Diss. ETH No. 20757

Discontinuous Galerkin FEM in Computer Graphics

A dissertation submitted to
ETH Zurich

for the Degree of
Doctor of Sciences

presented by
Peter Kaufmann
Dipl. Informatik-Ing., ETH Zurich, Switzerland
born February 13, 1980
citizen of Meinisberg (BE), Switzerland

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Olga Sorkine-Hornung, co-examiner
Prof. Dr. Mario Botsch, co-examiner

2013

Abstract

The finite element method (FEM) is the method of choice for computing the numerical solution to many problems that can be formulated as partial differential equations (PDEs) on irregular domains. Today, it is used in almost every field of engineering and was also quickly adopted by computer graphics research where it is now heavily used in the areas of physically-based simulation and geometric modeling. In contrast to standard FEM, which we also refer to as *continuous Galerkin* FEM (CG FEM), *discontinuous Galerkin* FEM (DG FEM) allows its basis functions to be discontinuous across elements. The lost coupling between elements is then restored by introducing penalty terms that act to reduce these discontinuities. Just like CG FEM, DG FEM has a solid mathematical foundation and can be shown to converge to the exact solution under certain conditions.

The reduced continuity requirements of DG FEM result in several advantages, which this thesis aims to exploit in the context of computer graphics: elements can have arbitrary shapes; the creation of a valid element mesh as well as the modification of the mesh over time are simplified; each element is endowed with its own set of basis functions, and the quality of the approximation to the exact solution can be chosen on a per-element basis. The work presented here applies DG FEM in the context of physically-based simulation of deformable solids and shells, as well as image warping. The method is applied to linear as well as nonlinear problems, defined through second and fourth order elliptic PDEs in two and three spatial dimensions. In each of these applications, different aspects of DG FEM are at the focus of attention. For the simulation of deformable solids, the arbitrary element shapes allow for topological changes without further element remeshing. Combined with an exact integration scheme and corotated elasticity, this enables applications such as the efficient simulation of progressive cutting, dynamic mesh refinement, and a novel mesh generation approach. In the context of shell simulations, a DG method for the Kirchhoff-Love shell equations is merged with the *extended* FEM (XFEM) to come up with a novel method for the simulation of highly detailed cutting and fracturing of shells through the use of *enrichment textures*. For image warping, DG FEM enables the generation of adaptive meshes with higher order basis functions to make the problem of temporally consistent video warping more tractable.

Zusammenfassung

Die Methode der finiten Elemente (FEM) ist die bevorzugte Methode zur Simulation vieler Probleme welche als partielle Differentialgleichungen in einem irregulären Gebiet formuliert werden können. Die Methode wird heute in praktisch jedem Teilgebiet der Ingenieurwissenschaften eingesetzt und fand auch rasch im noch relativ jungen Gebiet der Computergrafik Anwendung, dabei vorwiegend bei physikalisch basierten Simulationen sowie der geometrischen Modellierung. Im Gegensatz zur üblichen FEM, welche wir auch als *continuous Galerkin* FEM (CG FEM) bezeichnen, verwendet die *discontinuous Galerkin* FEM (DG FEM) Ansatzfunktionen, welche über Elementgrenzen hinweg unstetig sein können. Die dadurch verlorene Kopplung der Elemente wird durch neu eingeführte Terme, welche die Diskontinuitäten penalisieren, wiederhergestellt. Ebenso wie CG FEM hat auch die DG FEM eine fundierte mathematische Basis, und es kann gezeigt werden, dass die Methode unter gewissen Umständen zur exakten analytischen Lösung konvergiert.

Die reduzierten Stetigkeitsanforderungen der DG FEM führen zu einer Vielzahl von Vorteilen, welche diese Dissertation im Kontext der Computergrafik anwendet: Elemente können beliebige Formen annehmen; Die Aufteilung eines Gebietes in Elemente sowie die dynamische Anpassung dieser Aufteilung werden vereinfacht; Jedes Element hat seine eigenen Ansatzfunktionen und die Güte, mit welcher ein Element die exakte Lösung zu approximieren vermag, kann für jedes Element einzeln gewählt werden. Die vorliegende Arbeit verwendet DG FEM zur physikalisch basierten Simulation von deformierbaren Festkörpern und Schalen, sowie dem Verformen von Bildern. Die Methode wird dabei auf lineare sowohl als auch nichtlineare Probleme angewendet, welche durch elliptische Differentialgleichungen zweiter und vierter Ordnung in zwei und drei Raumdimensionen definiert sind. Für jede der gezeigten Anwendungen liegt der Schwerpunkt wieder auf einem anderen Aspekte der DG FEM. Bei der Simulation von verformbaren Festkörpern wird es dank der beliebigen Elementformen möglich, topologische Änderungen zu simulieren ohne dabei die zerschnittenen Elemente weiter aufteilen zu müssen. Kombiniert mit einem exakten Integrationsschema und korotierter Elastizität ermöglicht dies Anwendungen wie die effiziente Simulation der graduellen Zerschneidung von Objekten, eine dynamische Verfeinerung der Elemente, sowie ein neuer Ansatz zur Erzeugung der initialen Elementaufteilung.

Im Kontext der Schalensimulationen wird eine DG Methode für die Kirchhoff-Love Schalengleichungen mit der erweiterten FEM (*extended* FEM) zusammengeführt um eine neue Methode zur Simulation von detaillierten Schnitten und Brüchen unter Verwendung von Anreicherungstexturen (*enrichment textures*) zu kreieren. Bei Anwendungen in der Bildverformung erlaubt DG FEM die Erzeugung von adaptiven Elementaufteilungen mit Ansatzfunktionen höherer Ordnung, welche das Berechnen der zeitlich konsistenten Verformung von Videomaterial weniger aufwändig machen.

Acknowledgments

First and foremost, I want to thank my advisor Prof. Markus Gross for not giving up on convincing me that doing a Ph.D. in computer graphics is not such a bad idea after all, and for introducing me to the fascinating topic of discontinuous Galerkin FEM. His broad knowledge in computer graphics and his vision of what could be achieved with the DG method were of great value and always driving me forward during my Ph.D.

I was very lucky to have Prof. Mario Botsch as a supervisor. He taught me the "dos and don'ts" of paper writing and I benefited tremendously from his guidance. His uncompromised attention to detail, his pragmatic way of approaching and solving a problem, and the way he kept his desk completely uncluttered even during the most stressful deadlines had a lasting impression on me. I also thank Prof. Eitan Grinspun for the continued collaborations, for showing us how to clearly formulate an idea and how to put that "spin" on a paper. Special thanks go to Dr. Sebastian Martin, whom I had the pleasure of collaborating with on several projects. Having someone like him to talk to about my projects, on a daily basis, proved to be invaluable. Working together towards a deadline, the countless hours we spent fleshing out new ideas on the whiteboard, and the resulting "Eureka" moments were clearly some of the most rewarding parts of my Ph.D. Big thanks to all the other people I had the chance of collaborating with and who also contributed in various ways to this thesis: Dr. Oliver Wang, Prof. Olga Sorkine-Hornung, Dr. Alex Sorkine-Hornung and Dr. Aljoscha Smolic. In addition I want to thank Prof. Christoph Schwab, Prof. N. Sukumar and Prof. Max Wardetzky for the inspiring and helpful discussions.

Thanks to the past and present members of Disney Research Zurich, the Computer Graphics Lab, the former Applied Geometry Group, and the Interactive Geometry Lab for making Zurich not only one of the hotspots for graphics research, but also a great place to be and spend time with friends. I especially thank Dr. Gian-Marco Baschera for his support and for contributing a lot toward the enjoyable work environment by being a great long term office mate.

I am also very grateful for the continuous support I received from my friends and family, especially my parents who supported me in so many ways during my studies.

This thesis is dedicated to Elisabeth Maderthaner who taught me an awful lot about discontinuities in real life. Lisi, I am proud of you and incredibly glad you found your way back home again.

Contents

CHAPTER 1

Introduction

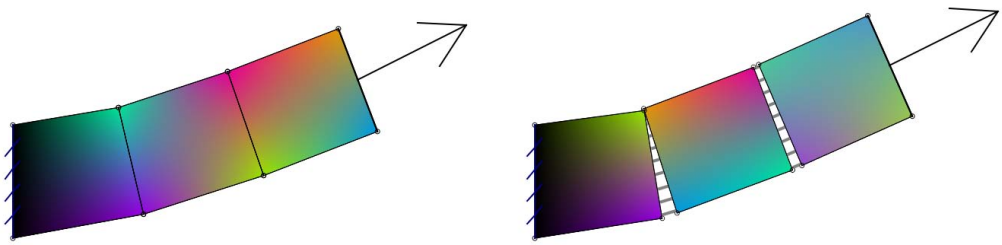

Figure 1.1: *The CG FEM (left) and DG FEM (right) solutions to a 2D elasticity problem using three elements. One edge is constrained, while a force is applied to the opposite edge. Colors indicate the area influenced by each basis function.*

Finite element methods (FEMs) have become an indispensable tool in computer graphics. Their primary use is for the physically-based simulation of deformable objects or fluids, with applications that range from computer animation to surgery simulation. But thanks to their solid mathematical foundation, they find more and more applications in other areas where problems can be formulated as partial differential equations (PDEs), such as in the field of geometric modeling. As with every tool, people tend to stretch the limits of what is possible with FEM, which can lead to problems: meshes containing badly shaped elements (*slivers*) cause numerical issues, or convergence may

be slow in some situations (*locking*). However, FEMs are still an active field of research and new variants that address these issues are constantly being developed. This thesis focuses on one such method, namely the *discontinuous Galerkin* FEM (DG FEM) [Cockburn, 2003], and investigates its application to problems in computer graphics.

DG FEM is a variant of the FEM that is characterized by its use of per-element basis functions. In standard continuous Galerkin FEM (CG FEM), each node of the element mesh is associated with a basis function, and a basis function assumes non-zero values in all elements that share the node. In contrast, in a typical DG FEM, each basis function assumes non-zero values only within exactly one element. As this inherently decouples the elements from each other, a "glue" energy is introduced, defined through a so-called *numerical flux*, that ties the elements back together. This additional energy term can be derived mathematically and just like CG FEM, the DG formulation is accurate, in the sense that the approximation converges toward the exact solution of the involved PDE under element refinement (while making additional assumptions, for example on the quality of the element mesh).

In comparison to CG FEM, the use of DG FEM enables adaptive refinement of mesh elements (*h-refinement*) and of the basis functions' polynomial degree (*p-refinement*) in a simple and efficient manner. Elements can assume arbitrary shapes and compared to CG FEM, the rules on mesh topology are less strict. The increased flexibility of DG FEM is what this thesis tries to exploit in several areas of computer graphics. Oftentimes additional advantages will emerge from the use of DG FEM: for the simulation of deformable solid objects, it allows for the exact integration of basis functions over polyhedral elements. For thin shells, it simplifies the computation of additional basis functions that can be used to represent detailed cuts and fracture curves. For image warping, it enables a simple adaptive quad-tree meshing approach.

1.1 Overview

This thesis presents applications of DG FEM to three different areas of computer graphics: the adaptive simulation of deformable solids, simulation of shell cutting and fracturing, and image warping with temporal consistency.

Simulation of Deformable Solids. In computer graphics, FEM simulations of deformable objects are mostly based on tetrahedral or hexahedral meshes. While this allows for simple and efficient implementations, topological changes of the simulation domain require complex and error-prone

remeshing to maintain a consistent simulation mesh. However, dynamically adjusting the mesh is of crucial importance in several simulation scenarios, including fracture, interactive cutting in medical applications, or adaptive refinement of complex domains.

The use of more general polyhedral elements in FEM was shown to considerably simplify cutting and fracture simulations [Wicke et al., 2007; Martin et al., 2008]. However, the strict conformity constraints of standard FEM require comparatively complex shape functions for those elements. In a slightly different context, the discontinuous element meshes of the PriMo framework enable adaptive mesh refinement for interactive shape deformation [Botsch et al., 2006; 2007]. However, due to the missing physical accuracy this method is not directly useful for physically-based simulations.

This work proposes a flexible and efficient simulation technique for corotated linear elasticity based on the discontinuous Galerkin finite element method. The presented approach conceptually generalizes the aforementioned techniques, and overcomes their limitations by combining their respective strengths: thanks to its convergence properties the DG FEM formulation is *physically accurate* and converges to the exact solution under element refinement. Similar to PriMo, the DG approach supports *arbitrary polyhedral elements* and *discontinuous meshes* with weakly enforced continuity, thereby allowing for easy and flexible mesh restructuring.

Enrichment Textures for Shells. The simulation of thin-walled structures, i.e. shells, has a long-standing history in computer graphics. Not only do they require a different mathematical treatment when compared to the simulation of solid objects, they also exhibit dramatic failure modes such as buckling, tearing, and fracturing. Simulating such phenomena requires identifying the location of discontinuities and predicting the response of neighboring material, but even representing such discontinuities is a non-trivial task. Works in the graphics and mechanics communities have considered adaptive refinement methods, which are well-suited for gradual variations in the scale of relevant features, but require too many levels of refinement to sharply resolve creases or fractures.

In an alternative point of departure, the extended finite element method (XFEM) enriches the representation with the specific basis functions required to capture the desired discontinuity. In doing so, XFEM introduces (unlike refinement) only a negligible number of new unknowns, and keeps the original mesh connectivity intact.

In this work the XFEM is extended toward the goals of graphical simulation by representing the discontinuities in *enrichment textures* that allow for a resolution much higher than that of the simulation mesh. While the presented method applies to arbitrary finite element methods in 2D and could even be extended to 3D, we focus on its application to shell simulations based on DG FEM. There are two good reasons for using DG FEM in this context: on one hand, DG FEM is a convenient approach to shell simulation because it can circumvent the strict C^1 continuity requirement imposed by the thin shell equations. On the other hand, the reduced continuity requirements also simplify the computation of the enrichment textures in our XFEM-based method.

Non-Linear Image Warping. Content-aware warping has recently been shown to be a powerful tool in a wide range of image editing applications. Warping techniques avoid difficult graphics problems such as reconstructing geometry, global illumination, and animation, while still providing convincing results. Such methods modify existing scenes by overlaying a mesh and solving for an optimal, locally-varying deformation that minimizes some application-specific set of constraints.

In traditional solutions, the constraints are defined in terms of vertex finite differences computed on a regular grid, or by discretizing the image into a quad mesh and computing per-quad energies from the distortion of grid edges. The error function is then minimized, generally by formulating it as a large sparse system of equations. However, these methods tightly couple error terms with the mesh structure, making it difficult to extend the problem formulation into new domains. Without carefully designing these error terms, there is also no guarantee for convergence when increasing the mesh resolution. Instead, this work introduces a *unifying* representation for a wide range of image editing tasks by using a finite element method that includes existing finite difference metrics as a *special case*. A single robust mathematical formulation of the general *continuous* image warping problem, combined with a finite element discretization, allows us to leverage deformation knowledge from mechanics and geometry communities.

Given this continuous framework, it becomes easier to validate and justify the use of specific energy functions that drive the warping. This work discusses how existing energy functions can be phrased in a continuous sense, and proposes simple novel energy functions with added benefits, such as effective prevention of warp inversions without resorting to workarounds or incorporating inequality constraints that require quadratic programming to solve.

In addition, this work presents a novel non-linear discontinuous Galerkin FEM formulation that allows working with meshes of arbitrary connectivity, with support for hanging nodes (edge nodes that do not belong to all elements that share the edge) that traditional FEMs cannot support. The DG FEM formulation also allows for higher order basis functions, where the order of the basis can be freely chosen on a per-element basis. This allows the sampling of high-resolution image information, while still performing a minimization on a small number of elements, and it improves the smoothness of the results with fewer elements.

1.2 Principal Contributions

This thesis makes the following main contributions:

- **A DG FEM-based simulation method for solids supporting arbitrary polyhedral elements and topological changes.** We present several extensions to the DG FEM that ultimately allow for the flexible and efficient simulation of deformable models for computer graphics applications. One key problem is the integration of functions over volumetric elements. While numerical integration is typically employed here, this work presents a fast and accurate volumetric integration technique for arbitrary polyhedral elements. To allow for large deformations while only using a simple linear elasticity model, the stiffness warping approach is generalized to DG FEM. To embed high-resolution surface meshes in coarse DG FEM meshes despite the inherent discontinuous nature of their displacement fields, a smooth interpolation method based on moving least squares (MLS) interpolation is presented (Chapter 4).
- **Harmonic Enrichment Textures.** We present a unified XFEM framework based on *harmonic enrichment functions*, which allows for multiple, intersecting, and arbitrarily-shaped cuts per element, while being easy to define, compute, and use. Furthermore, a discrete representation of the enrichment functions using *enrichment textures* simplifies the specification and computation of cuts and paves the way for the future incorporation of GPU-based techniques (Chapter 5).
- **Enriched, Corotated DG FEM Thin Shells.** The application of enrichment textures to thin shell simulations allows for the representation of discontinuities such as creasing, tearing, cutting and fracture. Using a new corotational extension to Noels's linear DG FEM thin

shells, accurate simulations of the material's dynamic response to time-varying discontinuities can be computed (Chapter 5).

- **DG FEM Warping Framework.** We propose a novel, general representation for continuous locally-varying image warping that models deformation using a DG finite element method, offering advantages such as arbitrary mesh connectivity, a well defined, continuous problem formulation, well behaved convergence properties, and support for higher order basis. These properties in turn allow for highly adaptive meshes that reduce the number of degrees of freedom and make the computation of temporally stable video retargeting solutions more tractable (Chapter 6).
- **Continuous Warping Formulation.** We formulate existing grid and cell-based image warping energies as continuous deformation energy densities. This allows us to relate them to material models known from continuum mechanics and analyze their invariance to translation, scaling and rotation. This approach further enables us to reuse principles from existing material models to find warp energies with certain desirable properties such as lack of self-intersections (Chapter 6).

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 discusses related work in the fields of FEM, DG FEM, simulation of deformable solids, shells, and image warping. Chapter 3 provides an introduction to the discontinuous Galerkin finite element method and its relation to standard (continuous) FEM for linear as well as non-linear elliptic problems. As this work focuses on the simulation of deformable objects, an introduction to non-linear elasticity theory is provided, followed by its reduction to a fully linear theory. Finally, the volumetric theory is applied to the special case of thin objects, which reproduces the standard Kirchhoff-Love thin shell model. In Chapter 4, the DG FEM is applied to linear elasticity and extended further to allow for large deformations through corotation. By using a novel analytic integration approach, the method can be applied to arbitrary polyhedral elements, which allows for simple mesh generation, adaptivity, and topological changes without re-meshing of the cut elements. Chapter 5 applies DG FEM to a corotated shell model to reduce the continuity requirements of the underlying basis functions. This, in turn, enables the development of a new method inspired by the extended finite element method (XFEM) which allows for the simulation of progressive cuts

and very detailed fracturing even when the underlying simulation mesh uses a coarse discretization. Chapter 6 applies the DG FEM in a slightly different context, namely the computation of optimal image warps. A non-linear DG FEM formulation is used here, as it allows for the definition of warp energies that disallow element inversions and thus produce non-intersecting image warps. Chapter 7 concludes the thesis and discusses its main contributions and potential future work. Additional material and proofs are given in the Appendices.

1.4 Publications

In the context of this thesis, the following publications have been accepted.

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM, *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (Dublin, Ireland, July 7-9, 2008)*, pp. 105–115, 2008.

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM, *Journal of Graphical Models, 2009, Special Issue of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2008*, vol. 71, no. 4, pp. 153–167, 2009.

P. KAUFMANN, S. MARTIN, M. BOTSCH, E. GRINSPUN, and M. GROSS. Enrichment Textures for Detailed Cutting of Shells, *Proceedings of ACM SIGGRAPH (New Orleans, USA, August 3-7, 2009), ACM Transactions on Graphics*, vol. 28, no. 3, pp. 50:1–50:10, 2009.

P. KAUFMANN, O. WANG, A. HORNUNG, O. SORKINE, A. SMOLIC, and M. GROSS. Finite Element Image Warping, *Proceedings of Eurographics (Girona, Spain, May 6-10, 2013), Computer Graphics Forum*, vol. 32, no. 2, 2013

CHAPTER 2

Related Work

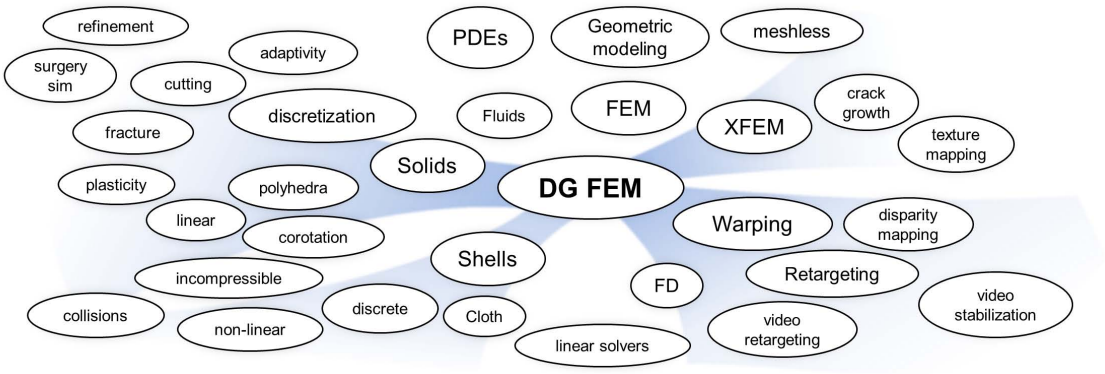

Figure 2.1: *Visualization of topics related to this thesis.*

The finite element method has a long history in computer graphics. It has been commonly used in domains such as physically based animation [Terzopoulos and Fleischer, 1988a], and remains the method of choice for the simulation of deformable objects. Recent applications include geometric modeling [Jacobson et al., 2010] and surface parameterization [Sharf et al., 2007]. One can distinguish between linear FEM where the corresponding energy function is quadratic in the unknowns and its minimum can be found by solving a sparse linear system, and the more general case of nonlinear FEM applicable to nonlinear minimization problems [Bonet and Wood, 1997].

Next to the "standard textbook" FEM [Hughes, 2000], a number of variants exist, including discontinuous Galerkin FEM (DG FEM) [Arnold et al., 2001], mixed FEM [Arnold, 1990], and extended FEM (XFEM) [Moës et al., 1999].

This chapter presents some of the related work on FEM relevant to computer graphics, with a focus on the physically based simulation of deformable solids and shells. In their corresponding contexts, related work on alternative methods such as finite difference approaches and meshless methods will be mentioned as well.

2.1 Discontinuous Galerkin FEM

The discontinuous Galerkin finite element method was first proposed by Reed and Hill [1973] as a method for solving neuron transport problems. The basic idea of DG FEM, i.e., employing discontinuous shape functions and weakly enforcing boundary constraints and inter-element continuity through penalty forces, is rather old [Babuška and Zlámal, 1973; Douglas and Dupont, 1976]. In the last decade, however, DG FEM regained increasing attention in applied mathematics [Arnold et al., 2001; Cockburn, 2003] and has since been successfully applied to hyperbolic, parabolic, elliptic, and mixed problems.

