, Modeling, and Visualization (2011). Berlin: The Eurographics Association, 2011, pp. 113–120.

[WL16] Jean-Christophe Weill and Franck Ledoux. Towards an automatic and reliable hexahedral meshing. Fifth Workshop on Grid Generation for Numerical Computations, Tetrahedron V, University of Liège, Belgium. 2016.

[WP16] Anna Wawrzinek and Konrad Polthier. “Integration of Generalized B-spline Functions on Catmull-Clark Surfaces at Singularities”. In: Computer-Aided Design 78 (2016), pp. 60–70. DOI: `10.1016/j.cad.2016.05.008`.

[YC05] Jun-Hai Yong and Fuhua Cheng. “Adaptive subdivision of Catmull-Clark subdivision surfaces”. In: Computer-Aided Design and Applications 2.1-4 (2005), pp. 253–261.

[ZK02] Denis Zorin and Daniel Kristjansson. “Evaluation of Piecewise Smooth Subdivision Surfaces”. In: The Visual Computer 18.5-6 (2002), pp. 299–315. DOI: `10.1007/s003710100149`.

[ZSS96] Denis Zorin, Peter Schröder, and Wim Sweldens. “Interpolating subdivision for meshes with arbitrary topology”. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM. 1996, pp. 189–192. DOI: `10.1145/237170.237254`.

[ZTT77] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Robert Lee Taylor. The Finite Element Method. Vol. 3. London: McGraw-hill, 1977.

[ZZT05] O. C. Zienkiewicz, J. Z. Zhu, and Robert Leroy Taylor. The finite element method: Its basis and fundamentals. 6th ed. Oxford and Boston: Elsevier Butterworth-Heinemann, 2005. ISBN: 9780750663205.

A. Publications

A.1. Publications as Main Author

Rixels: Towards Secure Interactive 3D Graphics in Engineering Clouds
C. Altenhofen, A. Dietrich, A. Stork, and D. W. Fellner
Published in *Transactions on Internet Research*
Volume 12, 2016
IPSI

Implicit Mesh Generation using Volumetric Subdivision
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner
Published in the *13th Workshop on Virtual Reality Interaction and Physical Simulation*
2017
The Eurographics Association

Volumetric Subdivision for consistent implicit Mesh Generation
C. Altenhofen, F. Schuwirth, A. Stork, and D. W. Fellner
Published in *Computers & Graphics*
Volume 69, Supplement C, 2017
Elsevier

Integrating Interactive Design and Simulation for Mass Customized 3D-Printed Objects – A Cup Holder Example
C. Altenhofen, F. Loosmann, J. S. Mueller-Roemer, T. Grasser, T. H. Luu, and A. Stork
Published in *Solid Freeform Fabrication Symposium*
Volume 28, 2017
The Minerals, Metals & Materials Society TMS

Continuous Property Gradation for Multi-Material 3D-Printed Objects
C. Altenhofen, T. H. Luu, T. Grasser, M. Dennstädt, J. S. Mueller-Roemer, D. Weber, and A. Stork
Published in *Solid Freeform Fabrication Symposium*
Volume 29, 2018
The Minerals, Metals & Materials Society TMS

Direct Limit Volumes: Constant-Time Limit Evaluation for Catmull-Clark Solids
C. Altenhofen, J. Müller, D. Weber, A. Stork and D. W. Fellner
Published in *Pacific Graphics Short Papers*
2018
The Eurographics Association

Analyzing Mesh vs. Model Quality for Isogeometric Analysis on Catmull-Clark Solids
C. Altenhofen, T. Ewald, A. Stork, and D. W. Fellner
Submitted to *IEEE Computer Graphics and Applications*
Special Issue on Geometric Modeling and Processing
IEEE Computer Society

A.2. Co-Authored Publications

Efficient Self-Shadowing using Image-Based Lighting on Glossy Surfaces
M. Knuth, C. Altenhofen, A. Kuijper, and J. Bender
Published in *Vision, Modeling & Visualization VMV*
2014
The Eurographics Association

A p-Multigrid Algorithm using Cubic Finite Elements for Efficient Deformation Simulation
D. Weber, J. S. Mueller-Roemer, C. Altenhofen, A. Stork, and D. W. Fellner
Published in *11th Workshop on Virtual Reality Interaction and Physical Simulation*
2014
The Eurographics Association

Deformation Simulation using Cubic Finite Elements and Efficient p-Multigrid Methods
D. Weber, J. S. Mueller-Roemer, C. Altenhofen, A. Stork, and D. W. Fellner
Published in *Computers & Graphics*
Volume 53, 2015
Elsevier

Flexible Integration of Cloud-Based Engineering Services using Semantic Technologies
C. Stahl, E. Bellos, C. Altenhofen, and J. Hjelmervik
Published in *IEEE International Conference on Industrial Technology*
2015
IEEE Computer Society