The main strength of DG FEM is its support for irregular, non-conforming meshes, and for shape functions of different polynomial degree, which in combination allows for flexible hp-refinement. In applied mathematics and mechanics, DG FEM has successfully been employed for linear and nonlinear elasticity [Lew et al., 2004; Ten Eyck and Lew, 2006; Wihler, 2006], where it was shown to provide an accuracy similar to CG FEM at comparable computational cost. Another advantage of DG FEM is the absence of locking even for nearly incompressible deformable objects [Wihler, 2006]. The method can be customized for higher order continuity, such as weakly enforcing C^1 continuity using only C^0 continuous basis functions. This makes DG FEM a practical choice for the simulation of linear [Noels and Radovitzky, 2008] and non-linear [Noels, 2009] shells. DG FEM can also be combined with other methods. For example, a combination of DG FEM and extended FEM (XFEM) was studied by Gracie et al. [2008].

2.2 Physically Based Simulation

In computer graphics, Terzopoulos et al. [1987] pioneered the use physically-based methods for graphics application, and they have since been successfully

employed for the simulation of deformable solids [Müller et al., 2002], thin shells [Grinspun et al., 2003], cloth [Baraff and Witkin, 1998], and fluids [Stam, 1999]. For the related work presented here, our focus is on the simulation of deformable solid objects and shells. In both cases, we also look at existing approaches for the simulation of fracture, cutting, and mesh adaptivity. What these effects have in common is that they require changes to the underlying discretization, which in turn is closely dependent on the underlying simulation method.

2.2.1 Simulation of Deformable Solids

Most real-world materials exhibit a highly non-linear deformation behavior when put under stress, and the need to handle large rotational deformations of objects leads to non-linear deformation models. However, since physical accuracy is not the primary goal in most graphics applications, one can often resort to the physically plausible, robust, and efficient *corotated linear elasticity* [Müller and Gross, 2004; Hauth and Strasser, 2004] instead of running a full non-linear simulation. To alleviate some of the problems associated with FEM simulations, such as locking for nearly incompressible materials, methods have been introduced that automatically reduce the physical accuracy of the simulation when these problems occur [Irving et al., 2007]. For a more detailed survey of this topic the reader may consult Nealen et al. [2006].

Cutting & Fracture of Solids. The simulation of cutting and fracture of deformable objects dates back to the pioneering work of Terzopoulos et al. [1987; 1988b], and was brought to the forefront by O'Brien and Hodgins [1999] in their work on brittle (and ductile [2002]) fracture of volumetric elastica. As an alternative to simulation, some have considered purely procedural approaches [Desbenoit et al., 2005]. However, we focus our discussion to physically simulated material response in the presence of cuts and fracture.

When combined with FEM, the topological changes induced by cutting pose two distinct challenges: first, to adapt basis functions, boundary conditions, and to update the stiffness matrix; second, to restructure the underlying mesh connectivity so as to represent the newly-created surfaces at a high resolution. Most research has focused on solid objects, where the problem of updating mesh connectivity is particularly challenging.

Apart from simply removing the primitives of the underlying element mesh touched by the cutting plane [Forest et al., 2002], fracturing of deformable solids can efficiently be performed by restricting cuts to exist-

ing element boundaries or predefined positions [Müller and Gross, 2004; Terzopoulos and Fleischer, 1988b]. However, these approaches are typically not accurate enough for more sophisticated simulations. Splitting individual elements allows for precise fracturing and cutting, but in turn requires element decompositions [Bielser et al., 1999; Bielser and Gross, 2000; Bielser et al., 2003] and/or general remeshing [O'Brien and Hodgins, 1999; O'Brien et al., 2002; Steinemann et al., 2006a]. When accommodating the crack surface, special care has to be taken to avoid numerically unstable sliver elements. Similarly, Bargteil et al. [2007] performed remeshing to remove degenerate elements during large plastic deformations.

Meshless approaches intrinsically avoid remeshing by using particles instead of a simulation mesh [Müller et al., 2004a]. While this considerably simplifies the actual topological changes, the material distance, which controls the mutual influence of simulation nodes, has to be adjusted. This can be accomplished either by recomputing special shape functions [Pauly et al., 2005] or by updating a distance graph [Steinemann et al., 2006b]. Note, however, that these approaches still require resampling in order to guarantee a sufficiently dense discretization in the vicinity of cracks and cuts.

A mesh-based alternative to remeshing is the virtual node algorithm [Molino et al., 2004], which, instead of splitting elements, duplicates them and embeds the surface in both copies. While the original approach was limited to cutting each element at most three times, its generalization [Sifakis et al., 2007a; 2007b] overcomes this restriction by embedding a high-resolution two-dimensional material boundary mesh in the coarser tetrahedral mesh. These two works served as the foundation for the efficient fracture of rigid materials proposed by Bao et al. [2007].

Wicke et al. [2007] and Martin et al. [2008] avoid remeshing of cut elements into consistent tetrahedra by directly supporting general polyhedra in FEM simulations. The drawback of their methods, however, is the comparatively complex computation and integration of the employed generalized barycentric shape functions.

Adaptive Simulation. The steadily growing complexity of geometric objects as well as of physical models results in an increasing demand for adaptive simulations, allowing to concentrate computing resources to interesting regions of the simulation domain [Debunne et al., 2001; Grinspun et al., 2002; Capell et al., 2002; Otaduy et al., 2007]. When adaptively refining the mesh, special care has to be taken to avoid or to properly handle hanging nodes.

This problem can be circumvented by subdividing basis functions instead of elements [Grinspun et al., 2002; Capell et al., 2002]. However, in order to ensure linear independence of basis functions, Grinspun et al. [2002] restrict the refinement to one level difference between neighboring elements. In contrast, the hybrid simulation [Sifakis et al., 2007b] allows for multi-level hanging nodes by constraining them to edges using either hard or soft constraints.

Another approach for reducing computational complexity is to embed a high resolution surface mesh into a coarser simulation mesh [Faloutsos et al., 1997; Capell et al., 2002; Molino et al., 2004; Müller and Gross, 2004; Müller et al., 2004b; James et al., 2004; Sifakis et al., 2007b]. The nodal displacements of the coarse mesh are then interpolated onto the surface mesh. A similar space deformation approach was employed for interactive shape deformation in Botsch et al. [2007], where furthermore a discontinuous mesh with "glue-like" continuity energies allowed for easy and flexible mesh refinement.

2.2.2 Simulation of Thin Shells

Shells were amongst the first objects to be simulated in computer graphics [Terzopoulos et al., 1987] and a huge variety of models is in use today. Approaches using explicit stretch, bending and shearing forces are typically used for cloth simulations [Baraff and Witkin, 1998; Bridson et al., 2002]. More accurate models can be derived by applying geometric operators over triangle meshes [Grinspun et al., 2003] that approximate a physically-motivated thin shell deformation energy. Finite element based shell models can directly be derived from the continuum formulation of shell mechanics. However, they must use basis functions that are able to reproduce C^1 continuous displacements fields and finding such basis functions for irregular meshes is a non-trivial task. As a solution to this problem, some approaches introduce additional non-nodal degrees of freedom such as derivatives at edge mid-points [Zienkiewicz and Taylor, 2000] while others circumvent the problem by increasing the support of basis functions over more elements [Cirak et al., 2000]. Further specialized models exist, for example to handle cloth with non-flat rest state [Bridson et al., 2003] or inextensible cloth and shells [Goldenthal et al., 2007; English and Bridson, 2008].

Shell Cutting & Fracturing. The graphical simulation of shell cutting and fracture has recently received specific attention. Mesh-based methods such as those of Boux de Casson and Laugier [2000], who tear discrete models of cloth [Baraff and Witkin, 1998; Choi and Ko, 2002], and Gingold et al. [2004],

who fracture discrete shells [Grinspun et al., 2003], split meshes along existing edges. Müller [2008] describes a fast method for simulating tearing cloth using position based dynamics. Guo et al. [2006] and Wicke et al. [2005] propose meshless methods so as not to be limited by mesh connectivity. The approach presented in this work differs in that it targets the widely-used mesh-based setting, but seeks to do so without restricting cuts to mesh resolution by turning to basis enrichment in an FEM context.

Basis Enrichment & XFEM. An alternative way of implementing cutting and fracture simulations is by directly modifying the underlying basis functions. The CHARMS framework enriches the FEM basis [Grinspun et al., 2002] and supports topological changes, but cutting and fracture are not considered. The XFEM method, introduced by Belytschko and Black [1999], explicitly targets fracture. Since XFEM has been explored over the past decade, a more comprehensive summary requires a thorough survey, such as the one by Abdelaziz and Hamouine [2008]; only a couple of representative works are discussed in this section. Moës et al. [1999] explicitly consider multiple *straight-line* crack tips within one element. Moës et al. [2002] further studied curved crack tips in three dimensions. Huang et al. [2003] simulate multiple cracks, but the example problem (mudcracks) assumes that cracks do not intersect. Stazi et al. [2003] consider higher-order elements and quadratic cracks. In general, this body of work employs (in order to accurately resolve strain) elemental radii orders of magnitude smaller than the characteristic radii of crack shapes; furthermore, for improved quadrature, they partition a split element into subdomains (sometimes with a hierarchical construction). This makes it challenging to consider complex cut shapes. This work considers the diametric opposite: finely-detailed cuts *inside* one or a few elements.

2.3 Image Warping

Traditional image-based warping is a long running and large area of research within computer graphics. Beier et al. [1992] present a classic example of mesh-based image warping that morphs between images by mapping features. More recently, advances in computing power have allowed for content-aware image warping techniques that compute globally optimal distortions of images. These methods have been successful in a wide range of applications, such as: media retargeting [Shamir and Sorkine, 2009], video stabilization [Liu et al., 2009], fish-eye lens distortion correction [Carroll et

al., 2009], perspective modification [Carroll et al., 2010], stereoscopic editing [Lang et al., 2010], and image-based rendering [Chaurasia et al., 2011].

Discretizations. Most image warping solvers operate on a uniform grid, discretize their deformation energies using finite difference-type methods, and solve the resulting optimization problem using iterative methods [Shamir and Sorkine, 2009]. With the exception of a few methods, FEM has been largely ignored in the image warping domain. One such method proposes the use of finite elements in medical image warping for registration [Gee, 1994]. However, in this case, a simple linear finite element model is used.

Deformation Energies. Depending on the application, image warping approaches use different energy terms to penalize different kinds of image distortion. One can either penalize all deformations except for translations [Shamir and Sorkine, 2009], penalize rotations [Wang et al., 2008; Laffont et al., 2010], allow rotations but penalize scaling [Wang et al., 2010], or penalize all deformations except similarity transformations [Zhang et al., 2009]. More recently, Panozzo et al. [2012] demonstrated that very convincing image warping results can be achieved in real time by only allowing axis-aligned deformations.

Temporally Stable Video Retargeting. In the context of video retargeting, previous methods that compute solutions for full video sequences have had to choose between two options: representing videos with a sparsely sampled mesh, which gives insufficient control over regions that require high-frequency changes in distortion [Wang et al., 2010], or using a dense representation, which quickly scales beyond reasonable computation for video sequences. As a result, many methods have attempted to reduce the effects of temporal artifacts while solving for local deformations by enforcing neighboring frame consistency on a frame-by-frame or windowed basis [Guttmann et al., 2009; Krähenbühl et al., 2009; Greisen et al., 2012], or by introducing motion-aware importance maps [Wang et al., 2009; Niu et al., 2010]. Alternative methods entirely avoid a solve over the whole video cube by treating the spatial and temporal components of the video retargeting problem independently [Wang et al., 2011]

In contrast to previous methods, this work presents a method that can solve a full sequence of frames at once without sacrificing accuracy, using an adaptive FEM mesh to substantially reduce the total number of degrees of freedom of the problem without harming visual quality. Non-uniform meshes have

previously been employed in the context of image warping. For example, using meshes with multiple levels of refinement for a content-aware zooming application [Laffont et al., 2010]. In this case, a Delaunay triangulation creates an initial mesh with denser mesh levels created by triangle subdivision. However, this method can only use a combination of several fixed-resolution meshes.

CHAPTER 3

Fundamentals

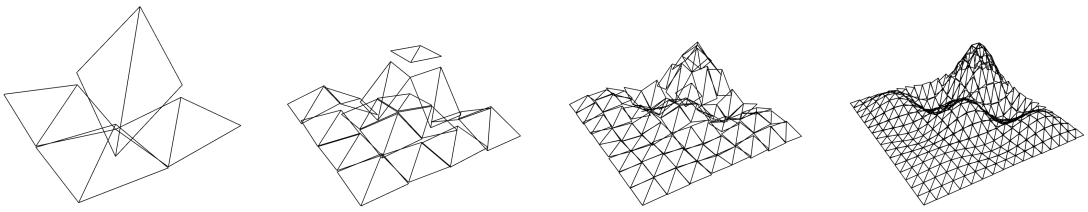

Figure 3.1: *DG FEM solutions of a 2D Poisson problem for different discretization levels.*

This chapter provides an introduction to the discontinuous Galerkin finite element method (DG FEM) by first considering the 'standard' case of applying continuous Galerkin (CG) FEM to a model problem, then showing how the derivation of DG methods differs from the more traditional approaches (Section 3.1). In Section 3.2, a more generic derivation for DG FEM is shown, which will also be applicable to non-linear problems.

One of the main applications of FEM in computer graphics is the dynamic simulation of deformable objects, and the idea of applying DG FEM to the same kind of problems follows naturally. In order to derive the corresponding partial differential equations (PDEs), we need a basic understanding of continuum mechanics for the non-linear and the simplified linear case (Section 3.3 and Section 3.4). The simulation of deformable objects that are thin in one spatial direction requires special treatment, and the derivation of the corresponding equations is shown in Section 3.5.

3.1 Introduction to DG FEM

This section introduces the concepts of DG FEM and points out the main differences to standard CG FEM. More details on CG FEM can be found in textbooks such as Bathe [1995] and Hughes [2000], while the survey articles of Arnold et al. [2001] and Cockburn [2003] provide a useful introduction to DG FEM.

Model Problem. In the following, both CG and DG FEM are discussed based on a simple 2D Poisson problem with homogeneous Dirichlet boundary constraints

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^2, \quad u = 0 \quad \text{on } \partial\Omega, \tag{3.1}$$

where $u : \Omega \to \mathbb{R}$ is an unknown function and $f : \Omega \to \mathbb{R}$ a given forcing term (see Fig. 3.2). Using index notation, the 2D Poisson problem can be written as

$$-\sum_i u_{,ii} = f \quad \text{in } \Omega,$$

with the coma denoting partial differentiation with respect to the indicated spatial dimension, e.g. $u_{,2} = \partial u / \partial x_2 = \partial u / \partial y$ for $\mathbf{x} = (x_1, x_2)^T = (x, y)^T$.

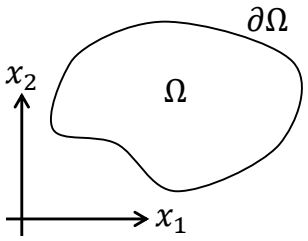

Figure 3.2: *Domain and boundary of a 2D Poisson problem.*

Strong Form, Weak Form, Energy Minimization. The so-called *strong form* (3.1) of the problem, which defines the PDE for which we need to find a solution, is not the only way of formulating the problem. Depending on the context and the approach taken, other representations can seem more natural and provide us with additional insights into the problem. For the problems considered in this context, the solutions to the PDE are actually minimizers for a potential energy functional $E[u]$. When looking at elasticity problems for example, this energy functional corresponds directly to the physical deformation energy of the simulated system. To find a solution u for which the energy functional assumes a (local) minimum, one can require its

first variation $\delta E[u]$ to be equal to zero. This results in the so-called *weak form* which employs a test function v and plays a fundamental role in the FEM. Converting a problem from its strong form to the weak form (or the other way around) is a standard technique, but requires the involved functions to fulfill some smoothness criteria. Also, the strong form can be recovered directly from the energy functional by applying the Euler-Lagrange equation. Fig. 3.3 shows the relations and transitions between these three representations.

One way of succinctly describing the FEM is by considering it to be a method for solving the discretized weak form (see Section 3.1.1). An alternative point of view is to directly discretize the energy, making fewer assumptions on the underlying problem and applying standard techniques for function minimization (see Section 3.3). The classical FEM steps, such as computing per-element stiffness matrices and updating a global stiffness matrix, will emerge automatically as substeps of the minimization algorithm.

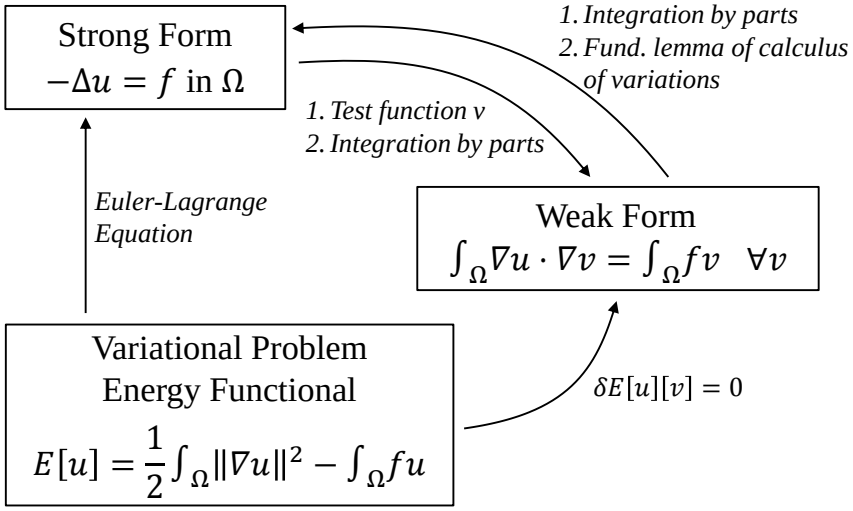

Figure 3.3: *Relations between the strong form, weak form and energy minimization problem for the model problem.*

3.1.1 CG FEM

The *integration by parts identity* states that for all scalar functions $u, v \in H^1(\Omega)$ it holds that

$$\int_\Omega u_{,i} v = \int_{\partial\Omega} u v n_i - \int_\Omega u v_{,i} \,, \tag{3.2}$$

where n_i is the i-th component of the outward unit normal of $\partial\Omega$. The Sobolev space $H^k(\Omega)$ is defined as $H^k(\Omega) := \{w \big| w \in L_2(\Omega), \ldots, D^k w \in L_2(\Omega)\}$, i.e. it

contains all functions whose derivatives up to order k are square integrable, as $L_2(\Omega) := \{w | \int_\Omega |w|^2 < \infty\}$.

The standard FEM approach is to multiply the above so-called *strong form* (3.1) by a suitable scalar test function v, resulting in

$$-\Delta u\, v = f\, v \quad \forall v,$$

and performing an integration by parts using (3.2), yielding the *weak form*

$$a_{\mathrm{CG}}(u,v) := \int_\Omega \nabla u \cdot \nabla v \;=\; \int_\Omega f v, \tag{3.3}$$

which is defined in terms of the bilinear form $a_{\mathrm{CG}}(\cdot,\cdot)$. The goal is to find a function u, such that the weak form (3.3) holds for all suitable test functions v vanishing on the boundary $\partial\Omega$.

Discretization. In order to discretize (3.3) the domain Ω is partitioned into finite elements $K \in \mathcal{T}$ (see Fig. 3.4). On top of this tessellation a set of basis functions $\{N_1, \ldots, N_n\}$ with $N_i : \Omega \to \mathbb{R}$ is defined and used to approximate u as

$$u(\mathbf{x}) \;\approx\; \sum_{i=1}^{n} u_i N_i(\mathbf{x}). \tag{3.4}$$

For a weak form containing m'th partial derivatives, standard FEM requires basis functions N_i from the Sobolev space $H^m(\Omega)$. This in particular restricts the basis functions to be *conforming*, i.e., C^m continuous within and C^{m-1} continuous across elements [Hughes, 2000]. For our Poisson example with weak form (3.3) the N_i therefore have to be C^0 *continuous* across elements.

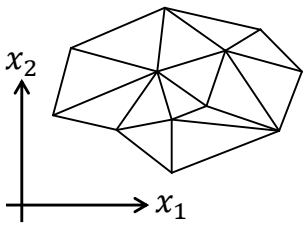

Figure 3.4: *One possible tessellation of the domain Ω using triangular elements.*

Approximating both u and v by the shape functions N_i and exploiting the bilinearity of $a(\cdot,\cdot)$ leads to

$$\sum_{ij} v_i K_{ij} u_j = \sum_i v_i f_i \quad \forall v_i$$

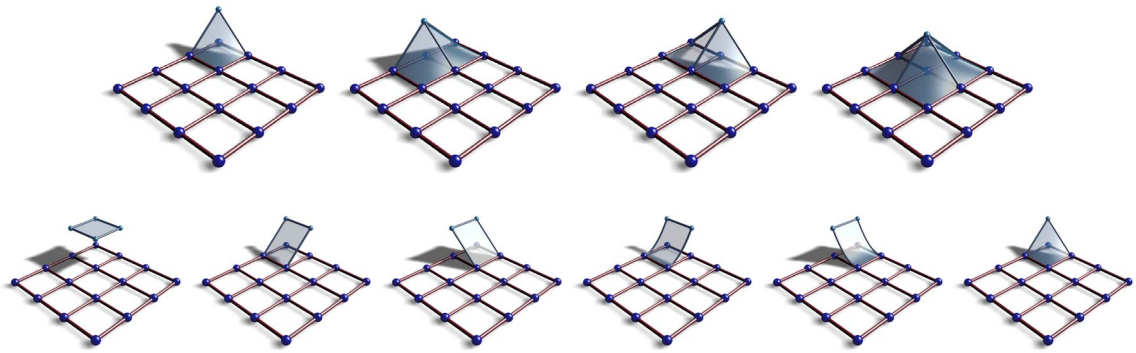

Figure 3.5: *Bilinear continuous basis functions (top row) and quadratic polynomial discontinuous basis functions (bottom row) associated with the corner element of a regular* 3×3 *quad mesh.*

with $K_{ij} = a_{\mathrm{CG}}(N_i, N_j)$ and $f_i = \int_\Omega f\, N_i$. Choosing $v_i = \delta_{ik}$ for $k = 1, \ldots, n$, where δ denotes the Kronecker delta, leads to a system of n linear equations that can be written in matrix notation as

$$\mathbf{K} \cdot \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}, \tag{3.5}$$

where the matrix $\mathbf{K}$ consists of entries $(\mathbf{K})_{ij} = K_{ij}$. This linear system is solved for the unknown coefficients u_i to find the discretized solution to the original problem.

3.1.2 DG Primal Formulation

In contrast to the above approach, DG FEM allows for *non-conforming* or *discontinuous* shape functions N_i (see Fig. 3.5), thereby resulting in discontinuous approximations of u. The weak form will therefore first be formulated for each element $K \in \mathcal{T}$ individually, and those are to be combined by taking the discontinuities across neighboring elements into account. Before doing so, the second order PDE of the strong form (3.1) is split into two first order PDEs by introducing the helper function $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^2$:

$$\boldsymbol{\sigma} = \nabla u\,, \quad -\nabla \cdot \boldsymbol{\sigma} = f \tag{3.6}$$

or using index notation,

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2)^T = (u_{,1}, u_{,2})^T\,, \quad -\sum_i \sigma_{i,i} = f.$$

To derive the weak form of a single element K, these two equations are multiplied by scalar- and vector-valued test functions v and $\boldsymbol{\tau}$, respectively. Integrating the result by parts over K yields additional boundary integrals over ∂K, leading to the *local* weak form of element K

$$\int_K \boldsymbol{\sigma} \cdot \boldsymbol{\tau} \;=\; -\int_K u \nabla \cdot \boldsymbol{\tau} \;+\; \int_{\partial K} u\, \boldsymbol{\tau} \cdot \mathbf{n}_K, \tag{3.7}$$

$$\int_K \boldsymbol{\sigma} \cdot \nabla v \;=\; \int_K f v \;+\; \int_{\partial K} v\, \boldsymbol{\sigma} \cdot \mathbf{n}_K, \tag{3.8}$$

where $\mathbf{n}_K$ denotes the unit outward normal of K.

The *global* weak form, which integrates over the whole domain Ω, is built by summing up the individual elements' weak forms (3.7), (3.8). Note that in CG FEM the boundary integrals over interior edges would cancel out, eventually leading to (3.3). In the DG setting, however, u and $\boldsymbol{\sigma}$ are *discontinuous* across elements, hence requiring special attention to be paid to the integrals over ∂K.

To account for that, the DG formulation replaces the functions u and $\boldsymbol{\sigma}$ in those boundary integrals by their so-called *numerical fluxes* $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$, respectively. The fluxes are responsible for "gluing together" the functions u and $\boldsymbol{\sigma}$ across element boundaries, which is achieved by some penalty term that *weakly* enforces continuity. Concrete examples for the fluxes $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$ will be presented later. For now they can be imagined as the average of the function values from both sides of the edge. This yields the global weak form

$$\int_\Omega \boldsymbol{\sigma} \cdot \boldsymbol{\tau} \;=\; -\int_\Omega u \nabla \cdot \boldsymbol{\tau} \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} \hat{u}\, \boldsymbol{\tau} \cdot \mathbf{n}_K, \tag{3.9}$$

$$\int_\Omega \boldsymbol{\sigma} \cdot \nabla v \;=\; \int_\Omega f v \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} v\, \hat{\boldsymbol{\sigma}} \cdot \mathbf{n}_K. \tag{3.10}$$

After introducing the fluxes $\hat{u}$ and $\hat{\boldsymbol{\sigma}}$, the helper function $\boldsymbol{\sigma}$ can be removed by choosing $\boldsymbol{\tau} = \nabla v$ in (3.9) and inserting the result into (3.10). Applying integration by parts once more then leads to

$$\int_\Omega \nabla u \cdot \nabla v \;+\; \sum_{K \in \mathcal{T}} \int_{\partial K} \left((\hat{u} - u)\, \nabla v - v\, \hat{\boldsymbol{\sigma}} \right) \cdot \mathbf{n}_K = \int_\Omega f v. \tag{3.11}$$

In the above equations each interior edge e, shared by two elements K^- and K^+, is integrated over twice, since $e \subset \partial K^-$ and $e \subset \partial K^+$. In order to exploit this, let us for a function q on e denote by

$$q^\pm \;:=\; q|_{\partial K^\pm}$$

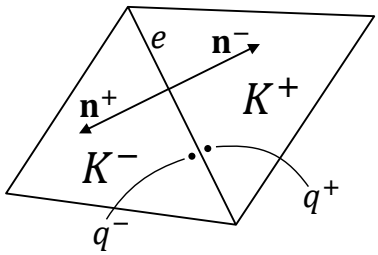

Figure 3.6: *Adjacent elements K^- and K^+ sharing edge e along with their corresponding outward unit normals $\mathbf{n}^-$ and $\mathbf{n}^+$ and bi-valued function $q^\pm$ evaluated on either side of the edge.*

its function value taken from either ∂K^+ or ∂K^-, respectively (see Fig. 3.6). With this we define[1] the *average* operator $\{\cdot\}$ and the *jump* operator $[\![\cdot]\!]$ for scalar-valued functions u and vector-valued functions σ as

$$\{u\} := \frac{1}{2}\left(u^- + u^+\right), \qquad [\![u]\!] := u^-\mathbf{n}^- + u^+\mathbf{n}^+, \tag{3.12}$$
$$\{\sigma\} := \frac{1}{2}\left(\sigma^- + \sigma^+\right), \qquad [\![\sigma]\!] := \sigma^- \cdot \mathbf{n}^- + \sigma^+ \cdot \mathbf{n}^+,$$

with "$\cdot$" denoting the vector dot product. Note that the average operator maps scalars to scalars and vectors to vectors, whereas the jump operator swaps these representations. With those operators, and with

$$\Gamma := \cup_K \partial K \quad \text{and} \quad \Gamma^\circ := \Gamma \setminus \partial\Omega$$

denoting the set of all edges and all interior edges, respectively, we can avoid integrating twice over interior edges and simplify (3.11) to

$$\begin{aligned} a_{\mathrm{DG}}(u,v) &:= \int_\Omega \nabla u \cdot \nabla v \\ &+ \boxed{\int_\Gamma \left([\![\hat{u} - u]\!] \cdot \{\nabla v\} - [\![v]\!] \cdot \{\hat{\sigma}\}\right)} \\ &+ \boxed{\int_{\Gamma^\circ} \left(\{\hat{u} - u\} \cdot [\![\nabla v]\!] - \{v\} \cdot [\![\hat{\sigma}]\!]\right)} \\ &= \int_\Omega f v. \end{aligned} \tag{3.13}$$

This equation is called the *primal formulation*, and is the DG equivalent to the CG weak form (3.3). It differs in the framed edge integrals only, which — with suitable fluxes $\hat{u}$ and $\hat{\sigma}$ — penalize the discontinuities across elements, as discussed in the following.

[1] Unfortunately, several conflicting definitions of these operators are used in the literature. In order to stay consistent with existing work, slightly different definitions will have to be employed in other parts of this thesis.

3.1.3 DG Weak Form

The actual choice of numerical fluxes is where the various DG FEM methods differ, and it is an important design decision, since the fluxes determine important properties like consistency, symmetry, and stability of the finite element method. In the following, only two particular choices of fluxes will be presented, along with a discussion of their consequences. For an in-depth discussion of different fluxes and their respective properties the reader is referred to Arnold et al. [2001].

BZ Method. Since fluxes are responsible for weakly enforcing inter-element continuity, i.e., for "gluing" neighboring elements, a straightforward approach is to penalize the squared jump $(u^- - u^+)^2$. This corresponds to the method of Babuška and Zlámal [1973], denoted by BZ, which employs the fluxes

$$\hat{u} := u|_K, \quad \hat{\sigma} := -\eta_e [\![u]\!].$$

η_e is a penalty term that can assume different values for each edge e. Inserting the fluxes into (3.13) and simplifying the resulting equations by exploiting the identities

$$\{\{\cdot\}\} = \{\cdot\}, \quad \{[\![\cdot]\!]\} = [\![\cdot]\!], \quad [\![\{\cdot\}]\!] = [\![[\![\cdot]\!]]\!] = 0,$$

leads to the weak form of the BZ method

$$\begin{aligned} a_{\mathrm{BZ}}(u,v) &:= \int_\Omega \nabla u \cdot \nabla v + \boxed{\int_\Gamma \eta_e [\![u]\!] \cdot [\![v]\!]} \\ &= \int_\Omega f v, \end{aligned} \tag{3.14}$$

which differs from the CG weak form (3.3) in the framed penalty term. This term is weighted by a scalar function $\eta_e = \eta \, \|e\|^{-1}$ inversely proportional to the edge length $\|e\|$. Analyzing the internal energy

$$a_{\mathrm{BZ}}(u,u) = \int_\Omega \nabla u \cdot \nabla u + \int_\Gamma \eta_e [\![u]\!] \cdot [\![u]\!]$$

reveals that the BZ method in fact penalizes the squared jump $[\![u]\!] \cdot [\![u]\!] = (u^- - u^+)^2$.

Just as in CG FEM, approximating u and v by shape functions N_i as in (3.4) leads to a linear system equivalent to (3.5), with matrix entries determined by $K_{ij} = a_{\mathrm{BZ}}(N_i, N_j)$. The important difference is the contribution of edges, i.e., the framed integral over Γ in (3.14). Note that for *continuous* functions

u and v, as in the case of CG FEM, this integral would vanish, since then $[\![u]\!] = [\![v]\!] = 0$ and $v = 0$ on $\partial\Omega$, thereby reproducing the CG weak form (3.3).

The BZ method is geometrically intuitive and easy to implement. Moreover, it is *stable* in the sense that the stiffness matrix $\mathbf{K}$ is positive definite for any $\eta > 0$. However, as detailed in Arnold et al. [2001], the method is *not consistent*: A continuous solution u of the problem might not satisfy the BZ weak form (3.14). Consequently, the approximate solution u does in general not converge toward the exact solution under element refinement.

Consistency Example. As a simple example showing that the BZ method is not consistent, consider the continuous solution $u = x^2$ and the resulting forcing term $-\Delta u = f = -2$. v can be chosen such that it is equal to one inside an internal element K^1 with positive area A_{K^1}, and vanishes inside all other elements. As a result, $\nabla v = 0$ inside each element and the first term in (3.14) vanishes. The second term vanishes because u is continuous and thus $[\![u]\!] = 0$, resulting in $a_{\mathrm{BZ}} = 0$. However, the right-hand side of (3.14) is equal to $\int_{K^1} f = -2A_{K^1} < 0$, and thus the weak form is not satisfied in this case.

IP Method. A more accurate alternative to the BZ method is the so-called *interior penalty* (IP) method [Douglas and Dupont, 1976], whose fluxes are defined as

$$\hat{u} := \{u\}, \quad \hat{\sigma} := \{\nabla u\} - \eta_e [\![u]\!]. \tag{3.15}$$

Inserting them into (3.13) and simplifying terms yields the IP weak form

$$\begin{aligned} a_{\mathrm{IP}}(u,v) \; &:= \int_\Omega \nabla u \cdot \nabla v \\ &\boxed{- \int_\Gamma ([\![v]\!] \cdot \{\nabla u\} + [\![u]\!] \cdot \{\nabla v\} - \eta_e [\![u]\!] \cdot [\![v]\!])} \\ &= \int_\Omega f v. \end{aligned} \tag{3.16}$$

The IP method uses three individual penalty terms in the framed Γ-integral:

- The first term ensures *consistency*: Any continuous solution u of the problem (3.1) also satisfies the DG weak form (3.16).
- The second term achieves *symmetry* of the bilinear form $a_{\mathrm{IP}}(u,v)$, and thereby also of the stiffness matrix $\mathbf{K}$.
- The last term ensures *stability*: For a sufficiently large penalty η it guarantees $a_{\mathrm{IP}}(u,u) > 0$, i.e., $\mathbf{K}$ to be positive definite.

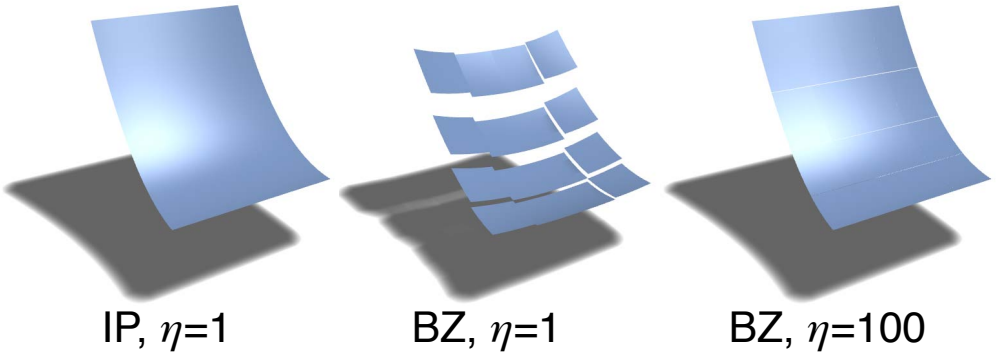

Figure 3.7: *Solution of $\Delta u = 2$, with Dirichlet boundary conditions corresponding to $u(x,y) = x^2$, using quadratic shape functions N_i on a 4×4 quad mesh. The* consistent *IP method finds the exact solution x^2 independently of the penalty η, since x^2 lies in the space of shape functions. This is not the case for the* inconsistent *BZ method, although increasing η improves the approximation by decreasing the jumps $[\![u]\!]$.*

The IP fluxes are one of few choices to yield a stable as well as consistent DG method. Consistency guarantees that if a *continuous* solution of either (3.3) or (3.16) exists in the space of shape functions N_i, then the IP method will find this solution as a function u with $[\![u]\!] = 0$ (see Fig. 3.7).

Consistency Example. In the previous section, we have shown that the BZ method is not consistent by applying it to a simple example using the continuous solution $u = x^2$ with forcing term $-\Delta u = f = -2$. v was chosen to be piecewise constant, assuming a value of 1 inside a particular element K^1 and 0 everywhere else. Applying the IP weak form to this example, we find that most of the terms in (3.16) vanish due to $\nabla v = 0$ and $[\![u]\!] = 0$, resulting in the equation

$$-\int_\Gamma [\![v]\!] \cdot \{\nabla u\} = \int_\Omega f v.$$

Applying the definitions of the jump and average operators, and writing f as $f = -\nabla \cdot \nabla u$, we get

$$-\int_\Gamma (v^- \mathbf{n}^- + v^+ \mathbf{n}^+) \cdot \nabla u = -\int_\Omega \nabla \cdot \nabla u \, v.$$

If we now take into account that $v = 1$ only inside element K^1, and $v = 0$ everywhere else, this simplifies to

$$\int_{\partial K^1} \mathbf{n} \cdot \nabla u = \int_{K^1} \nabla \cdot \nabla u.$$

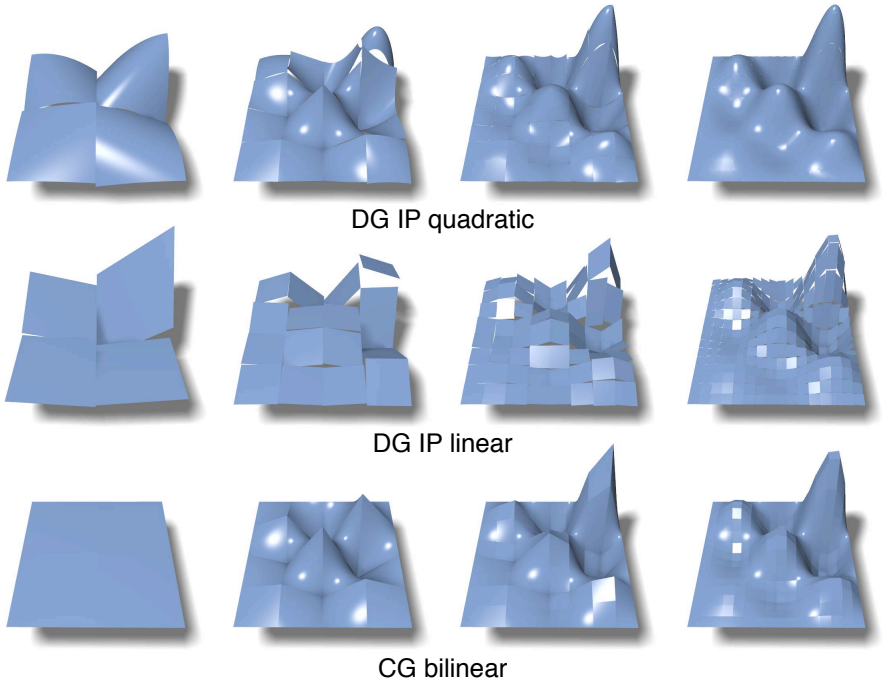

Figure 3.8: *Solution of the Poisson equation $-\Delta u = f$ on a regular quadrilateral grid of resolutions 2^2, 4^2, 8^2, and 16^2, using CG FEM and DG FEM.*

This equation, however, is nothing else but the divergence theorem applied to ∇u (see Eq. (B.4)), so it holds true and shows that the IP method is in fact consistent for this particular example.

Polynomial Basis Functions. Furthermore, due to consistency and stability the IP method converges under refinement towards the exact solution of the PDE, with a convergence rate determined by the degree of N_i [Arnold et al., 2001].

This leads to the main advantage of DG FEM: The missing conformity constraints allow simple polynomials $\{1, x, y, x^2, xy, \ldots, y^k\}$ of degree k to be used as basis functions for each element K. The convergence behavior of linear and quadratic DG basis functions is demonstrated in Fig. 3.8, which also shows bilinear CG FEM for comparison. As expected, the IP method converges regularly, at a rate similar to CG for linear shape functions, and at a faster rate for quadratic ones. By consequence, the jumps decrease under element refinement, eventually reconstructing the exact, continuous solution [Cockburn, 2003]. For the same number of DOFs and basis functions of the same degree, CG FEM can be observed to be more accurate than DG FEM by a constant factor.

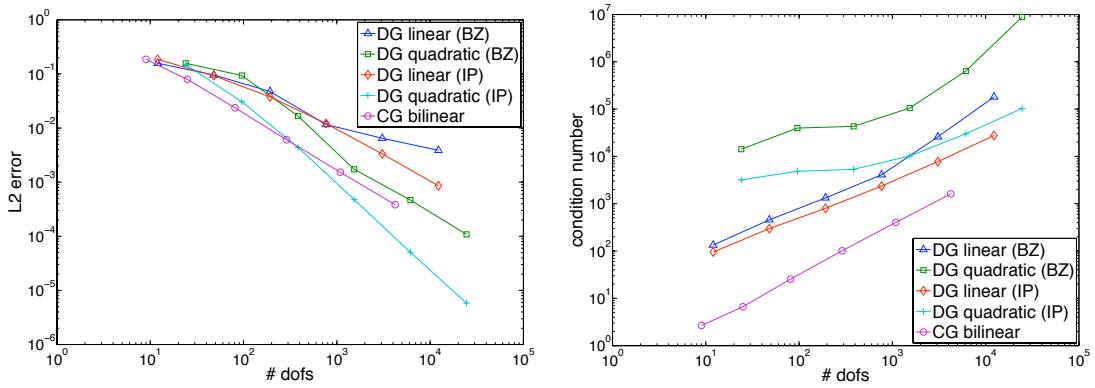

Figure 3.9: *Error plots for the solution of the Poisson equation on a regular grid. The plots compare the L^2 errors $\|\Delta u + f\|$ and the condition numbers of the stiffness matrix* $\mathbf{K}$ *for the BZ and IP method using linear and quadratic basis functions, and also include bilinear CG FEM as a reference.*

While the use of polynomial basis functions already simplifies simulations on regular 2D grids, the true value of this added flexibility will be demonstrated in Chapter 4, in the context of elasticity simulations on irregular 3D meshes of dynamically changing topology.

3.2 DG FEM For Non-Linear Problems

For the problems considered so far, the corresponding energy was always quadratic in u and the weak form linear in u. Discretizing the weak form then resulted in a linear system to be solved for the unknown degrees of freedom. However, many practical applications of the FEM such as the simulation of solids using accurate material models result in energy minimization problems that do not follow this simple form. This section shows how discontinuous Galerkin finite element methods can be derived for these kind of problems, following the derivations of Arnold et al. [2001] and Ten Eyck and Lew [2006].

Non-Linear Problem. In particular, we consider the problem of minimizing an energy functional of the form

$$E[u] = \int_\Omega W(\mathbf{x}, \nabla u)$$

where the *energy density function* W can non-linearly depend on the position $\mathbf{x}$ and the first derivatives of u. Note that the forcing term $\int_\Omega f u$ has been omitted for simplicity, but adding it back in would be trivial.

For the Poisson problem considered in the earlier derivations, the energy density would correspond to

$$W_{\text{Poisson}}(\mathbf{x}, \nabla u) = \frac{1}{2}\nabla u \cdot \nabla u,$$

so we can still recover this example as a special case of the non-linear problem. We will however treat W as an arbitrary smooth function of $\mathbf{x}$ and ∇u from now on.

3.2.1 DG Derivative

Applying the integration by parts theorem Eq. (B.3) to a sufficiently smooth scalar valued function u and a vector valued function $\mathbf{z}$ over element K gives

$$\int_K \nabla u \cdot \mathbf{z} = \int_{\partial K} u\, \mathbf{n}_K \cdot \mathbf{z} - \int_K u \nabla \cdot \mathbf{z} \tag{3.17}$$

with $\mathbf{n}_K$ denoting the outward unit normal of element K. This is sound as long as u and $\mathbf{z}$ are continuously differentiable on the closure of Ω. However, what we are actually interested in is the case where u is a bi-valued function on ∂K and the integral over ∂K is using the numerical flux $\hat{u}$, an approximation to the bi-valued function, instead of u. Replacing u by the numerical flux $\hat{u}$ on ∂K and replacing ∇u by the so-called *DG derivative* $D_{\text{DG}}u$ gives

$$\int_K D_{\text{DG}}u \cdot \mathbf{z} = \int_{\partial K} \hat{u}\, \mathbf{n}_K \cdot \mathbf{z} - \int_K u \nabla \cdot \mathbf{z}. \tag{3.18}$$

As will be shown later, given a numerical flux $\hat{u}$, we can find a DG derivative $D_{\text{DG}}u$ such that the above equation holds for all elements K. Note that if u is continuous over all of Ω and $\hat{u} = u$, we recover $D_{\text{DG}}u = \nabla u$. For discontinuous u, $D_{\text{DG}}u$ will be an alternative version of the gradient of u that takes into account the numerical fluxes on the element boundaries. We can then use $D_{\text{DG}}u$ in place of ∇u in the energy minimization functional to get

$$E[u] = \int_\Omega W(\mathbf{x}, D_{\text{DG}}u) = \sum_K \int_K W(\mathbf{x}, D_{\text{DG}}u). \tag{3.19}$$

This energy allows u to be discontinuous across elements, and after discretizing u using finite element basis functions, can be minimized using a Newton method, see Section 3.3.2. What remains is the computation of the DG derivative $D_{\text{DG}}u$ given numerical fluxes $\hat{u}$.