JIT-Compilation for Interactive Scientific Visualization
J. S. Mueller Roemer and C. Altenhofen
Published in *Short Papers Proceedings: 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*
2016
WSCG

Ternary Sparse Matrix Representation for Volumetric Mesh Subdivision and Processing on GPUs
J. S. Mueller Roemer, C. Altenhofen, and A. Stork
Published in *Computer Graphics Forum*
Volume 36, 2017
The Eurographics Association

The CloudFlow Infrastructure for Multi-Vendor Engineering Workflows: Concept and Validation
H. H. Holm, V. Gezer, S. Hermawati, C. Altenhofen, and J. Hjelmervik
Published in *International Journal on Advances in Internet Technology*
Volume 10, 2017
IARIA

Personalized Visual-Interactive Music Classification
C. Ritter, C. Altenhofen, M. Zeppelzauer, A. Kuijper, T. Schreck, and J. Bernard
Published in *EuroVA@ EuroVis*
2018
The Eurographics Association

Efficient Slicing of Catmull-Clark Solids for 3D Printed Objects with Functionally Graded Materials
T. H. Luu, C. Altenhofen, T. Ewald, A. Stork, and D. W. Feller
Published in *Computers & graphics*
Volume 82, 2019
Elsevier

Parametric Shape Optimization for Combined Additive-Subtractive Manufacturing
L. Tamellini, M. Chiumenti, C. Altenhofen, M. Attene, O. Barrowclough, M. Livesu, F. Marini, M. Martinelli, and V. Skytt
Published in *The Journal of the Minerals, Metals & Materials Society (TMS) JOM*
Volume 72, 2020
The Minerals, Metals & Materials Society TMS

Executing Cyclic Scientific Workflows in the Cloud
M. Krämer, H. M. Würz, and C. Altenhofen
Under review for the *Journal of Cloud Computing: Advances, Systems and Applications (JoCCASA)*
Springer

B. Supervisory Activities

B.1. Master's Theses

Intuitive 3D-Interaktion für Design und Modellierung von volumetrischen Strukturen
Florian Berz
2015

Gewichts- und Steifigkeitsoptimierung für die additive Fertigung
Sascha Wombacher
2017

Subdivision-Based CSM and its Application to Shape Optimization
Joel Müller
2017

GPU-Parallel Constant-Time Limit Evaluation of Catmull-Clark Solids
Sebastian Besler
2020

B.2. Bachelor's Theses

Modellierung und Subdivision volumetrischer Netze
Markus Hoth
2015

A Subdivision-Based Approach to the Heat Equation for Simulation-Based Modeling
Joel Müller
2015

Schnelle und wasserdichte Triangulierung von Punktwolken für die interaktive Simulation
Marko Rücker
2015

Direct Volume Visualization of Catmull-Clark Solids, Exploring Hardware-Accelerated Ray Tracing
Tobias Stensbeck
2020

Sketch-Based Definition and Modification of Geometric Parameters for Mass Customization
David Berman
2020

B.3. GRIS Practical Courses

Progressive Triangle Streaming for Remote Visualization of Simulation Results
Johannes Heucher
2015

Modeling Volumetric Control Meshes in Blender
Eduard Dobermann
2016

Mass-Spring Systems for Inner Control Points
Lennart Jarms
2017

Local Refinement of Catmull-Clark Solids
Kenten Fina
2018

Improving Mesh Quality of Catmull-Clark Solids
Sebastian Besler
2019

C. Acronyms

AM Additive Manufacturing

B-Rep Boundary Representation

C1-5 Contribution 1-5
CAD Computer-Aided Design
CC Catmull-Clark
CG Conjugate Gradient
CSM Computational Solid Mechanics

DoF Degree of Freedom

EE Extraordinary Edge
EV Extraordinary Vertex

FE Finite Element
FEA Finite Element Analysis
FEM Finite Element Method
FFF Fused Filament Fabrication
FVM Finite Volume Method

GP Gauss-Legendre Quadrature Point

IGA Isogeometric Analysis

KEF KillEdgeFace
KEV KillEdgeVertex
KEVFC KillEdgeVertexFaceCell
KVC KillVertexInCell
KVE KillVertexEdge

MEF	MakeEdgeFace
MEV	MakeEdgeVertex
MEVFC	MakeEdgeVertexFaceCell
MPQ	Mesh Parametrization Quality
MPR	Mesh Parametrization Regularity
MVC	MakeVertexInCell
MVE	MakeVertexEdge
NURBS	Non-Uniform Rational B-splines
OVM	OpenVolumeMesh
PDE	Partial Differential Equation
PLA	Polylactic Acid
RQ1-5	Research Question 1-5
SLE	System of Linear Equations
V-Rep	Volumetric Representation