Finding the DG Derivative. Computing the sum of Eq. (3.18) over all elements K and making use of the definition of the jump and average operators as defined in (3.12) we get

$$\sum_K \int_K D_{\mathrm{DG}} u \cdot \mathbf{z} = \int_\Gamma [\![\hat{u}]\!] \cdot \{\mathbf{z}\} + \int_\Gamma \{\hat{u}\} [\![\mathbf{z}]\!] - \sum_K \int_K u \nabla \cdot \mathbf{z}.$$

Here, Γ is the union of all element edges and we assume the jump and average operators to be appropriately defined for edges on $\partial\Omega$. Performing the same summation for (3.17) gives

$$\sum_K \int_K \nabla u \cdot \mathbf{z} = \int_\Gamma [\![u]\!] \cdot \{\mathbf{z}\} + \int_\Gamma \{u\} [\![\mathbf{z}]\!] - \sum_K \int_K u \nabla \cdot \mathbf{z}.$$

Subtracting these two equations results in

$$\sum_K \int_K (D_{\mathrm{DG}} u - \nabla u) \cdot \mathbf{z} = \int_\Gamma [\![\hat{u} - u]\!] \cdot \{\mathbf{z}\} + \int_\Gamma \{\hat{u} - u\} [\![\mathbf{z}]\!].$$

Next, the linear *lifting operators* $\mathbf{R}$ and $\mathbf{L}$ are defined as

$$\int_\Omega \mathbf{R}(\mathbf{u}) \cdot \mathbf{z} = -\int_\Gamma \mathbf{u} \cdot \{\mathbf{z}\}$$

and

$$\int_\Omega \mathbf{L}(u) \cdot \mathbf{z} = -\int_\Gamma u [\![\mathbf{z}]\!].$$

The computation of these lifting operators will be shown when looking at the discretization of the problem. For now, it is sufficient to know that they allow us to convert edge integrals to area integrals and thus allow us to write the above equation as

$$\sum_K \int_K (D_{\mathrm{DG}} u - \nabla u - \mathbf{R}([\![u - \hat{u}]\!]) - \mathbf{L}(\{u - \hat{u}\})) \cdot \mathbf{z} = 0.$$

As $\mathbf{z}$ is arbitrary, this defines the DG derivative as

$$D_{\mathrm{DG}} u = \nabla u + \mathbf{R}([\![u - \hat{u}]\!]) + \mathbf{L}(\{u - \hat{u}\}).$$

Note that as mentioned earlier, it follows from this definition that $D_{\mathrm{DG}} u = \nabla u$ if u is continuous and $\hat{u} = u$. Also note that this can be trivially extended to vector fields $\mathbf{u}$, in which case $D_{\mathrm{DG}} \mathbf{u}$ is a two-tensor instead of a vector.

3.2.2 Choice of Fluxes

In the DG FEM derivation shown in Section 3.1.2, numerical fluxes had to be defined for both the function u and its gradient σ. However in the more generic derivation shown in this section, we only need to define the flux of u while derivatives are automatically taken care of by the lifting operators.

One common choice of flux [Lew et al., 2004; Bassi and Rebay, 1997; Brezzi et al., 2000] is to set

$$\hat{u} = \{u\}$$

on interior edges. The DG derivative then simplifies to

$$D_{\mathrm{DG}} u = \nabla u + \mathbf{R}(\llbracket u \rrbracket). \tag{3.20}$$

3.2.3 Discretization and Lifting Operators

Just as described in Section 3.1.2, the function u is discretized using per-element basis functions N_i that are sufficiently smooth within elements and assume non-zero values only within exactly one element:

$$u(\mathbf{x}) = \sum_{i=1}^{n} u_i N_i(\mathbf{x})$$

Recall that u_i are the degrees of freedom of the problem.

In order to discretize the lifting operator, we need to define an additional set of per-element basis functions L_j which serve as a basis for the derivatives ∇u. For example, when using linear per-element basis functions N_i, it is sufficient to have one constant derivative basis function L_K for each element K which assumes a value of 1 in element K and vanishes in all other elements.

With this discretization, the lifting operator $\mathbf{R}$ becomes a linear transformation between the derivative basis L_j and the basis N_i and can be expressed using a 3-tensor R_{adi} as

$$\mathbf{R}(\mathbf{n}u) = \mathbf{R}(\mathbf{n}\sum_a u_a N_a) = \sum_a u_a \mathbf{R}(\mathbf{n}N_a) = \sum_{a,d,i} u_a \mathbf{e}_i L_d R_{adi} \tag{3.21}$$

where $\mathbf{e}_i$ is the i-th standard basis vector. Recalling the definition of the lifting operator

$$\int_\Omega \mathbf{R}(\mathbf{u}) \cdot \mathbf{z} = -\int_\Gamma \mathbf{u} \cdot \{\mathbf{z}\}$$

and using $\mathbf{u} = \mathbf{n}N_a$ and $\mathbf{z} = \mathbf{e}_i L_f$ while discretizing $\mathbf{R}$ we get

$$\sum_d \left(\int_\Omega L_d L_f \right) R_{adi} = -\int_\Gamma N_a n_i \{L_f\}$$

where n_i is the i-th component of the outward unit normal $\mathbf{n}$. R_{adi} can now be computed by inverting the mass matrix with entries $M_{df} = \int_\Omega L_d L_f$. This inversion can be computed efficiently because of the element-wise basis functions, causing the mass matrix to be block-diagonal. Note that the coefficients R_{adi} only depend on the discretization but not on the degrees of freedom, so they can be precomputed and stored in the element for which L_d assumes non-zero-values.

The total energy Eq. (3.19) can now be computed in terms of degrees of freedom u_i using an appropriate definition of the DG derivative such as Eq. (3.20) and by discretizing the lifting operator $\mathbf{R}$ with Eq. (3.21) using the precomputed factors R_{adi}.

3.2.4 Stabilization

For some non-linear problems, the stability of the DG discretization can be improved by introducing a stabilization term [Ten Eyck and Lew, 2006]. One possible way of achieving this is by adding the following term (boxed) to the energy E

$$E[u] = \sum_K \int_K W(\mathbf{x}, D_{\mathrm{DG}} u) + \boxed{\frac{\beta}{h} \int_\Gamma [\![u - \hat{u}]\!] \cdot [\![u - \hat{u}]\!]}$$

where β is a positive penalty weight and h a measure of the mesh fineness.

3.3 Non-Linear Elasticity

One of the most common applications of FEM is the simulation of deformable objects. The corresponding partial differential equations can be derived from basic principles from continuum mechanics. In this section, the most simple case of a *hyperelastic* material is considered. A material is termed hyperelastic if the work done by the stresses during a deformation process only depends on the initial state of the material and its current state, independently of the deformation path [Bonet and Wood, 1997]. In particular, an object made out of a hyperelastic material will return to its undeformed rest state when the external forces and boundary conditions are removed (and if the dynamic simulation is damped).

3.3.1 Continuum Formulation

Deformation Energy. Consider a 3D object with material coordinates $\mathbf{X} = (X_1, X_2, X_3)^T \in \Omega \subset \mathbb{R}^3$ with a deformation field $\boldsymbol{\varphi} : \Omega \to \mathbb{R}^3$, mapping a point $\mathbf{X}$ to $\mathbf{x} = (x_1, x_2, x_3)^T = \boldsymbol{\varphi}(\mathbf{X})$. We are interested in finding the *deformation energy* E for a given deformation field $\boldsymbol{\varphi}$. To this end, a *deformation energy density* $\Psi[\boldsymbol{\varphi}](\mathbf{X})$ is defined, describing the deformation energy of an infinitesimal material volume at position $\mathbf{X}$. Note that we consider Ψ to be a functional at this point, so we do not have to specify yet how exactly it depends on $\boldsymbol{\varphi}$. The deformation energy density is integrated over the whole domain Ω to get the total deformation energy functional

$$E[\boldsymbol{\varphi}] = \int_\Omega \Psi[\boldsymbol{\varphi}](\mathbf{X}). \tag{3.22}$$

The object is at rest if E is in a local minimum with respect to the deformation field $\boldsymbol{\varphi}$. To solve this problem, we have to find a configuration $\boldsymbol{\varphi}$ such that the first variation of E vanishes, i.e. $\delta E[\boldsymbol{\varphi}] = 0$, given some boundary conditions.

Deformation Energy Densities. In order to define meaningful energy densities Ψ, and establish their dependency on $\boldsymbol{\varphi}$, we first need to introduce a couple of quantities describing the local deformation of an infinitesimal material volume. The 3×3 *deformation gradient* $\mathbf{F}$ with entries

$$(\mathbf{F}(\mathbf{X}))_{ij} = F_{ij}(\mathbf{X}) = \left.\frac{\partial \varphi_i}{\partial X_j}\right|_{\mathbf{X}}$$

measures the local (linearized) behavior of the deformation at position $\mathbf{X}$. In the following, the dependency on $\mathbf{X}$ will not be stated explicitly but is tacitly assumed. The symmetric *right Cauchy-Green tensor* $\mathbf{C}$ is defined as

$$\mathbf{C} = \mathbf{F}^T\mathbf{F}$$

and allows for the definition of the *Green strain tensor* $\mathbf{E}$

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}).$$

The deformation gradient $\mathbf{F}$ can be decomposed into a rotational component $\mathbf{R}$ and a stretch component $\mathbf{U}$ as $\mathbf{F} = \mathbf{R}\mathbf{U}$. The right Cauchy-Green tensor can now be written as $\mathbf{C} = \mathbf{U}^T\mathbf{R}^T\mathbf{R}\mathbf{U} = \mathbf{U}^T\mathbf{U}$. In other words, $\mathbf{C}$ (and consequently also $\mathbf{E}$) is invariant under rotations.

As the energy density Ψ only depends on the local (linearized) deformation of the material and the local material properties, it is a function of the deformation gradient $\mathbf{F}$ and the position $\mathbf{X}$. However, assuming homogeneous

material properties, and noting that for physical applications Ψ must be invariant under rigid body motions, it can be simplified to be a function of $\mathbf{C}$ only, i.e. $\Psi(\mathbf{C})$. Additionally assuming that the material behavior is isotropic, i.e. identical in any direction, Ψ will only depend on the three scalar invariants of the tensor $\mathbf{C}$ [Bonet and Wood, 1997]:

$$\begin{aligned} I_C &= \operatorname{tr} \mathbf{C} \\ II_C &= \operatorname{tr} \mathbf{C}\mathbf{C} = ||\mathbf{C}||^2 \\ III_C &= \det \mathbf{C} = J^2 \end{aligned}$$

where $J = \det \mathbf{F}$, so one can write $\Psi(I_C, II_C, III_C)$.

Stresses. In an equilibrium configuration, the variation of the deformation energy must vanish. Expressing this in terms of the Green strain gives

$$\delta E[\boldsymbol{\varphi}] = \int_\Omega \frac{\partial \Psi}{\partial \mathbf{E}} : \delta \mathbf{E}[\boldsymbol{\varphi}]. \tag{3.23}$$

The colon operator ":" denotes the tensor product between two matrices $\mathbf{A}$ and $\mathbf{B}$ or between a matrix $\mathbf{A}$ and a 4-tensor $\mathbf{C}$ as

$$\begin{aligned} \mathbf{A}:\mathbf{B} &:= \sum_{ij} \mathbf{A}_{ij}\mathbf{B}_{ij}, \\ \mathbf{A}:\mathbf{C} &:= \sum_{ij} \mathbf{A}_{ij}\mathbf{C}_{ijkl}, \\ \mathbf{C}:\mathbf{A} &:= \sum_{kl} \mathbf{C}_{ijkl}\mathbf{A}_{kl}. \end{aligned}$$

For energy densities that are expressed in terms of $\mathbf{C}$, the partial derivative with respect to $\mathbf{E}$ is trivial to compute as

$$\frac{\partial \Psi}{\partial E_{ij}} = \sum_{kl} \frac{\partial \Psi}{\partial C_{kl}} \frac{\partial C_{kl}}{\partial E_{ij}} = 2 \frac{\partial \Psi}{\partial C_{ij}}.$$

The term $\partial\Psi/\partial\mathbf{E}$ actually has a physical meaning: it corresponds to the *second Piola-Kirchhoff stress tensor* and is usually denoted by the symbol $\mathbf{S}$. When applied to an area element with *undeformed* unit normal vector $\mathbf{N}$ and *undeformed* area dA, it returns a non-physical force *in the reference coordinate system*. This can be expressed as

$$\mathbf{F}^{-1}\mathbf{f} = \mathbf{S}\mathbf{N}\,dA,$$

where $\mathbf{f}$ is the actual physical force acting on the area element. The pair $\mathbf{S}$, $\mathbf{E}$ is said to be *work conjugate*, because their product corresponds to the work

per volume. The definition of stresses (such as the second Piola-Kirchhoff stress $\mathbf{S}$) in terms of deformation measures (such as the right Cauchy-Green tensor $\mathbf{C}$) is known as the *constitutive equation* of the material.

Another useful quantity is the *first Piola-Kirchhoff stress* denoted by $\mathbf{P}$, which relates to the second Piola-Kirchhoff stress as $\mathbf{S} = \mathbf{F}^{-1}\mathbf{P}$. It returns the physical force when being applied to an *undeformed* area element:

$$\mathbf{f} = \mathbf{P}\mathbf{N}\,dA.$$

Finally, the *Cauchy stress* σ relates the area element with *deformed* unit normal vector $\mathbf{n}$ and *deformed* area da to the physical force $\mathbf{f}$:

$$\mathbf{f} = \sigma\,\mathbf{n}\,da.$$

Note that the first and second Piola-Kirchhoff stress and the Cauchy stress are merely different representations of the same physical stress, i.e. of a force acting on an area element. They only differ in whether the deformed or the undeformed area element is considered, and whether the force is represented in deformed or undeformed coordinates.

Constitutive Equations. One particularly simple material model is the *St. Venant-Kirchhoff material*, whose energy density per unit volume, expressed in $\mathbf{E}$, is

$$\Psi(\mathbf{E}) = \frac{1}{2}\lambda(\operatorname{tr}\mathbf{E})^2 + \mu\mathbf{E} : \mathbf{E}.$$

Computing the derivative with respect to $\mathbf{E}$ gives the corresponding second Piola-Kirchhoff stress

$$\mathbf{S} = \lambda(\operatorname{tr}\mathbf{E})\mathbf{I} + 2\mu\mathbf{E}, \tag{3.24}$$

with λ (Lamé's first parameter) and μ (shear modulus) denoting material parameters. These can be expressed in terms of the materials *Young's modulus* E, describing the stiffness of the material (not to be confused with the deformation energy E), and its *Poisson's ratio* ν, describing how well the material preserves its volume under stress (to first order in the strain):

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)},$$
$$\mu = \frac{E}{2(1+\nu)}.$$

Note that with the constitutive equation (3.24), the second Piola-Kirchhoff stress turns out to be linear in $\mathbf{E}$, a fact we will make use of when considering linear elasticity in the following section.

3.3.2 Discretization and Solution

To find a deformation field $\boldsymbol{\varphi}$ that minimizes the total deformation energy (3.22), $\boldsymbol{\varphi}$ first needs to be discretized as a linear combination of basis functions N_i:

$$\boldsymbol{\varphi}(\mathbf{X}) \approx \sum_{i=1}^{n} \mathbf{x}_i N_i(\mathbf{X}) \quad \text{with} \quad \mathbf{x}_i \in \mathbb{R}^3.$$

This defines the deformation field in terms of n unknown vector degrees of freedom $\mathbf{x}_1, \ldots, \mathbf{x}_n$. For given DOF values, the deformation gradient $\mathbf{F}$ can now be computed as

$$\mathbf{F}(\mathbf{X}) = \left.\frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{X}}\right|_{\mathbf{X}} = \sum_{i=1}^{n} \mathbf{x}_i \left(\left.\frac{\partial N_i}{\partial \mathbf{X}}\right|_{\mathbf{X}} \right)^T,$$

and from this the right Cauchy-Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ can be expressed in terms of the DOFs. Any deformation energy density depending only on $\mathbf{C}$ can now be expressed in terms of the DOFs as well. Introducing the vector $\mathbf{d} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T)^T = (d_1, \ldots, d_{3n})^T$ of length $3n$ containing all *scalar* DOFs, the deformation energy can be explicitly computed as

$$E(\mathbf{d}) = \int_\Omega \Psi(\mathbf{C}(\mathbf{X}, \mathbf{d})).$$

Non-Linear Optimization. To minimize this energy, a standard Newton-Raphson method can be employed [Nocedal and Wright, 2000]. Starting with an initial guess $\mathbf{d}$ for the DOFs, the increment $\Delta\mathbf{d}$ of $\mathbf{d}$ is found by solving the linear system

$$\mathbf{H}(\mathbf{d})\,\Delta\mathbf{d} = -\mathbf{f}(\mathbf{d}) \tag{3.25}$$

where the $3n$ force vector $\mathbf{f}$ contains the first derivatives of the energy E with respect to the DOFs as

$$(\mathbf{f}(\mathbf{d}))_i = \left.\frac{\partial E}{\partial d_i}\right|_{\mathbf{d}}$$

and the sparse $3n \times 3n$ Hessian matrix $\mathbf{H}$ contains the second derivatives of E:

$$(\mathbf{H}(\mathbf{d}))_{ij} = \left.\frac{\partial^2 E}{\partial d_i \partial d_j}\right|_{\mathbf{d}}.$$

After the increment $\Delta\mathbf{d}$ has been found, the DOFs are updated as

$$\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}.$$

A few important details need to be considered at this point. First, for the increment $\Delta\mathbf{d}$ to point in a direction of decreasing energy E, $\mathbf{H}$ needs to be

positive definite. As this may not always be the case when simulating elastic deformations, a simple workaround consists in adding a multiple of the identity matrix to $\mathbf{H}$, i.e. modifying $\mathbf{H}$ to

$$\mathbf{H} \leftarrow \mathbf{H} + \beta \mathbf{I}$$

where $\beta \geq 0$ is a small value, but large enough to render the updated $\mathbf{H}$ positive definite. In a practical implementation, one can use a direct solver based on a sparse sparse Cholesky factorization [Chen et al., 2008; Schenk et al., 2001] to find the update $\Delta\mathbf{d}$. If $\mathbf{H}$ is not positive definite, the factorization will fail and β needs to be increased. Adding a multiple of the identity to $\mathbf{H}$ can be interpreted as letting the solution $\Delta\mathbf{d}$ point more towards the direction of steepest descent, as for $\mathbf{H} = \mathbf{I}$, (3.25) would simplify to $\Delta\mathbf{d} = -\mathbf{f}(\mathbf{d})$.

Second, the increment $\Delta\mathbf{d}$, although pointing in a direction of decreasing energy E, may be too large, resulting in an *increase* in energy. To guarantee that a local minimum is found, the update should be changed to

$$\mathbf{d} \leftarrow \mathbf{d} + \alpha\Delta\mathbf{d}$$

where $\alpha \leq 1$ is chosen such that $E(\mathbf{d} + \alpha\Delta\mathbf{d}) < E(\mathbf{d})$. This can be achieved by starting with $\alpha = 1$ and halving α until the criterion is met.

In Algorithm 3.1, *LinSolve* solves a system of linear equations. *ApplyHardConstraints* modifies $\mathbf{H}$ and $\mathbf{f}$ such that for all constrained vector DOFs i, $\mathbf{f}_i = 0$, $\mathbf{H}_{ii} = \mathbf{I}$ and $\mathbf{H}_{ij} = \mathbf{H}_{ji} = \mathbf{0}$ for $j \neq i$. *ComputeHf* computes the first and second derivatives of E with respect to the DOFs, which similarly to *ComputeE* can be evaluated on a per-element basis when the algorithm is applied in the context of FEM problems.

3.4 Linear Elasticity

For small deformations, it suffices to consider a linearized version of elasticity, where the relationship between stress and strain is linear, and the strain is linear in the deformation. This simplification is well suited for computer graphics applications, because the resulting deformation behavior is already quite plausible. However, as the strain is linear in the deformation, it is no longer invariant under rotations, leading to artifacts whenever the object is rotated. To amend this, a so-called *corotational* approach is usually employed, which rotates the material elements back to their undeformed state before computing stresses. This will be further detailed in Chapter 4 and only the purely linear case is considered in this section.

```
1  Solve(d_init)
2    d ← d_init
3    do
4      E ← ComputeE(d)
5      H, f ← ComputeHf(d)
6      H, f ← ApplyHardConstraints(H, f)
7      Find small β ≥ 0 s.t. H + βI is pos. def.
8      Δd ← LinSolve((H + βI)Δd = −f)
9      Find large α ≤ 1 s.t. ComputeE(d + αΔd) < E
10     d += αΔd
11   while ||f||^2 > tol
12   return d
13 end
```

Algorithm 3.1: Newton-Raphson solver minimizing the energy E.

Similar derivations of CG FEM for linear elasticity can be found in many textbooks (e.g., Hughes [2000]) and also in the recent survey by Nealen et al. [2006].

3.4.1 Continuum Formulation

Material Linearity. Writing the variation of the deformation energy (3.23) in terms of the second Piola-Kirchhoff stress $\mathbf{S}$ gives

$$\delta E[\boldsymbol{\varphi}] = \int_\Omega \mathbf{S} : \delta\mathbf{E}[\boldsymbol{\varphi}].$$

Assuming the simple St. Venant-Kirchhoff material model (3.24), the second Piola-Kirchhoff stress is linear in $\mathbf{E}$ and can be written as

$$S_{ij} = \sum_{kl} \mathcal{C}_{ijkl} E_{kl}$$

where the elasticity 4-tensor $\mathcal{C}$ emerges with entries

$$\mathcal{C}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}). \tag{3.26}$$

If we further assume that the initial state of the object corresponds to a stress-free rest state, the deformation energy can be written as

$$E = \int_\Omega \mathbf{E} : \mathcal{C} : \mathbf{E},$$

which is quadratic in the Green strain $\mathbf{E}$. In the following, we will linearize $\mathbf{E}$ with respect to displacement DOFs, turning E into a quadratic function of the DOFs.

Geometric Linearity. Defining the *displacement field* $\mathbf{u}$ as

$$\mathbf{u}(\mathbf{X}) = \boldsymbol{\varphi}(\mathbf{X}) - \mathbf{X}$$

and linearizing the Green strain $\mathbf{E}$ in $\mathbf{u}$ results in the linear *Cauchy strain tensor* $\boldsymbol{\varepsilon}$ with entries

$$\varepsilon_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}).$$

Using vector notation, this is equivalent to

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right). \tag{3.27}$$

Quadratic Energy and Strong Form. Using the linearized strain, the second Piola-Kirchhoff stress then becomes $\mathbf{S} = \mathcal{C} : \boldsymbol{\epsilon}$ and the quadratic deformation energy is

$$E = \int_\Omega \boldsymbol{\epsilon} : \mathcal{C} : \boldsymbol{\epsilon}.$$

Assuming small deformations $\mathbf{u}$, the first and second Piola-Kirchhoff stress and the Cauchy stress are equivalent. In linear elasticity, one typically uses the symbol σ to denote the linearized stress, and one writes

$$\sigma(\mathbf{u}) \;=\; \mathcal{C} : \boldsymbol{\epsilon}(\mathbf{u}), \tag{3.28}$$

which is the definition of a *Hookean material*, where the Cauchy strain is linearly related to the stress through the symmetric 4-tensor $\mathcal{C}$.

Another way of deriving the deformation energy is by considering the static equilibrium between internal forces (due to linearized stresses) and external forces $\mathbf{f}$, which is expressed by

$$-\nabla \cdot \sigma(\mathbf{u}) \;=\; \mathbf{f}. \tag{3.29}$$

This equation constitutes the strong form of the linear elasticity problem and can be used as a starting point for finding the weak form and the deformation energy. Conversely, the strong form is nothing else but the Euler-Lagrange equation of the underlying energy minimization problem.

3.4.2 Discretization and Solution

Equations (3.28) and (3.29), in combination with suitable boundary constraints on $\partial\Omega$, constitute the strong form of elastostatics. Multiplying them by test

functions, integrating by parts over Ω, and combining the resulting equations yields the weak form of CG FEM:

$$a_{\mathrm{CG}}(\mathbf{u},\mathbf{v}) := \int_\Omega \boldsymbol{\epsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) \;=\; \int_\Omega \mathbf{f}\cdot\mathbf{v}, \tag{3.30}$$

Analogous to (3.4) and (3.5), discretizing $\mathbf{u}$ as

$$\mathbf{u}(\mathbf{X}) \;\approx\; \sum_{i=1}^{n} \mathbf{u}_i N_i(\mathbf{X}) \quad \text{with} \quad \mathbf{u}_i \in \mathbb{R}^3 \tag{3.31}$$

leads to a $3n \times 3n$ linear system $\mathbf{KU} = \mathbf{F}$ with

$$\begin{aligned} \mathbf{K}_{ij} &= a_{\mathrm{CG}}\big(N_i, N_j\big)\cdot \mathbf{I}_3 \;\in \mathbb{R}^{3\times 3}, \\ \mathbf{U}_i &= \mathbf{u}_i \;\in \mathbb{R}^3, \\ \mathbf{F}_i &= \int_\Omega \mathbf{f} N_i \;\in \mathbb{R}^3, \end{aligned}$$

where $\mathbf{I}_3$ denotes the 3×3 identity matrix.

3.5 Kirchhoff-Love Shell Mechanics

This section reviews the basic equations of the Kirchhoff-Love shell theory in order to establish the required notation and introduce quantities that will be referred to in subsequent chapters. A rather concise derivation of the thin-shell theory is given in Cirak et al. [2000], while the detailed derivations can be found in Simo and Fox [1989].

The Kirchhoff-Love theory for thin shells makes two main assumptions [Wempner and Talaslidis, 2003]:

- The height of the shell is assumed to be small compared to its overall size.
- A vector normal to the shell that is transformed according to the local deformation of the shell always remains normal to the shell and un-stretched.

The first assumption allows for first-order approximations in the direction normal to the shell, while the second assumption results in shearing deformations being neglected.

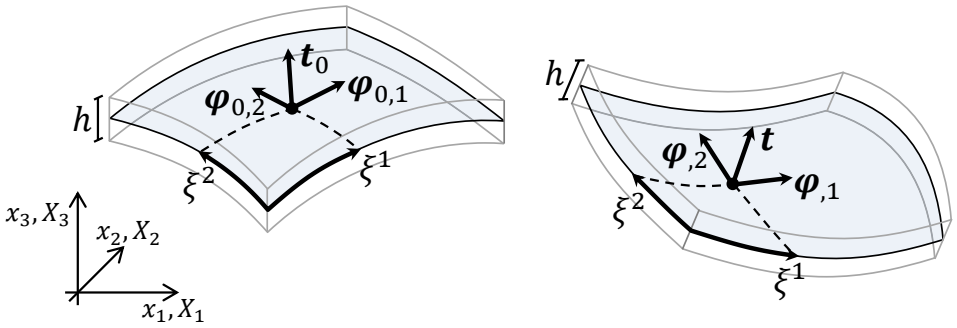

Figure 3.10: *Undeformed (left) and deformed (right) configurations of a shell with local coordinate systems at positions $\boldsymbol{\varphi}_0(\xi^1,\xi^2)$ and $\boldsymbol{\varphi}(\xi^1,\xi^2)$, respectively.*

3.5.1 Shell Geometry

The deformation of a thin shell can be fully described by the deformation of its mid-surface, which is a two-dimensional surface embedded in $\mathbb{R}^3$. The shell extends up to distance $h/2$ from the mid-surface, where h is the height of the shell. The mid-surface is parameterized using curvilinear coordinates $(\xi^1,\xi^2)^T \in \Omega \subset \mathbb{R}^2$, which allows us to define the current (deformed) configuration of the shell using a function $\boldsymbol{\varphi}(\xi^1,\xi^2) : \Omega \to \mathbb{R}^3$. Similarly, the initial (undeformed) configuration of the shell is defined by a function $\boldsymbol{\varphi}_0(\xi^1,\xi^2)$.

The surface basis vectors (tangent vectors) of the deformed and undeformed configuration, respectively, can be computed as $\boldsymbol{\varphi}_{,\alpha}(\xi^1,\xi^2)$ and $\boldsymbol{\varphi}_{0,\alpha}(\xi^1,\xi^2)$. Note that here and in the following, Greek indices indicate values 1 and 2, while a comma denotes partial differentiation. For example, $\boldsymbol{\varphi}_{,\alpha}(\xi^1,\xi^2) = \frac{\partial \boldsymbol{\varphi}(\xi^1,\xi^2)}{\partial \xi^\alpha}$. In the following, the explicit dependency of these quantities on (ξ^1,ξ^2) will be dropped, so we simply write $\boldsymbol{\varphi}$ instead of $\boldsymbol{\varphi}(\xi^1,\xi^2)$.

As any normal to the shell surface will always stay normal to the shell under deformation, the deformed surface normal vector $\mathbf{t}$ can be computed from the basis vectors as

$$\mathbf{t} = \frac{\boldsymbol{\varphi}_{,1} \times \boldsymbol{\varphi}_{,2}}{\|\boldsymbol{\varphi}_{,1} \times \boldsymbol{\varphi}_{,2}\|}$$

and similarly for the undeformed surface normal vector $\mathbf{t}_0$. The three vectors $\boldsymbol{\varphi}_{,1}$, $\boldsymbol{\varphi}_{,2}$ and $\mathbf{t}$ define a local coordinate system for a point $(\xi^1,\xi^2)^T$ on the mid-surface of the shell (see Fig. 3.10).

The deformed configuration can also be described relatively to the undeformed configuration, using a displacement field $\mathbf{u} : \Omega \to \mathbb{R}^3$ that defines the displacement of each point on the mid-surface. The deformed configuration can thus be written as

$$\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 + \mathbf{u}. \tag{3.32}$$

3.5.2 Shell Mechanics

The Kirchhoff-Love shell equations can be derived from the volumetric elasticity theory as follows: introducing a third coordinate ξ^3 extending into the normal direction, we can parameterize any position $\mathbf{X} \in \mathbb{R}^3$ relative to the mid-surface:

$$\mathbf{X}(\xi^1,\xi^2,\xi^3) = \boldsymbol{\varphi}_0(\xi^1,\xi^2) + \xi^3 \mathbf{t}_0(\xi^1,\xi^2)$$

and the corresponding deformed position is

$$\mathbf{x}(\xi^1,\xi^2,\xi^3) = \boldsymbol{\varphi}(\xi^1,\xi^2) + \xi^3 \mathbf{t}(\xi^1,\xi^2).$$

The partial derivatives of the deformed position are

$$\frac{\partial \mathbf{x}}{\partial \xi^1} = \boldsymbol{\varphi}_{,1} + \xi^3 \mathbf{t}_{,1} \qquad \frac{\partial \mathbf{x}}{\partial \xi^2} = \boldsymbol{\varphi}_{,2} + \xi^3 \mathbf{t}_{,2} \qquad \frac{\partial \mathbf{x}}{\partial \xi^3} = \mathbf{t},$$

and correspondingly for the undeformed position $\mathbf{X}$.

Recalling the definition of the Green strain from Section 3.3

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I})$$

and noting that the deformation gradient $\mathbf{F}$ can be written as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-1},$$

the Green strain $\mathbf{E}$ becomes

$$\begin{aligned} \mathbf{E} &= \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) = \frac{1}{2}\left(\left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-T} \left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \right)^{T} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-1} \right) - \mathbf{I} \\ &= \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-T} \mathbf{E}^{\xi} \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{-1} \end{aligned}$$

where

$$\mathbf{E}^{\xi} = \frac{1}{2}\left(\left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \right)^{T} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} - \left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right)^{T} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \right).$$

is the Green strain in the $(\xi^1,\xi^2,\xi^3)^T$ coordinate system. The products of partial derivatives of $\mathbf{x}$ and $\mathbf{X}$ with respect to $\boldsymbol{\xi}$ are the *metric tensors*, and $\mathbf{E}^{\xi}$ is thus proportional to the difference between the metric tensor of the undeformed and the deformed configuration.

To first order in the shell thickness direction ξ^3, the strain $\mathbf{E}^\xi$ decomposes into two tensors $\boldsymbol{\varepsilon}$ and $\boldsymbol{\rho}$:

$$E^\xi_{ij} = \varepsilon_{ij} + \xi^3 \rho_{ij}.$$

From the assumption that the normal $\mathbf{t}$ is always normal to $\boldsymbol{\varphi}_{,1}$ and $\boldsymbol{\varphi}_{,2}$ and $\mathbf{t} \cdot \mathbf{t} = 1$, it follows that $\varepsilon_{i3} = \varepsilon_{3i} = \rho_{i3} = \rho_{3i} = 0$. The Green strain $\mathbf{E}^\xi$ can thus be represented as a 2×2 matrix. Furthermore, using the identity $\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t}_{,\beta} = -\boldsymbol{\varphi}_{,\alpha\beta} \cdot \mathbf{t}$ following from $(\mathbf{t} \cdot \boldsymbol{\varphi}_{,\alpha})_{,\beta} = 0$, one can shown that $\boldsymbol{\varepsilon}$ corresponds to the difference between the *first fundamental form* of the undeformed and the deformed configuration, and $\boldsymbol{\rho}$ to the difference between the *second fundamental form* of the undeformed and the deformed configuration. The tensor $\boldsymbol{\varepsilon}$ only depends on the local basis vectors (i.e., first derivatives of $\boldsymbol{\varphi}$) and is known as the *stretching* or *membrane strain* tensor, while the tensor $\boldsymbol{\rho}$ is referred to as the *bending strain* tensor and also depends on *derivatives* of the basis vectors, thus measuring the local change in curvature.

Describing the deformed configuration $\boldsymbol{\varphi}$ in terms of the displacement field $\mathbf{u}$ according to (3.32), to first order in $\mathbf{u}$ the membrane strain two-tensor $\boldsymbol{\varepsilon}$ becomes

$$\varepsilon_{\alpha\beta}(\mathbf{u}) := \frac{1}{2}(\boldsymbol{\varphi}_{0,\alpha} \cdot \mathbf{u}_{,\beta} + \mathbf{u}_{,\alpha} \cdot \boldsymbol{\varphi}_{0,\beta}) \tag{3.33}$$

and the bending strain two-tensor $\boldsymbol{\rho}$ becomes

$$\begin{aligned} \rho_{\alpha\beta}(\mathbf{u}) \quad := \quad & \boldsymbol{\varphi}_{0,\alpha\beta} \cdot \mathbf{t}_0 \frac{1}{\bar{j}_0}(\mathbf{u}_{,1} \cdot (\boldsymbol{\varphi}_{0,2} \times \mathbf{t}_0) - \mathbf{u}_{,2} \cdot (\boldsymbol{\varphi}_{0,1} \times \mathbf{t}_0)) \\ & + \frac{1}{\bar{j}_0}(\mathbf{u}_{,1} \cdot (\boldsymbol{\varphi}_{0,\alpha\beta} \times \boldsymbol{\varphi}_{0,2}) - \mathbf{u}_{,2} \cdot (\boldsymbol{\varphi}_{0,\alpha\beta} \times \boldsymbol{\varphi}_{0,1})) \\ & - \mathbf{u}_{,\alpha\beta} \cdot \mathbf{t}_0. \end{aligned} \tag{3.34}$$

In the above equation, $\bar{j}_0$ is the determinant of the Jacobian of the undeformed mid-surface, defined as

$$\bar{j}_0 := \|\boldsymbol{\varphi}_{0,1} \times \boldsymbol{\varphi}_{0,2}\|. \tag{3.35}$$

Note that quantities such as $\boldsymbol{\varphi}_0$, $\mathbf{t}_0$ and $\bar{j}_0$ depend on the undeformed configuration only and can thus be precomputed for any position $(\xi^1, \xi^2)^T$.

Computing stresses from strains using a linear constitutive law, a formulation for the deformation energy can be found, which, by applying the variational principle, leads to the weak formulation

$$a(\mathbf{u}, \mathbf{v}) = f(\mathbf{v}),$$

which must hold for all test functions $\mathbf{v} : \Omega \to \mathbb{R}^3$. The right-hand side f represents external forces, and the weak form $a(\mathbf{u}, \mathbf{v})$ is defined as

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &:= \int_\Omega \epsilon_{\alpha\beta}(\mathbf{v}) \mathcal{H}_n^{\alpha\beta\gamma\delta} \epsilon_{\gamma\delta}(\mathbf{u}) \, \mathrm{d}\mathcal{S} \\ &\quad + \int_\Omega \rho_{\alpha\beta}(\mathbf{v}) \mathcal{H}_m^{\alpha\beta\gamma\delta} \rho_{\gamma\delta}(\mathbf{u}) \, \mathrm{d}\mathcal{S}. \end{aligned} \tag{3.36}$$

Note that Einstein sum notation is in effect here, meaning that repeated indices are summed over.

In the above weak form, $\mathcal{H}_n$ and $\mathcal{H}_m$ define the constitutive relations between stress and strain. They are defined as

$$\mathcal{H}_n^{\alpha\beta\gamma\delta} := \frac{Eh}{1-\nu^2} \mathcal{H}^{\alpha\beta\gamma\delta}, \tag{3.37}$$

$$\mathcal{H}_m^{\alpha\beta\gamma\delta} := \frac{Eh^3}{12(1-\nu^2)} \mathcal{H}^{\alpha\beta\gamma\delta}, \tag{3.38}$$

where

$$\begin{aligned} \mathcal{H} := \quad & \nu(\boldsymbol{\varphi}_0^{,\alpha} \cdot \boldsymbol{\varphi}_0^{,\beta})(\boldsymbol{\varphi}_0^{,\gamma} \cdot \boldsymbol{\varphi}_0^{,\delta}) \\ & + \frac{1}{2}(1-\nu)(\boldsymbol{\varphi}_0^{,\alpha} \cdot \boldsymbol{\varphi}_0^{,\gamma})(\boldsymbol{\varphi}_0^{,\delta} \cdot \boldsymbol{\varphi}_0^{,\beta}) \\ & + \frac{1}{2}(1-\nu)(\boldsymbol{\varphi}_0^{,\alpha} \cdot \boldsymbol{\varphi}_0^{,\delta})(\boldsymbol{\varphi}_0^{,\gamma} \cdot \boldsymbol{\varphi}_0^{,\beta}). \end{aligned}$$

$\boldsymbol{\varphi}_0^{,1}$ and $\boldsymbol{\varphi}_0^{,2}$ denote the *contravariant basis vectors*. Defining the symmetric *metric tensor* $\mathbf{G}$ as

$$\mathbf{G} := \begin{bmatrix} \boldsymbol{\varphi}_{0,1} \cdot \boldsymbol{\varphi}_{0,1} & \boldsymbol{\varphi}_{0,1} \cdot \boldsymbol{\varphi}_{0,2} \\ \boldsymbol{\varphi}_{0,2} \cdot \boldsymbol{\varphi}_{0,1} & \boldsymbol{\varphi}_{0,2} \cdot \boldsymbol{\varphi}_{0,2} \end{bmatrix},$$

the contravariant basis vectors can be computed as

$$\begin{bmatrix} | & | \\ \boldsymbol{\varphi}_0^{,1} & \boldsymbol{\varphi}_0^{,2} \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \boldsymbol{\varphi}_{0,1} & \boldsymbol{\varphi}_{0,2} \\ | & | \end{bmatrix} \mathbf{G}^{-T}. \tag{3.39}$$

Note that the constitutive relations (3.37) and (3.38) also depend on the Young's modulus E and Poisson's ratio ν of the material.

3.6 Outlook

After introducing the basics of PDEs and their related representations, this chapter presented the fundamental concepts of FEM and DG FEM, for linear

as well as non-linear problems. It also discussed the basics of linear and non-linear elasticity and a physically sound specialization to linear shell mechanics.

The following chapters will combine these theories, showing how DG FEM can be applied in the context of linear and non-linear elasticity problems as well as linear corotational shell simulations. In each of these areas, the focus will be on different features of the DG FEM, and different aspects of DG FEM will be taken advantage of to come up with solutions that improve over the state of the art.

CHAPTER 4

Simulation of Deformable Solids

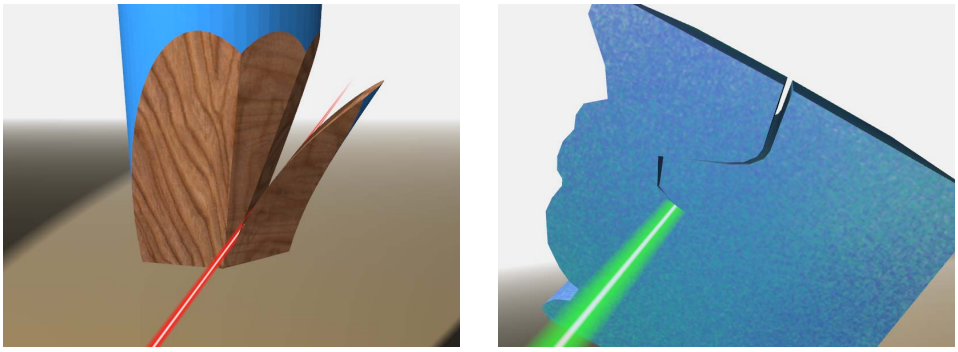

Figure 4.1: *A discontinuous Galerkin-based simulation method for deformable solids allows for the simulation of arbitrary polyhedral elements, enabling topology changes without remeshing. Sharpening a pencil consisting of a single convex element (left). Cutting a bunny out of a cube (right).*

In this chapter, a simulation technique for elastically deformable objects based on the discontinuous Galerkin finite element method is proposed. In contrast to traditional FEM, it overcomes the restrictions of conforming basis functions by allowing for discontinuous elements with weakly enforced continuity constraints. This added flexibility enables the simulation of arbitrarily shaped, convex and non-convex polyhedral elements, while still using simple polynomial basis functions. For the accurate strain integration over these elements,

an analytic technique based on the divergence theorem is presented. Being able to handle arbitrary elements eventually allows for simple and efficient techniques for volumetric mesh generation, adaptive mesh refinement, and robust cutting. Furthermore, the presented DG method does not to suffer from locking artifacts even for nearly incompressible materials, a problem that in standard FEM requires special handling.

After having introduced the main concepts of DG FEM based on a simple 2D Poisson problem in Section 3.1, this chapter shows how to derive equations and techniques for 3D linear elasticity (Section 4.2). DG FEM is extended further by directly simulating arbitrary polyhedra (Section 4.3), by generalizing stiffness warping to DG FEM (Section 4.4), and by using embedded simulation (Section 4.5) with suitable collision handling (Section 4.6). Equipped with those techniques, the versatility of the presented framework is demonstrated on a set of different applications in Section 4.7.

4.1 Overview

In computer graphics, FEM simulations of deformable objects are usually based on simple tetrahedral or hexahedral meshes. To simulate topological changes due to cutting or fracturing of material, one has to split elements and perform a remeshing step in order to avoid ill-shaped elements and to maintain a reasonable mesh quality. Due to the relaxed conformity constraints, a DG FEM-based approach allows for arbitrary convex and non-convex polyhedral elements, which greatly simplifies the mesh restructuring step. In the context of cutting and fracturing the approach presented in this work is most similar to Wicke et al. [2007] and Martin et al. [2008], but it is more flexible and more efficient due to the use of simple polynomial shape functions.

Furthermore, in order to support flexible simulations of deformable models for computer graphics applications, this work extends DG FEM by the following components:

- The simulation of arbitrary polyhedral elements using simple and efficient polynomial basis functions and a fast and accurate volumetric integration technique (Section 4.3).
- A generalization of stiffness warping to discontinuous polyhedral elements, thereby allowing linear strain measures to be used even in the presence of large deformations (Section 4.4).
- For embedded simulations: the reconstruction of a smooth displacement field from the discontinuous mesh, based on moving least

squares (MLS) interpolation (Section 4.5), along with a suitable collision handling technique (Section 4.6).

The versatility of the presented approach is demonstrated on several examples, including slicing-based mesh generation, adaptive stress-based element refinement, flexible and efficient cutting, and locking analysis (Section 4.7).

4.2 Linear Elasticity using DG FEM

The basic concepts of DG FEM have been introduced on the 2D Poisson problem in Section 3.1.2. Based on this introduction, this section shows how the same approach can be applied to the simulation of 3D linearly elastic deformations as presented in Section 3.4.

4.2.1 DG Weak Form

The derivation of the DG weak form for linear elasticity closely follows the procedure presented in Sections 3.1.2 and 3.1.3, where the derivation of a DG method for the 2D Poisson problem has been shown. Equations (3.28) and (3.29), which constitute the strong form of elastostatics, are multiplied by test functions and integrated over each element K, yielding the individual elements' weak forms. Those are summed up, fluxes $\hat{\mathbf{u}}$ and $\hat{\sigma}$ are introduced, and the two resulting equations are combined into one. The resulting equation corresponds to (3.11) for the Poisson problem, and is to be simplified using the average and jump operators.

Those, however, have to be slightly redefined for vector-valued functions $\mathbf{u}$ and matrix-valued functions σ on a face f shared by two elements K^- and K^+, such that the jump operator maps vectors to matrices and vice versa. Using the outer product $\mathbf{u} \otimes \mathbf{n} := \mathbf{u}\mathbf{n}^T$ we define

$$\{\mathbf{u}\} := \frac{1}{2}\left(\mathbf{u}^- + \mathbf{u}^+\right), \qquad [\![\mathbf{u}]\!] := \mathbf{u}^- \otimes \mathbf{n}^- + \mathbf{u}^+ \otimes \mathbf{n}^+,$$
$$\{\sigma\} := \frac{1}{2}\left(\sigma^- + \sigma^+\right), \qquad [\![\sigma]\!] := \sigma^-\mathbf{n}^- + \sigma^+\mathbf{n}^+.$$

BZ Method. Minimizing the jump $[\![\mathbf{u}]\!] : [\![\mathbf{u}]\!] = \|\mathbf{u}^- - \mathbf{u}^+\|^2$ by choosing the fluxes of Babuška and Zlámal [1973] leads to the weak form of the BZ method, which uses a_{BZ} instead of a_{CG} in (3.30):

$$a_{\mathrm{BZ}}(\mathbf{u},\mathbf{v}) := \int_\Omega \boldsymbol{\epsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) + \int_\Gamma \eta_f \, [\![\mathbf{u}]\!] : [\![\mathbf{v}]\!]. \tag{4.1}$$

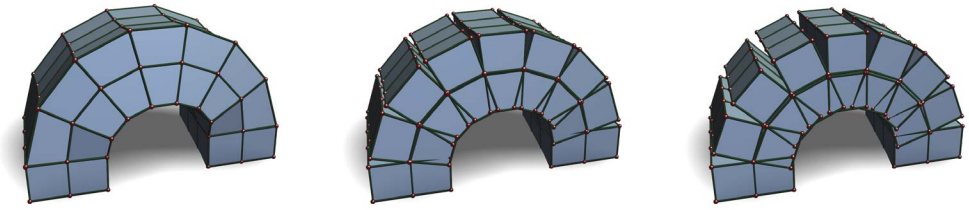

Figure 4.2: *Comparison between CG FEM (left), DG FEM (center), and the elastically coupled rigid cells of PriMo (right). The DG method conceptually spans the whole space from CG to PriMo, since for sufficiently large penalties η it approximates the CG results, and for an extremely stiff material and lower penalty η it reproduces the rigid cells of PriMo.*

The penalty parameter η_f is defined per face f according to Hansbo and Larson [2002]:

$$\eta_f \;=\; \eta \cdot \text{area}(f) \cdot \left(\frac{1}{\text{vol}(K^-)} + \frac{1}{\text{vol}(K^+)} \right) \tag{4.2}$$

using a global penalty parameter $\eta > 0$, which was typically in the order of 10^1–10^2 for the simulations shown in this work.

The internal elastic energy of the deformed object can then be written as

$$a_{\text{BZ}}(\mathbf{u},\mathbf{u}) = \int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{u}) + \int_\Gamma \eta_f \left\| \mathbf{u}^- - \mathbf{u}^+ \right\|^2,$$

which reveals an interesting connection to both CG FEM and the elastically coupled *rigid* cells of PriMo [Botsch et al., 2006]: CG computes elastic energies *within* elements only, using the Ω-integral, whereas PriMo employs only the "glue" energy *between* elements, represented by the Γ-integral.

Since BZ is based on *both* energy terms, with properly chosen penalty weight and material stiffness it can reproduce both methods, and can hence be considered as their generalization (see Fig. 4.2). As such, it combines the strengths of both approaches, since it inherits the physical accuracy of CG FEM, as well as the flexibility in element shapes and meshing of PriMo [Botsch et al., 2007], as will be demonstrated in Section 4.3.

The BZ penalty term is equivalent to both the glue energy of PriMo [Botsch et al., 2006] and the soft bindings employed by Sifakis et al. [2007b]. However, as already discussed in Section 3.1.3 and shown in Fig. 3.7, the BZ method is not *consistent* and therefore does not provide any convergence guarantees. Our experiments have nevertheless shown the BZ method to be very well suited for applications aiming at *physically plausible* deformations only.

IP Method. However, if physical accuracy is important, other DG fluxes, such as those of the IP method, should be chosen instead. For linear elasticity, the weak form of the IP method [Douglas and Dupont, 1976] is defined by

$$\begin{aligned} a_{\mathrm{IP}}(\mathbf{u},\mathbf{v}) := & \int_\Omega \boldsymbol{\epsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) \\ & - \int_\Gamma \big([\![\mathbf{v}]\!] : \{\boldsymbol{\sigma}(\mathbf{u})\} \\ & \quad + [\![\mathbf{u}]\!] : \{\boldsymbol{\sigma}(\mathbf{v})\} - \eta_f [\![\mathbf{u}]\!] : [\![\mathbf{v}]\!] \big). \end{aligned} \tag{4.3}$$

A detailed derivation of this weak form can be found in Section A.1.

Analogous to the Poisson problem (3.16), the three penalty terms ensure consistency, symmetry, and stability, and the method is guaranteed to converge under element refinement. Moreover, the IP method is still relatively easy to implement (see Section 4.2.2). While other (more complex) numerical fluxes exist (e.g., Ten Eyck and Lew [2006], Wihler [2006]), for our applications the BZ and IP methods performed very well and have been fully sufficient.

4.2.2 Discretization and Matrix Assembly

Discretization. To implement DG FEM for linear elasticity, we discretize $\mathbf{u}$ and $\mathbf{v}$ using n basis functions and set up the stiffness matrix $\mathbf{K}$:

$$\mathbf{u}(\mathbf{x}) \approx \sum_{i=1}^{n} \mathbf{u}_i N_i(\mathbf{x}) \quad \text{with} \quad \mathbf{u}_i \in \mathbb{R}^3, \tag{4.4}$$

$$\mathbf{v}(\mathbf{x}) \approx \sum_{i=1}^{n} \mathbf{v}_i N_i(\mathbf{x}) \quad \text{with} \quad \mathbf{v}_i \in \mathbb{R}^3. \tag{4.5}$$

Note that in this context, $\mathbf{x} \in \Omega \subset \mathbb{R}^3$ denotes a position in the undeformed configuration. Since this is very similar to CG FEM, the reader is referred to Hughes [2000] and Nealen et al. [2006] for more details on the following derivations.

Using a $3 \times 3n$ interpolation matrix $\mathbf{H}(\mathbf{x})$ built from the basis functions $N_i(\mathbf{x})$

$$\mathbf{H}(\mathbf{x}) = \left[\begin{array}{ccc|ccc|c} N_1(\mathbf{x}) & 0 & 0 & N_2(\mathbf{x}) & 0 & 0 & \\ 0 & N_1(\mathbf{x}) & 0 & 0 & N_2(\mathbf{x}) & 0 & \dots \\ 0 & 0 & N_1(\mathbf{x}) & 0 & 0 & N_2(\mathbf{x}) & \end{array}\right]$$

and a $3n$ vector $\mathbf{U}$ containing the unknown coefficients $\mathbf{u}_i \in \mathbb{R}^3$, the discretization (4.4) of $\mathbf{u}$ can be written in matrix notation as $\mathbf{u}(\mathbf{x}) = \mathbf{H}(\mathbf{x})\,\mathbf{U}$. Equivalently, the test function $\mathbf{v}$ can be represented as $\mathbf{v}(\mathbf{x}) = \mathbf{H}(\mathbf{x})\,\mathbf{V}$.

Moreover, using Voigt notation, stress and strain can be represented by 6D vectors $\bar{\sigma}$ and $\bar{\epsilon}$ composed of the independent entries of the symmetric 3×3 matrices σ and ϵ (defined in Equations (3.27) and (3.28)), respectively:

$$\begin{aligned} \bar{\sigma}(\mathbf{u}) &= (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{23}, \sigma_{13}, \sigma_{12})^T, \\ \bar{\epsilon}(\mathbf{u}) &= (\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, 2\epsilon_{23}, 2\epsilon_{13}, 2\epsilon_{12})^T. \end{aligned}$$

This leads to the matrix notation of the linear stress-strain relationship

$$\bar{\sigma}(\mathbf{u}(\mathbf{x})) = \bar{\mathbf{C}}\bar{\epsilon}(\mathbf{u}(\mathbf{x})) = \bar{\mathbf{C}}\mathbf{B}(\mathbf{x})\mathbf{U}, \tag{4.6}$$

with a constant symmetric 6×6 matrix $\bar{\mathbf{C}}$ built from the coefficients of the constitutive tensor $\mathcal{C}$ (see Eq. (3.26)),

$$\bar{\mathbf{C}} = \begin{bmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix},$$

and a $6 \times 3n$ matrix $\mathbf{B}(\mathbf{x})$ containing first derivatives of the basis functions N_i:

$$\mathbf{B}(\mathbf{x}) = \left[\begin{array}{ccc|ccc|c} N_{1,1}(\mathbf{x}) & 0 & 0 & N_{2,1}(\mathbf{x}) & 0 & 0 & \\ 0 & N_{1,2}(\mathbf{x}) & 0 & 0 & N_{2,2}(\mathbf{x}) & 0 & \\ 0 & 0 & N_{1,3}(\mathbf{x}) & 0 & 0 & N_{2,3}(\mathbf{x}) & \dots \\ 0 & N_{1,3}(\mathbf{x}) & N_{1,2}(\mathbf{x}) & 0 & N_{2,3}(\mathbf{x}) & N_{2,2}(\mathbf{x}) & \\ N_{1,3}(\mathbf{x}) & 0 & N_{1,1}(\mathbf{x}) & N_{2,3}(\mathbf{x}) & 0 & N_{2,1}(\mathbf{x}) & \\ N_{1,2}(\mathbf{x}) & N_{1,1}(\mathbf{x}) & 0 & N_{2,2}(\mathbf{x}) & N_{2,1}(\mathbf{x}) & 0 & \end{array}\right]$$

For the assembly of the stiffness matrix we use the above matrix notations to write the IP weak form (4.3) in terms of *element contributions* (Ω-integrals) and *face contributions* (Γ-integrals). Note that for the BZ method (4.1) only the last of the three face contributions in (4.3) is needed.

The element contributions are written in terms of element stiffness matrices $\mathbf{K}_K$ as in CG FEM:

$$\begin{aligned} \int_\Omega \epsilon(\mathbf{v}) : \mathbf{C} : \epsilon(\mathbf{u}) &= \sum_{K \in \mathcal{T}} \mathbf{V}^T \mathbf{K}_K \mathbf{U} \\ \text{with} \quad \mathbf{K}_K &= \int_K \mathbf{B}^T(\mathbf{x}) \bar{\mathbf{C}} \mathbf{B}(\mathbf{x}). \end{aligned} \tag{4.7}$$

After expanding the operators $\{\cdot\}$ and $[\![\cdot]\!]$, and exploiting $\mathbf{n}^- = -\mathbf{n}^+$, the first two face contributions of $f = K^- \cap K^+$ have the form

$$[\![\mathbf{v}]\!] : \{\sigma(\mathbf{u})\} \;=\; \left((\mathbf{v}^+ - \mathbf{v}^-) \otimes \mathbf{n}^+\right) : \frac{1}{2}\left(\sigma^-(\mathbf{u}) + \sigma^+(\mathbf{u})\right).$$

To write this in matrix notation, we need a 6×3 matrix

$$\mathbf{N}_f := \begin{bmatrix} n_1^+ & 0 & 0 & 0 & n_3^+ & n_2^+ \\ 0 & n_2^+ & 0 & n_3^+ & 0 & n_1^+ \\ 0 & 0 & n_3^+ & n_2^+ & n_1^+ & 0 \end{bmatrix}^T,$$

and differences and averages of matrices $\mathbf{B}$ and $\mathbf{H}$

$$\mathbf{H}_f^{[\![]\!]} := \left(\mathbf{H}_f^+ - \mathbf{H}_f^-\right), \quad \mathbf{B}_f^{\{\}} := \frac{1}{2}\left(\mathbf{B}_f^+ + \mathbf{B}_f^-\right),$$

which themselves are defined through restrictions

$$\mathbf{B}_f^{\pm} := \mathbf{B}|_{K^\pm}, \quad \mathbf{H}_f^{\pm} := \mathbf{H}|_{K^\pm},$$

containing only the entries of $\mathbf{B}$ or $\mathbf{H}$ corresponding to basis functions of $K^\pm$.

With these matrices we can define three stiffness matrices $\mathbf{K}_{f1}$, $\mathbf{K}_{f2}$, and $\mathbf{K}_{f3}$ for each face f:

$$\begin{aligned} \mathbf{K}_{f1} &= \int_f -{\mathbf{H}_f^{[\![]\!]}}^T \mathbf{N}_f^T \bar{\mathbf{C}} \mathbf{B}_f^{\{\}}, \\ \mathbf{K}_{f2} &= \int_f -{\mathbf{B}_f^{\{\}}}^T \bar{\mathbf{C}} \mathbf{N}_f \mathbf{H}_f^{[\![]\!]}, \\ \mathbf{K}_{f3} &= \int_f \eta_f {\mathbf{H}_f^{[\![]\!]}}^T \mathbf{H}_f^{[\![]\!]}. \end{aligned}$$

The three face contributions in (4.3) can now be written in terms of these face stiffness matrices as

$$-\int_\Gamma [\![\mathbf{v}]\!] : \{\sigma(\mathbf{u})\} \;=\; \sum_{f \in \mathcal{T}} \mathbf{V}^T \mathbf{K}_{f1} \mathbf{U}, \tag{4.8}$$

$$-\int_\Gamma [\![\mathbf{u}]\!] : \{\sigma(\mathbf{v})\} \;=\; \sum_{f \in \mathcal{T}} \mathbf{V}^T \mathbf{K}_{f2} \mathbf{U}, \tag{4.9}$$

$$\int_\Gamma \eta_f [\![\mathbf{u}]\!] : [\![\mathbf{v}]\!] \;=\; \sum_{f \in \mathcal{T}} \mathbf{V}^T \mathbf{K}_{f3} \mathbf{U}. \tag{4.10}$$

The global $3n \times 3n$ stiffness matrix $\mathbf{K}$ can therefore be assembled by doing one pass over all elements $K \in \mathcal{T}$ and accumulating their contributions $\mathbf{K}_K$,

and a second pass over all faces $f \in \mathcal{T}$ that accumulates their contributions $\mathbf{K}_{fi}$. Equivalently to CG, the external force vector $\mathbf{F}$ is assembled from the elements' contributions $\int_K \mathbf{H}(\mathbf{x})^T \mathbf{f}$. Note that even for linear basis functions the integrands $\mathbf{H}(\mathbf{x})$ are not constant, requiring integration techniques as discussed in Section 4.3. The discretized weak form

$$\mathbf{V}^T\mathbf{K}\mathbf{U} = \mathbf{V}^T\mathbf{F}$$

has to hold for all test functions $\mathbf{v}$, i.e., all vectors $\mathbf{V}$, leading to the linear system $\mathbf{K}\mathbf{U} = \mathbf{F}$ to be solved for the *static* solution $\mathbf{U}$.

Assembly. The assembly of element and face stiffness matrices into the global stiffness matrix $\mathbf{K}$ can be formulated most easily in terms of individual 3×3 matrices. Let $\mathbf{K}_{K[i,j]}$ denote the 3×3 submatrix of $\mathbf{K}_K$ corresponding to the global stiffness matrix entry $\mathbf{K}_{ij}$. The matrices $\mathbf{K}_{K[i,j]}$ can be precomputed for all elements K using (4.7), and only those matrices that are non-zero need to be stored. Due to symmetry it holds that $\mathbf{K}_{K[i,j]} = \mathbf{K}_{K[j,i]}{}^T$, which further reduces the number of matrices to be precomputed. Similarly, $\mathbf{K}_{f1[ij]}$ and $\mathbf{K}_{f3[ij]}$ can be precomputed for all faces f. Note that $\mathbf{K}_{f3[ij]} = \mathbf{K}_{f3[ji]}{}^T$ and $\mathbf{K}_{f2[ij]} = \mathbf{K}_{f1[ji]}{}^T$, so $\mathbf{K}_{f2}$ does not need to be precomputed explicitly.

$\mathbf{K}_{ij}$ can now be defined in terms of element and face contributions as follows:

$$\mathbf{K}_{ij} = \sum_K \mathbf{K}_{K[ij]} + \sum_f \left(\mathbf{K}_{f1[ij]} + \mathbf{K}_{f1[ji]}{}^T + \mathbf{K}_{f3[ij]} \right).$$

Using a notation where the operator $\leftarrow$ denotes assembly into the global stiffness matrix for all K, f, this can equivalently be written as:

$$\begin{aligned}
\mathbf{K}_{ij} &\leftarrow \mathbf{K}_{K[ij]} \\
\mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f1[ij]} \\
\mathbf{K}_{ji} &\leftarrow \mathbf{K}_{f1[ij]}{}^T \\
\mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f3[ij]}
\end{aligned}$$

Boundary Conditions. Dirichlet boundary constraints can be prescribed in DG FEM as weak or strong constraints. The latter simply removes some DOFs from the system, i.e., fixes the coefficients $\mathbf{u}_i$ for the corresponding N_i. Weak boundary conditions are imposed by appropriately defining averages and jumps at boundary elements. For a prescribed displacement $\mathbf{g}$ this means to define the function values on the "free" side of boundary faces $f \in \partial\Omega$ as

$$\begin{aligned}
\mathbf{u}^- &:= \mathbf{g}, & \sigma^-(\mathbf{u}) &:= \sigma^+(\mathbf{u}), \\
\mathbf{v}^- &:= \mathbf{0}, & \sigma^-(\mathbf{v}) &:= \sigma^+(\mathbf{v}).
\end{aligned}$$

Time Integration. Dynamic simulations of deformable objects with time-varying $\mathbf{U}(t)$ and $\mathbf{F}(t)$ require additional inertial and damping forces, resulting in the governing equations

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{D}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}, \tag{4.11}$$

with mass matrix $\mathbf{M}$ and damping matrix $\mathbf{D}$, equivalent to CG FEM [Nealen et al., 2006]. The 3×3 submatrices of the mass matrix $\mathbf{M}$ can be computed as

$$\mathbf{M}_{ij} = \rho \sum_K \int_K N_i(\mathbf{x})\, N_j(\mathbf{x})\ \mathbf{I}_3,$$

where ρ is the uniform density of the object. The damping matrix $\mathbf{D}$ is usually computed as a linear combination of the stiffness matrix $\mathbf{K}$ and the mass matrix $\mathbf{M}$.

In order to guarantee stability a semi-implicit Euler time-integration is employed: Integrating the system forward in time from state $(\mathbf{U}_k, \dot{\mathbf{U}}_k)$ to $(\mathbf{U}_{k+1}, \dot{\mathbf{U}}_{k+1})$ using a time step size Δt, the new DOFs and their acceleration are expressed in terms of the new velocities as

$$\mathbf{U}_{k+1} \approx \mathbf{U}_k + \dot{\mathbf{U}}_{k+1}\Delta t,$$
$$\ddot{\mathbf{U}}_{k+1} \approx \frac{\dot{\mathbf{U}}_{k+1} - \dot{\mathbf{U}}_k}{\Delta t},$$

resulting in $\dot{\mathbf{U}}_{k+1}$ being the only unknown for time step $k+1$. Formulating the governing equations (4.11) for time step $k+1$ and reordering the resulting terms gives the sparse, symmetric, positive definite linear system

$$(\mathbf{K}\Delta t^2 + \mathbf{D}\Delta t + \mathbf{M})\, \dot{\mathbf{U}}_{k+1} = (\mathbf{F} - \mathbf{K}\mathbf{U}_k)\Delta t + \mathbf{M}\dot{\mathbf{U}}_k \tag{4.12}$$

to be solved for $\dot{\mathbf{U}}_{k+1}$ in each time step, from which in turn the values of the DOFs $\mathbf{U}_{k+1}$ can be computed.

Two kinds of linear system solvers were applied to the resulting equations: preconditioned conjugate gradients [Saad and van der Vorst, 2000] and sparse Cholesky factorization [Toledo et al., 2003]. While both worked well in all the experiments performed, the Cholesky solver turned out to scale better to larger problems thanks to its quasi-linear asymptotic complexity, as also observed in Botsch et al. [2005].

4.3 Arbitrary Polyhedral Elements

The main advantage of DG FEM is the possibility to use non-conforming, discontinuous shape functions N_i. This added flexibility allows us to employ simple degree-k polynomials $\{1, x, y, z, xy, \ldots, z^k\}$ as (non-nodal) basis

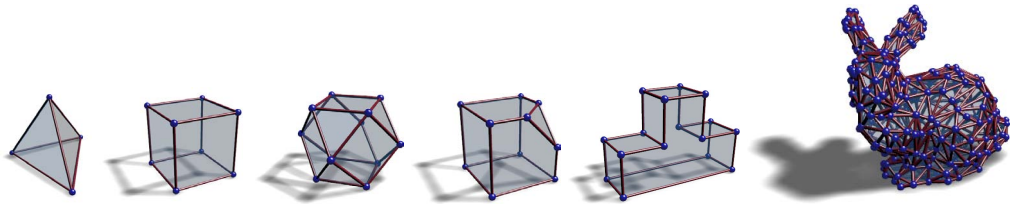

Figure 4.3: *Examples of possible element shapes include the common tetrahedral and hexahedral elements (left), as well as arbitrary convex and non-convex polyhedra (middle). It is even possible to treat the whole simulation domain as one single element (right).*

functions within each element K. In this work, either 4 linear or 10 quadratic basis functions were used per element. Notice that k should be ≥ 1, since then the DG method can exactly reproduce rigid motions, yielding a linear, continuous displacement function $\mathbf{u}$ without jumps [Cockburn, 2003].

In contrast to nodal basis functions, these non-nodal basis functions no longer depend on the element shape, thereby enabling us to work with arbitrarily shaped elements. For practical reasons, however, element shapes are restricted to convex or non-convex polyhedra (i.e., planar faces and linear edges, see Fig. 4.3), which still is considerably more flexible than the convex polyhedra with triangulated faces of Wicke et al. [2007]. Compared to the harmonic shape functions of Martin et al. [2008], which also allow for non-convex elements, our polynomial basis functions are simpler and therefore more efficient to compute.

For a practical implementation we have to accurately and efficiently compute integrals of the form

$$\int_K N_a N_b, \quad \int_K \frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_j}, \quad \int_f N_a N_b, \quad \int_f \frac{\partial N_a}{\partial x_i} N_b,$$

over elements K and faces f, since they are the building blocks for the matrix assembly described in Section 4.2.2. While tetrahedra or hexahedra can be integrated analytically, general polyhedral elements with nodal basis functions typically require numerical integration, which trades accuracy for performance [Wicke et al., 2007]. In contrast, as will be shown in the following section, polynomial functions can be integrated analytically over a polyhedron. By using polynomial basis functions, all of the aforementioned element integrals can thus be computed exactly up to numerical round-off errors.

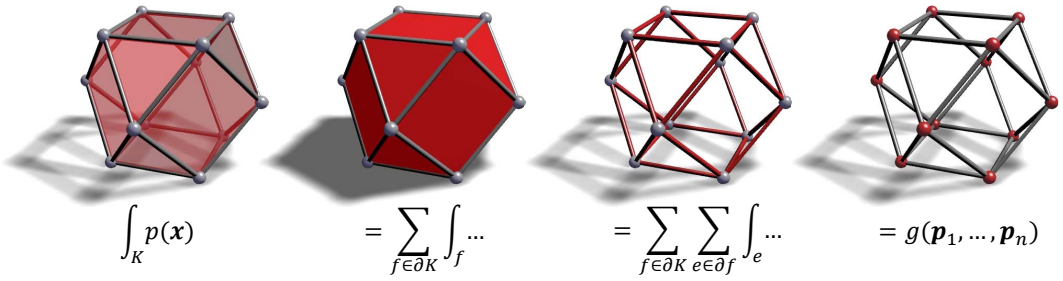

Figure 4.4: *The basic idea of divergence theorem-based volume integration: an integral over a polyhedron K is reduced to a sum of area integrals, a sum of line integrals, and finally becomes a function of the polyhedron's nodal positions.*

4.3.1 Divergence Theorem Integration

Using the divergence theorem, the volume integral of a degree-k polynomial p over an element K can be reduced to an area integral of a degree-$(k+1)$ polynomial q over its boundary ∂K, i.e., to a sum of integrals over its faces. Each face integral can in turn be reduced to line integrals over its edges $e \in \partial f$, which in the end results in degree-$(k+3)$ polynomials in the edge endpoints (see Fig. 4.4).

Divergence Theorem. For a function $f : \mathbb{R}^3 \to \mathbb{R}$ and a constant 3-vector $\mathbf{d}$ the following variant of the divergence theorem holds:

$$\int_\Omega \nabla f \cdot \mathbf{d} \, d\Omega = \int_\Gamma f \, \mathbf{d} \cdot \mathbf{n} \, d\Gamma, \tag{4.13}$$

where Ω is a volume bounded by the surface Γ. (4.13) also holds if Ω is a planar surface in $\mathbb{R}^3$ and Γ is the boundary of that surface. In this case, $\mathbf{d}$ must be orthogonal to the normal of the surface.

Volume Integration. We want to integrate a function $p : \mathbb{R}^3 \to \mathbb{R}$ over a polyhedral volume element K

$$J = \int_K p(\mathbf{x}) \, d\mathbf{x}$$

with $\mathbf{x} = (x_1, x_2, x_3)^T$. Assuming the existence of a function $q : \mathbb{R}^3 \to \mathbb{R}$ and a non-zero vector $\mathbf{d}$ with the property $p(\mathbf{x}) = \nabla q(\mathbf{x}) \cdot \mathbf{d}$, the integrand can be substituted. Applying the divergence theorem (4.13) gives

$$J = \int_K \nabla q(\mathbf{x}) \cdot \mathbf{d} \, d\mathbf{x} = \int_\Gamma q(\mathbf{x}) \, \mathbf{d} \cdot \mathbf{n} \, dA = \sum_{f \in \partial K} \int_f q(\mathbf{x}) \, dA \;\; \mathbf{d} \cdot \mathbf{n}^f \tag{4.14}$$

where $\mathbf{n}^f$ is the outward unit normal vector of element face f. Using the (yet to be defined) function q and vector $\mathbf{d}$, we are thus able to express the volume integral of a function p as a sum of surface integrals of a function q.

Surface Integration. To transition from surface integrals to line integrals, we start with the integral of a function $q : \mathbb{R}^3 \to \mathbb{R}$ over a planar face f:

$$I = \int_f q(\mathbf{x})\, dA.$$

Let $\mathbf{n}^f = (n_1^f, n_2^f, n_3^f)^T$ be the unit normal of the face. We assume that the boundary of the face consists of m straight edges, where the e-th edge connects nodes $\mathbf{p}_e$ and $\mathbf{p}_{e+1}$, $e \in \{1, \ldots, m\}$, and $\mathbf{p}_{m+1} := \mathbf{p}_1$. The plane equation of f is $\mathbf{x} \cdot \mathbf{n}^f - C = 0$ with $C := \mathbf{p}_1 \cdot \mathbf{n}^f$.

Following Mirtich's [1996] approach of projecting surface integrals onto coordinate planes, the face f is projected onto either the $x_1 = 0$, $x_2 = 0$ or $x_3 = 0$ plane, depending on the face's orientation. We want the projected face to have non-zero area, thus it is reasonable to use the maximum component of $\mathbf{n}^f$ as the direction of projection, i.e. to project face f onto the $x_d = 0$ plane with $d = \arg\max_i |n_i^f|$. Let the unit vector $\mathbf{e}_d = (\delta_{1d}, \delta_{2d}, \delta_{3d})^T$ denote the direction of projection where δ_{ij} is the Kronecker delta.

A point $\mathbf{x} = (x_1, x_2, x_3)^T$ on f is projected to $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \hat{x}_3)^T$ on $\hat{f}$ with

$$\hat{x}_i = (1 - \delta_{id}) x_i.$$

Using the plane equation of f, the inverse mapping (from $\hat{\mathbf{x}}$ to $\mathbf{x}$) can be found to be

$$\begin{aligned} x_d &= \frac{C - \hat{x}_{d+1} n_{d+1}^f - \hat{x}_{d+2} n_{d+2}^f}{n_d^f}, \\ x_{d+1} &= \hat{x}_{d+1}, \\ x_{d+2} &= \hat{x}_{d+2}, \end{aligned}$$

where the indices of x and $\hat{x}$ are assumed to wrap around, i.e. $x_4 := x_1$, $x_5 := x_2$, and so on.

The integral I can now be computed as

$$I = \frac{1}{|n_d^f|} \int_{\hat{f}} \hat{q}(\hat{\mathbf{x}})\, d\hat{\mathbf{x}} \quad \text{with} \quad \hat{q}(\hat{\mathbf{x}}) := q(\mathbf{x}(\hat{\mathbf{x}})).$$

Assuming the existence of a function $\hat{r} : \mathbb{R}^3 \to \mathbb{R}$ and a non-zero vector $\hat{\mathbf{d}}$ satisfying $\hat{q}(\hat{\mathbf{x}}) = \nabla \hat{r}(\hat{\mathbf{x}}) \cdot \hat{\mathbf{d}}$ and $\mathbf{e}_d \cdot \hat{\mathbf{d}} = 0$, the divergence theorem (4.13) can be applied to the projected face $\hat{f}$:

$$I = \frac{1}{|n_d^f|} \int_{\hat{f}} \nabla \hat{r}(\hat{\mathbf{x}}) \cdot \hat{\mathbf{d}} \, d\hat{\mathbf{x}} = \frac{1}{|n_d^f|} \sum_{\hat{e} \in \partial \hat{f}} \int_{\hat{e}} \hat{r}(\hat{\mathbf{x}}) ds \, \hat{\mathbf{d}} \cdot \hat{\mathbf{n}}_{\hat{e}}.$$

Here, the integral over the boundary of $\hat{f}$ has been replaced by a sum over integrals over the projected face's edges $\hat{e}$. $\hat{\mathbf{n}}_{\hat{e}}$ is the outward unit normal of edge $\hat{e}$ on the projection plane.

Line Integration. As all edges are assumed to be straight, they can be parameterized as

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{p}}_{\hat{e}}(1-t) + \hat{\mathbf{p}}_{\hat{e}+1} t$$

with $0 \leq t \leq 1$ and projected nodes $\hat{\mathbf{p}}_{\hat{e}}$. The integral becomes

$$I = \frac{1}{|n_d^f|} \sum_{\hat{e} \in \hat{f}} \hat{l}_{\hat{e}} \int_0^1 \hat{r}(\hat{\mathbf{x}}(t)) dt \, \hat{\mathbf{d}} \cdot \hat{\mathbf{n}}_{\hat{e}}$$

where $\hat{l}_{\hat{e}} := ||\hat{\mathbf{p}}_{\hat{e}+1} - \hat{\mathbf{p}}_{\hat{e}}||$ is the length of the projected edge $\hat{e}$. The normal of the projected edge $\hat{e}$ can be found to be

$$\hat{\mathbf{n}}_{\hat{e}} = \frac{1}{\hat{l}_{\hat{e}}} \operatorname{sign}(n_d^f)(\hat{\mathbf{p}}_{\hat{e}+1} - \hat{\mathbf{p}}_{\hat{e}}) \times \mathbf{e}_d.$$

The factor $\operatorname{sign}(n_d^f)$ is required in order to ensure that the normal of the projected edge is always pointing outwards. The integral now simplifies to

$$I = \frac{1}{n_d^f} \sum_{e \in f} \int_0^1 \hat{r}(\hat{\mathbf{x}}(t)) dt \, (\mathbf{p}_{e+1} - \mathbf{p}_e) \cdot (\mathbf{e}_d \times \hat{\mathbf{d}}) \tag{4.15}$$

where we have made use of the fact that $\hat{\mathbf{p}}_e \cdot (\mathbf{e}_d \times \hat{\mathbf{d}}) = \mathbf{p}_e \cdot (\mathbf{e}_d \times \hat{\mathbf{d}})$.

Volume to Nodal Positions. Using (4.15) to compute the face integral in (4.14) finally gives a closed form for the computation of $J = \int_K p(\mathbf{x})$:

$$J = \sum_{f \in \partial K} \frac{1}{n_d^f} \sum_{e \in f} \left(\int_0^1 \hat{r}(\hat{\mathbf{x}}(t)) dt \right) (\mathbf{p}_{e+1} - \mathbf{p}_e) \cdot (\mathbf{e}_d \times \hat{\mathbf{d}})(\mathbf{d} \cdot \mathbf{n}^f) \tag{4.16}$$

If p is a polynomial in $\mathbf{x}$, q can be computed by integrating p in direction x_d, then substituting $\hat{\mathbf{x}}$ for $\mathbf{x}$ in q to get $\hat{q}$. Integration of $\hat{q}$ in direction $\hat{\mathbf{d}}$ gives $\hat{r}(\hat{\mathbf{x}})$. Parameterizing $\hat{\mathbf{x}}$ with t and analytically computing the integral $\int_0^1$ then results in a polynomial in $\mathbf{p}_e$ and $\mathbf{p}_{e+1}$. The integral (4.16) thus evaluates to a polynomial in the nodal positions of the polyhedral element K.

4.3.2 Integration Algorithm

The following steps show how to compute $\int_K p(\mathbf{x})\,d\mathbf{x}$ for an arbitrary polynomial $p(\mathbf{x})$ with $\mathbf{x} = (x_1, x_2, x_3)^T$.

1. Integrate $p(\mathbf{x})$ formally to obtain the polynomial $q(\mathbf{x})$ in $\mathbf{x}$:

$$q(\mathbf{x}) \leftarrow \int p(\mathbf{x})\,dx_1$$

2. Perform the following three steps for $i \in \{1,2,3\}$, with j and k defined as $j = (i \bmod 3) + 1$ and $k = ((i+1) \bmod 3) + 1$.

   a) Transform $q(\mathbf{x})$ into the polynomial $\hat{q}(\hat{\mathbf{x}}, \mathbf{n})$ in $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \hat{x}_3)^T$ with the symbolical constant $\mathbf{n} = (n_1, n_2, n_3)^T$ by performing the following substitutions:

$$\begin{aligned} x_i &\rightarrow \frac{1}{n_i} - \hat{x}_j \frac{n_j}{n_i} - \hat{x}_k \frac{n_k}{n_i}, \\ x_j &\rightarrow \hat{x}_j, \\ x_k &\rightarrow \hat{x}_k \end{aligned}$$

   b) Integrate $\hat{q}(\hat{\mathbf{x}}, \mathbf{n})$ formally to obtain the polynomial $\hat{r}(\hat{\mathbf{x}}, \mathbf{n})$ in $\hat{\mathbf{x}}$:

$$\hat{r}(\hat{\mathbf{x}}, \mathbf{n}) \leftarrow \int \hat{q}(\hat{\mathbf{x}}, \mathbf{n})\,d\hat{x}_j$$

   c) Integrate formally over the edge connecting $\mathbf{a} = (a_1, a_2, a_3)^T$ and $\mathbf{b} = (b_1, b_2, b_3)^T$ to get a polynomial in $\mathbf{a}$ and $\mathbf{b}$:

$$P_i(\mathbf{a}, \mathbf{b}, \mathbf{n}) \leftarrow \; (b_k - a_k) \int_0^1 \hat{r}(\mathbf{a}\,(1-t) + \mathbf{b}\,t, \mathbf{n})\,dt$$

3. The integral over the volume can now be computed as follows, where $\mathbf{n}^f = (n_1^f, n_2^f, n_3^f)^T$ defines the plane of face f as $\{\mathbf{x} \in \mathbb{R}^3 | \mathbf{x} \cdot \mathbf{n}^f = 1\}$. $d_f \in \{1,2,3\}$ is the direction of projection for face f which must be chosen such that $n_{d_f}^f \neq 0$. $\mathbf{a}^e$ and $\mathbf{b}^e$ are the start and end points of edge e.

$$\int_K p(\mathbf{x})\,d\mathbf{x} = \sum_{f \in \partial K} \frac{n_1^f}{n_{d_f}^f} \sum_{e \in \partial f} P_{d_f}(\mathbf{a}^e, \mathbf{b}^e, \mathbf{n}^f)$$

In a practical implementation, a code generation tool would be used to create the code for computing the polynomials P_i, $i \in \{1,2,3\}$, for one particular polynomial $p(\mathbf{x})$.

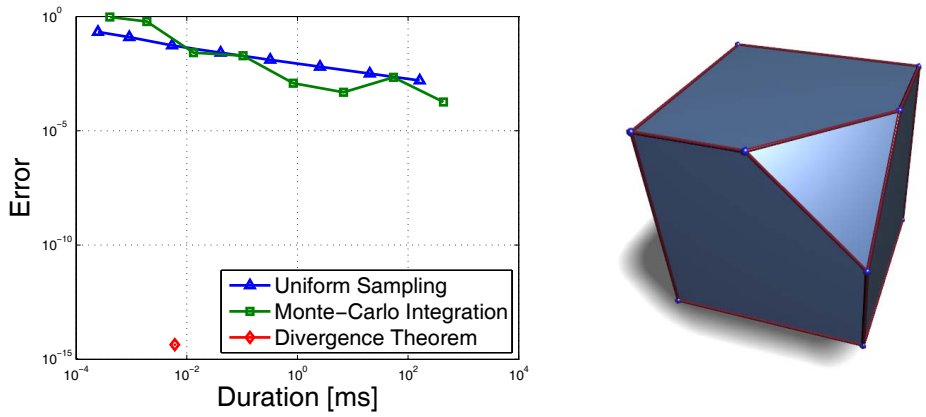

Figure 4.5: *Comparison of different numerical integration methods to the analytic method based on the divergence theorem. In this example the function $f(x,y,z) = x^2$ was integrated over the corner-cut cube model shown on the right.*

4.3.3 Evaluation

Fig. 4.5 compares the analytic integration to different numerical schemes in terms of accuracy and performance. While the method based on the divergence theorem is exact up to round-off errors, it is also reasonably efficient: a straightforward numerical integration still shows an error of about 10^{-2} for the same computation time. Compared to CG FEM using the mean value polyhedral elements of Wicke et al. [2007], the exact integration method is faster by an order of magnitude.

4.4 Stiffness Warping

Under large rotational deformations, linear FEM shows artifacts such as an unrealistic increase in volume. To avoid the cost of a full nonlinear simulation but still get physically plausible deformations in these cases, a corotated formulation can be employed, which computes elastic forces in a rotated coordinate frame defined for each element [Müller and Gross, 2004; Hauth and Strasser, 2004].

In linear CG FEM, the forces acting on the nodes of an element K are computed from nodal displacements $\mathbf{U}$ and the element stiffness matrix $\mathbf{K}_K$ defined in (4.7) as follows:

$$\mathbf{F}_K = \mathbf{K}_K\mathbf{U} = \mathbf{K}_K\left(\mathbf{X} - \mathbf{X}^0\right), \tag{4.17}$$

with $\mathbf{X}$ and $\mathbf{X}^0$ denoting the deformed and undeformed nodal positions, respectively. In order to avoid the aforementioned rotational artifacts, the *corotational*, or *warped stiffness* approach [Müller and Gross, 2004; Hauth and Strasser, 2004] first reverts the element's rotation, computes displacements and forces in the un-rotated state, and re-rotates the resulting forces:

$$\mathbf{F}_K \;=\; \mathbf{R}_K \mathbf{K}_K \left(\mathbf{R}_K^T \mathbf{X} - \mathbf{X}^0 \right), \tag{4.18}$$

where $\mathbf{R}_K$ is a block-diagonal matrix containing the 3×3 rotation matrix of element K on its diagonal.

This approach cannot be directly applied to DG for two reasons. First, the contributions resulting from integrals over interior faces are associated with two elements and require special treatment. Second, in case non-nodal basis functions are used, we will no longer be solving for nodal displacements, and $\mathbf{X}^0$ in (4.17) needs to be generalized to a set of degrees of freedom defining the undeformed state of the object in terms of the basis functions N_i.

4.4.1 Element and Face Contributions

Element contributions (4.7) can be treated just as in CG FEM using (4.18). We determine the rotations of general polyhedra by first fitting an affine transformation to the nodal displacements in the least squares sense, and then extracting its rotational component $\mathbf{R}_K$ using polar decomposition [Hauth and Strasser, 2004].

Note that for face contributions (4.8), (4.9), (4.10) we cannot simply apply (4.18) using the *face's rotation*, since that would lead to ghost forces and instabilities similar to the per-vertex stiffness warping of [Müller et al., 2002]. Moreover, the corotational method is only required to correct artifacts due to linear strain $\bar{\boldsymbol{\epsilon}} = \mathbf{B}\mathbf{U}$, and hence is not needed for (4.10).

For the face contributions (4.8) and (4.9) it is crucial that the strains $\mathbf{B}_f^+ \mathbf{U}$ and $\mathbf{B}_f^- \mathbf{U}$, which constitute $\mathbf{B}_f^{\{\}}$, are computed consistently with the strains of the element contributions (4.18) of K^+ and K^-. This requires to use the *elements' rotations* $\mathbf{R}_f^+$ and $\mathbf{R}_f^-$ for correcting $\mathbf{B}_f^+ \mathbf{U}$ and $\mathbf{B}_f^- \mathbf{U}$, respectively. We therefore have to split up the stiffness matrices $\mathbf{K}_{f1}$ and $\mathbf{K}_{f2}$ w.r.t. strain contributions from either K^+ or K^-, yielding the four stiffness matrices

$$\mathbf{K}_{f1}^{\pm} \;:=\; -\frac{1}{2} \int_f \mathbf{H}_f^{[\![]\!]^T} \mathbf{N}_f^T \bar{\mathbf{C}} \mathbf{B}_f^{\pm},$$

$$\mathbf{K}_{f2}^{\pm} \;:=\; -\frac{1}{2} \int_f \mathbf{B}_f^{\pm T} \bar{\mathbf{C}} \mathbf{N}_f \mathbf{H}_f^{[\![]\!]},$$

where $(\cdot)^{\pm}$ again denotes either $(\cdot)^{+}$ or $(\cdot)^{-}$. These stiffness matrices allow for a consistent warping of a face f's contributions, such that we get five corotated contributions:

$$\begin{aligned}\mathbf{F}_{f1}^{\pm} &= \mathbf{R}_f^{\pm}\mathbf{K}_{f1}^{\pm}\left(\mathbf{R}_f^{\pm T}\mathbf{X}-\mathbf{X^0}\right),\\ \mathbf{F}_{f2}^{\pm} &= \mathbf{R}_f^{\pm}\mathbf{K}_{f2}^{\pm}\left(\mathbf{R}_f^{\pm T}\mathbf{X}-\mathbf{X^0}\right),\\ \mathbf{F}_{f3} &= \mathbf{K}_{f3}\left(\mathbf{X}-\mathbf{X^0}\right).\end{aligned}$$

4.4.2 Non-Nodal Basis Functions

In order to use stiffness warping for non-nodal basis functions, we need to generalize the definition of the vector $\mathbf{X}^0$ representing the undeformed state. To this end, we have to find $\mathbf{x}^0 = (\mathbf{x}_1^{0T},\ldots,\mathbf{x}_n^{0T})^T$ satisfying the identity $\sum_i \mathbf{X}_i^0 N_i(\mathbf{x}) \equiv \mathbf{x}$. For nodal basis functions, this vector would contain the nodal positions of the undeformed mesh. Since for each element K our non-nodal basis functions always contain the linear polynomials (see Section 4.3), finding $\mathbf{X}^0$ is trivial. For each element K, if its linear basis functions are

$$N_{i_K}(\mathbf{x}) = x, \quad N_{j_K}(\mathbf{x}) = y, \quad N_{k_K}(\mathbf{x}) = z,$$

we simply set the corresponding coefficients to

$$\mathbf{x}_{i_K}^0 = (1,0,0)^T,\ \mathbf{x}_{j_K}^0 = (0,1,0)^T,\ \mathbf{x}_{k_K}^0 = (0,0,1)^T,$$

and use $\mathbf{x}_{l_K}^0 = (0,0,0)^T$ for all its other basis functions. This results in a vector $\mathbf{X}^0$ representing the undeformed state, based on which stiffness warping can be performed just as for nodal basis functions.

Note that for quadratic or higher order basis functions, stiffness warping only removes the global element rotation, whereas local rotations due to bending might remain. While this was not a problem in all of the experiments performed, such cases can easily be detected and the respective elements can be refined (see Section 4.7).

4.4.3 Warped Assembly

To formulate the assembly of the stiffness matrix (see Section 4.2.2) in the presence of stiffness warping, we need to first split up (4.18) into a term proportional to $\mathbf{U}$ and a static force term as follows:

$$\mathbf{R}_K\mathbf{K}_K\mathbf{R}_K^T\mathbf{U} = \mathbf{F}_K + \mathbf{R}_K\mathbf{K}_K(\mathbf{I}_{3n} - \mathbf{R}_K^T)\mathbf{X}^0.$$

Every time the element rotations change, the warped element contributions are to be re-assembled as follows:

$$\begin{aligned}\mathbf{K}_{ij} &\leftarrow \mathbf{R}_K \mathbf{K}_{K[ij]} \mathbf{R}_K^T \\ \mathbf{F}_i &\leftarrow \mathbf{R}_K \mathbf{K}_{K[ij]} (\mathbf{I}_3 - \mathbf{R}_K^T) \mathbf{X}_j^0\end{aligned}$$

Note that contrary to the notation used previously, $\mathbf{R}_K$ and $\mathbf{R}_f^{\pm}$ denote 3×3 rotation matrices here. The face contributions are assembled as follows:

$$\begin{aligned}\mathbf{K}_{ij} &\leftarrow \mathbf{R}_f^{\pm} \mathbf{K}_{f1[ij]}^{\pm} \mathbf{R}_f^{\pm T} \\ \mathbf{F}_i &\leftarrow \mathbf{R}_f^{\pm} \mathbf{K}_{f1[ij]}^{\pm} (\mathbf{I}_3 - \mathbf{R}_f^{\pm T}) \mathbf{X}_i^0 \\ \mathbf{K}_{ji} &\leftarrow \mathbf{R}_f^{\pm} \mathbf{K}_{f1[ij]}^{\pm\, T} \mathbf{R}_f^{\pm T} \\ \mathbf{F}_j &\leftarrow \mathbf{R}_f^{\pm} \mathbf{K}_{f1[ij]}^{\pm\, T} (\mathbf{I}_3 - \mathbf{R}_f^{\pm T}) \mathbf{X}_j^0 \\ \mathbf{K}_{ij} &\leftarrow \mathbf{K}_{f3[ij]}\end{aligned}$$

4.5 MLS-Based Surface Embedding

When it comes to the simulation of complex models, a common approach for keeping computation costs low is to embed a high resolution surface mesh into a lower resolution simulation mesh. The latter can be simulated efficiently, and its displacement field $\mathbf{u}(\mathbf{x})$ is used to deform the surface mesh [Faloutsos et al., 1997; Müller et al., 2004b; James et al., 2004; Sifakis et al., 2007b]. In DG FEM, the *discontinuous* displacement $\mathbf{u}$ cannot be applied directly to the high resolution surface, since it would lead to gaps and self-intersections.

In a first step, discontinuities are removed by stitching the simulation mesh. For each node $\mathbf{x}_i^0 \in \mathcal{T}$, the average of its different displacements $\mathbf{u}|_K(\mathbf{x}_i^0)$ corresponding to its incident elements $K \in \mathcal{N}_i$ is computed, similar to Botsch et al. [2006]:

$$\tilde{\mathbf{u}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{K \in \mathcal{N}_i} \mathbf{u}|_K \left(\mathbf{x}_i^0\right). \tag{4.19}$$

This results in a deformed, *continuous* simulation mesh, which is sufficient for visualizing the simulation mesh itself.

The averaged nodal displacements have to be interpolated within elements in order to deform the embedded mesh. For tetrahedral or hexahedral elements this amounts to simple linear or trilinear interpolation, respectively. For more

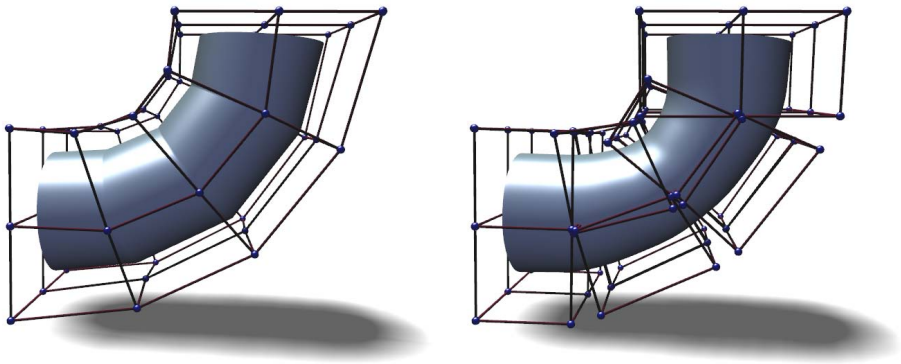

Figure 4.6: *Comparison of embedding techniques. Stitching the discontinuous simulation mesh, followed by barycentric interpolation, leads to C^0 artifacts (left). In contrast, the presented smooth MLS-based embedding yields a considerably higher surface quality (right).*

general convex or non-convex polyhedra, mean value coordinates [Floater et al., 2005; Ju et al., 2005] or harmonic coordinates [Joshi et al., 2007] can be employed. All these methods, however, correspond to a non-smooth, generalized barycentric C^0 interpolation, resulting in clearly visible shading artifacts for coarse simulation meshes (see Fig. 4.6, left).

Botsch et al. [2007] employ globally supported radial basis functions for high quality interpolation, but the involved dense linear systems are prohibitive for complex simulation meshes. To overcome these limitations, and inspired by meshless methods [Müller et al., 2004a; Pauly et al., 2005], this work proposes a smooth embedding based on moving-least-squares (MLS) interpolation.

If we denote by $\mathbf{x}_i^0$ the nodes of the undeformed simulation mesh, and by $\tilde{\mathbf{u}}_i$ their averaged displacements, then the displacement at a material point $\mathbf{x}$ is computed by fitting an affine transformation, which amounts to minimizing the weighted least square error

$$E_{\text{mls}}(\mathbf{x}) = \frac{1}{2}\sum_i \theta\left(\left\|\mathbf{x}-\mathbf{x}_i^0\right\|\right)\left\|\mathbf{C}(\mathbf{x})^T\mathbf{p}\left(\mathbf{x}_i^0\right)-\tilde{\mathbf{u}}_i\right\|^2, \tag{4.20}$$

with $\mathbf{p}(x,y,z) = (1,x,y,z)^T$ and $\theta(d)$ a (truncated) Gaussian weight function. Computing $\partial E_{\text{mls}}/\partial\mathbf{p} = \mathbf{0}$ results in a 4×4 linear system $\mathbf{A}(\mathbf{x})\,\mathbf{C}(\mathbf{x}) = \mathbf{b}(\mathbf{x})$ that yields the coefficients $\mathbf{C}(\mathbf{x})$ for the interpolated displacement $\tilde{\mathbf{u}}(\mathbf{x}) = \mathbf{C}(\mathbf{x})^T\mathbf{p}(\mathbf{x})$ at the position $\mathbf{x}$. This MLS-based embedding has several interesting properties:

- The smoothness of the interpolation is determined by the weighting kernels θ, resulting in a high quality embedding for our choice of Gaussian kernels (see Fig. 4.6, right).

- The use of linear polynomials $\mathbf{p}(\mathbf{x})$, in combination with the partition of unity property of MLS shape functions, guarantees the exact reproduction of linear displacements $\mathbf{u}$, i.e., in particular of rigid motions [Fries and Matthies, 2004].
- Since the approach is entirely meshless it can be used to interpolate within arbitrarily shaped elements. Choosing the support radius of θ proportional to the local sampling density at $\mathbf{x}_i^0$ (e.g., distances to one-ring neighbors), yields smooth interpolations even for irregular meshes.
- An accurate approximation of higher order polynomial displacements $\mathbf{u}$ only requires to add more samples $(\mathbf{x}_i^0, \tilde{\mathbf{u}}_i)$ to (4.20), such as edge, face, or element midpoints.
- The interpolated displacement $\tilde{\mathbf{u}}(\mathbf{x})$ of a vertex $\mathbf{x}$ of the embedded mesh linearly depends on $\mathbf{C}(\mathbf{x})$, which in turn linearly depends on the $\tilde{\mathbf{u}}_i$ used in (4.20), which finally linearly depend on $\mathbf{u}_i$ through (4.19) and (4.4). Hence, by combining these linear relationships, the weights $\theta\left(\left\|\mathbf{x} - \mathbf{x}_i^0\right\|\right)$ as well as the set $\mathcal{N}(\mathbf{x})$ of relevant basis functions N_i can be precomputed, such that during the simulation only

$$\tilde{\mathbf{u}}(\mathbf{x}) = \sum_{i \in \mathcal{N}(\mathbf{x})} \theta\left(\left\|\mathbf{x} - \mathbf{x}_i^0\right\|\right) N_i(\mathbf{x})\, \mathbf{u}_i =: \sum_{i \in \mathcal{N}(\mathbf{x})} W_i(\mathbf{x})\, \mathbf{u}_i \tag{4.21}$$

  has to be evaluated as a linear combination of the DOFs $\mathbf{u}_i$.

4.6 Collisions

Since collision handling is not the focus of this work, it has been restricted to simple penalty-based collision response within the semi-implicit time integration. The basic approach is equivalent to CG FEM, therefore only the differences due to the discontinuous displacement field $\mathbf{u}$ are discussed here.

Suppose that in the current time step we detect a collision at a displaced material point $\mathbf{x}_c + \tilde{\mathbf{u}}(\mathbf{x}_c)$. Since we use the interpolated displacement $\tilde{\mathbf{u}}$ of (4.21), $\mathbf{x}_c$ can be an arbitrary embedded point, e.g., a vertex of the embedded surface mesh. Nodal collisions using the stitched displacement (4.19) is just a special case of this formulation.

For collision response a penalty force proportional to the penetration depth is added to the system. Consider an oriented collision plane with outward unit

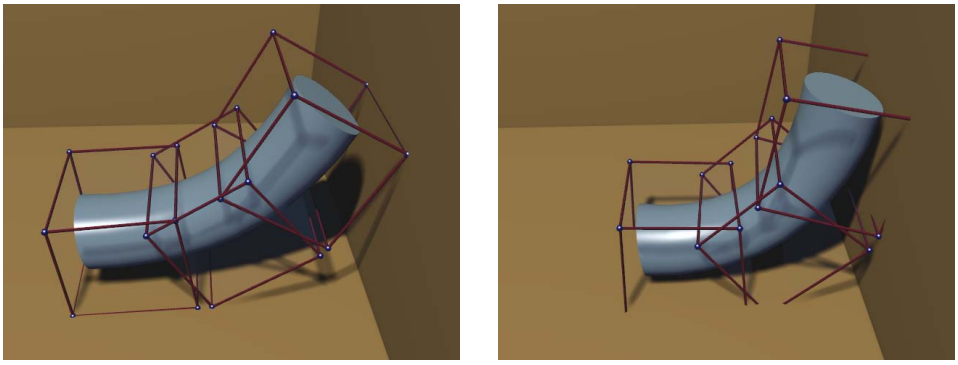

Figure 4.7: *Collision handling on the nodes of the simulation mesh (left) and the vertices of the embedded mesh (right).*

normal vector $\mathbf{n}$ and $\mathbf{p}$ being an arbitrary point on the plane. The penetration depth d can be computed as

$$d = \mathbf{p} \cdot \mathbf{n} - (\mathbf{x}_c + \tilde{\mathbf{u}}(\mathbf{x}_c)) \cdot \mathbf{n}.$$

Defining a collision force proportional to the penetration depth yields

$$\mathbf{f}(\mathbf{x}_c) = k\,d\,\mathbf{n} = \mathbf{A}\tilde{\mathbf{u}}(\mathbf{x}_c) + \mathbf{b}$$

with $\mathbf{A} = -k\,\mathbf{n}\mathbf{n}^T$ and $\mathbf{b} = k\,\mathbf{n}\mathbf{n}^T(\mathbf{p} - \mathbf{x}_c)$, where k is the collision penalty. This force is used in the semi-implicit solver whenever $d > 0$. The corresponding penalty energy is

$$E_{\text{coll}}(\mathbf{x}_c) \;=\; \frac{1}{2}\tilde{\mathbf{u}}(\mathbf{x}_c)^T\,\mathbf{A}\,\tilde{\mathbf{u}}(\mathbf{x}_c) \;+\; \tilde{\mathbf{u}}(\mathbf{x}_c)^T\,\mathbf{b},$$

which after inserting the definition of $\tilde{\mathbf{u}}$ in (4.21) becomes

$$\frac{1}{2}\sum_{i,j}\mathbf{u}_i^T W_i(\mathbf{x}_c)\,\mathbf{A}\,W_j(\mathbf{x}_c)\,\mathbf{u}_j \;+\; \sum_i \mathbf{u}_i^T W_i(\mathbf{x}_c)\,\mathbf{b}.$$

Since this collision energy corresponds to an external force, it has to be either subtracted from the internal potential energy $\frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U}$ or to be added to the external energy $\mathbf{U}^T\mathbf{F}$. Hence, we can incorporate the collision energy E_{coll} into the system (4.11) by updating 3×3 blocks of the stiffness matrix $\mathbf{K}$ and 3-vectors of the external force $\mathbf{F}$ (see Section 4.2.2):

$$\begin{aligned}\mathbf{K}_{ij} &-= W_i(\mathbf{x}_c)\,\mathbf{A}\,W_j(\mathbf{x}_c)\,,\\ \mathbf{F}_i &+= \mathbf{b}\,W_i(\mathbf{x}_c)\,,\end{aligned}$$

for all $i, j \in \mathcal{N}(\mathbf{x}_c)$, i.e., the set of basis functions W_i, respectively N_i, influencing the collision point $\mathbf{x}_c$ (see (4.21)).

Method	Resolution	#DOFs	Spars.	Int.	Ass.	Solve
BZ lin.	$10\times10\times10$	12k	0.28%	532	22	656
IP lin.	$10\times10\times10$	12k	0.62%	1437	87	734
CG trilin.	$15\times15\times15$	12k	0.58%	3750	41	641
BZ quad.	$10\times10\times10$	30k	0.28%	3062	152	7797
IP quad.	$10\times10\times10$	30k	0.64%	8344	621	8484

Table 4.1: *Comparison of BZ and IP with linear/quadratic basis functions to trilinear CG FEM for 3D elasticity. The mesh resolution is chosen to match the DOFs of DG and CG. The table lists matrix sparsity and timings (in ms) for volume integration, matrix assembly, and the solution of the linear system (taken on an Intel Core2 Duo 2.4 GHz).*

If the simulation mesh is also used for visualization, simple nodal collisions are sufficient in most cases, as for instance for the examples shown in Section 4.7. However, for embedded simulations collisions should be detected and handled on the vertices of the embedded surface (see Fig. 4.7).

4.7 Results

This section demonstrates how the possibility to use arbitrary polyhedral elements and simple polynomial shape functions can be exploited to derive a versatile and efficient simulation technique. Before presenting specific example applications, the general advantages and disadvantages of DG FEM compared to CG FEM are discussed.

DG FEM versus CG FEM for Linear Elasticity. A quantitative comparison between CG and DG using BZ/IP penalties and linear/quadratic basis functions, based on a 2D Poisson problem with analytically known solution has already been shown in Chapter 3, see Fig. 3.8. In addition, Table 4.1 gives some statistics and timings of the same five methods for 3D linear elasticity. Note that even for the same mesh and basis functions DG provides more degrees of freedom (DOFs) than CG, since nodes can "split" due to discontinuous displacements. The plots and timings are therefore with respect to DOFs.

The only additional parameter compared to CG FEM is the penalty weight η in (4.2), (4.3), which has to be sufficiently high to guarantee stability. One simple strategy to find a suitable penalty is to start with a low value and double it until $\mathbf{K}$ is positive definite. This worked well for all experiments performed in the context of this work and typically leads to η in the order

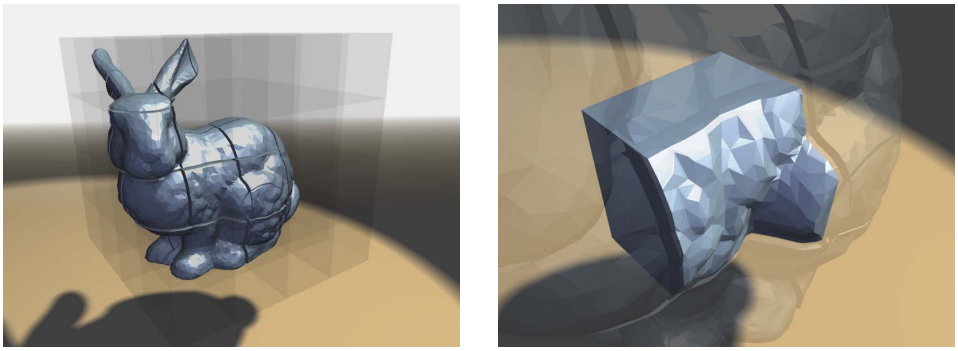

Figure 4.8: *Intersecting the bunny with a hex-grid generates 41 elements (left). Closeup view of a non-convex element (right).*

of 10^1–10^2. Note that η should not be too high, since otherwise the method resembles CG and does not exploit its additional DOFs (Fig. 4.2).

The missing consistency terms of BZ (see (4.1), (4.3)) allow for sparser matrices and higher efficiency. Furthermore, the method is stable for any positive penalty η. Although lacking theoretical convergence guarantees, BZ shows a reasonable convergence behavior in practice and gives visually convincing results. We therefore consider it well suited for typical graphics applications requiring physically plausible deformations only. For more accurate simulations the IP method is the better choice. Highly accurate results can be achieved using more complex numerical fluxes in combinations with nonlinear strain measures [Ten Eyck and Lew, 2006].

As has been shown in Fig. 3.8, both DG methods lead to higher condition numbers of the linear systems, which, however, has not been a problem in all our examples, for both the conjugate gradients solver as well as the sparse Cholesky factorization.

Since standard CG FEM is slightly more efficient than DG FEM for the same number of DOFs and basis functions of the same degree (Table 4.1) and also easier to implement, it will stay the preferred method for many applications. However, as soon as topological changes of the simulation mesh are required or if complex element shapes have to be simulated, the higher flexibility of DG FEM pays off, as for instance in the following examples.

Mesh Generation by Hexahedral Slicing. A challenge in simulating deformable objects is the preservation of surface detail without introducing an excessive amount of simulation primitives. Commonly used approaches include voxelization of the object's volume or tetrahedrization. While voxelization is simple to implement and results in well-behaved elements, it

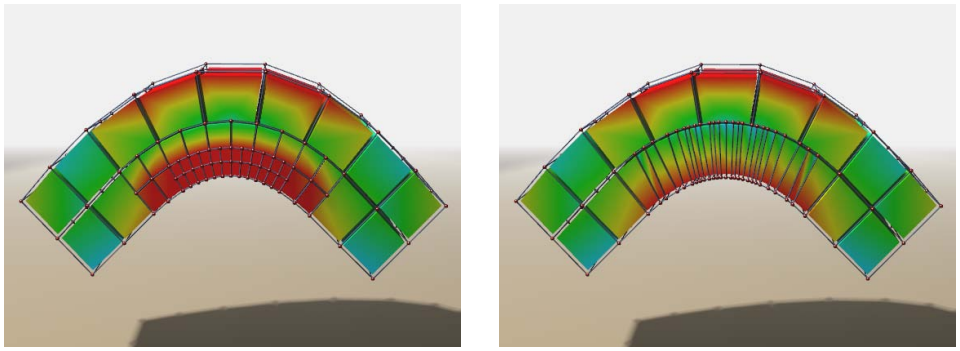

Figure 4.9: *A bar (36 hex-elements) is dynamically refined during bending. 1-to-8 subdivision results in 274 elements (left), whereas 1-to-2 refinement yields 77 elements (right).*

cannot accurately represent surface details unless a high number of elements is used. On the other hand, tetrahedral meshes can accurately represent objects defined by surface meshes, but result in a higher number of elements.

Using arbitrary elements in DG FEM gives rise to an interesting mesh generation algorithm that decouples the number of elements (and thus the DOFs) from the resolution of the surface mesh. Combining the strengths of both voxelization and tetrahedrization, the simulation mesh is generated by intersecting the object with a hexahedral grid. Each intersected cell then corresponds to a finite element, resulting in hexahedral elements in the interior and arbitrary polyhedra at the object's surface (see Fig. 4.8). Note that the strain energy is integrated over the exact volume of the object, whereas a pure embedded simulation could in this case lead to an erroneous coupling of the bunny's ears.

Dynamic Adaptivity. In order to make optimal use of the available computational resources, it is often desirable to adaptively enhance the resolution of a dynamic simulation around a specific area of interest. Using arbitrary elements in a DG framework allows for easy and flexible refinement.

We chose a simple criterion based on stress concentration, refining an element when its largest absolute principal stress exceeds a given threshold. For the actual topological refinement, we can, e.g., perform a regular 1-to-8 subdivision of hexahedral elements, conceptually similar to Grinspun et al. [2002]. An interesting alternative is the more flexible 1-to-2 split along the plane perpendicular to the principal stress direction, which generates fewer elements for the same refinement threshold (see Fig. 4.9).

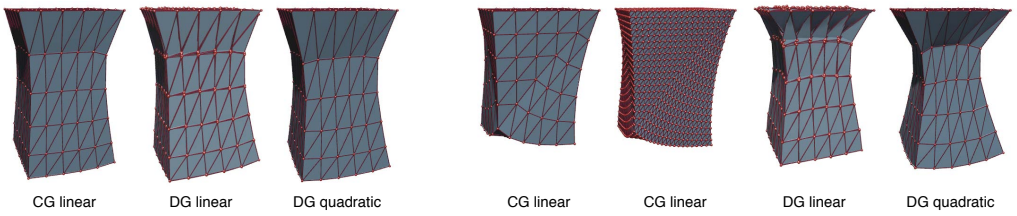

Figure 4.10: *A suspended cube consisting of 750 tetrahedra deforming under its own weight, simulated with linear CG FEM and linear/quadratic DG FEM (IP method), using a Poisson's ratio of $\nu = 0.3$ (left) and $\nu = 0.499$ (right). While DG gives the expected, symmetric solution, CG shows severe locking artifacts in the nearly incompressible case, even after mesh refinement.*

Note that the refinement of an element is in no way restricted by the refinement level of its neighbors. When splitting an element, one can simply copy the parent's coefficients for displacement $\mathbf{u}_i$ and velocity $\dot{\mathbf{u}}_i$ to its children. This heuristic causes slight popping artifacts in dynamic simulations, which could be avoided by a more sophisticated projection technique.

Cutting. Using DG FEM for cutting simulations has a couple of advantages over existing methods. Being able to simulate arbitrary elements avoids complex remeshing of the simulation domain (see Fig. 4.1), similar in spirit to Molino et al. [2004], Wicke et al. [2007] and Sifakis et al. [2007a]. Furthermore, thanks to the analytic integration the contributions of newly created elements can be computed very efficiently and accurately, avoiding the need for expensive numerical integration during the simulation. By storing and reusing individual edge and face integrals, after splitting an element we only need to recompute integrals over edges and faces intersecting the cut plane.

Poorly shaped elements with negligible volume cause numerical problems, equivalently to CG FEM. However, those elements can effectively be avoided by simply merging them with neighboring elements, exploiting the fact that our method is not restricted to convex elements. Note that also for mesh generation and dynamic refinement we either prevent the generation of degenerate elements, or remove them by the mentioned sliver merging technique.

Locking. In the case of nearly incompressible materials, as the Poisson's ratio ν approaches the limit value of 0.5, standard FEM is known to exhibit an overly stiff behavior termed *locking*.

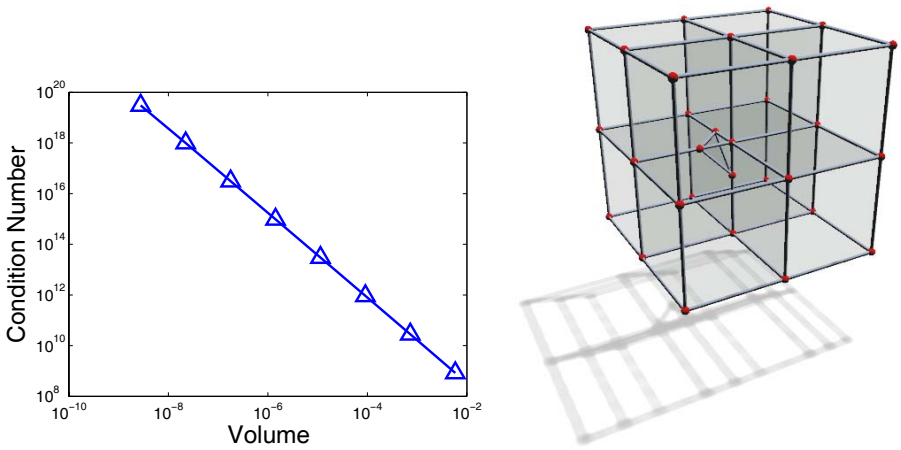

Figure 4.11: *Mesh of nine elements. As the volume of the innermost tetrahedral element decreases, the condition number of the global stiffness matrix increases.*

An intuitive explanation for this phenomenon is provided by the counting argument [Irving et al., 2007]: Each element introduces an additional volume constraint in order to preserve its volume locally. However, a continuous FEM mesh with n nodes has only $3n$ degrees of freedom, while in the specific case of a tetrahedral mesh, the number of elements is at least $4n$, resulting in an overconstrained system.

On the other hand, the additional degrees of freedom present in discontinuous Galerkin FEM allow the method to effectively circumvent locking, as demonstrated in Fig. 4.10. In this example, increasing the number of elements will not prevent locking in the CG FEM case. Also note that the locking CG FEM solution is strongly influenced by the topology of the simulation mesh, resulting in an asymmetric solution, whereas the DG FEM solution is free of such artifacts.

Sliver Elements. In standard FEM with nodal basis functions, the computation of shape functions and their derivatives typically involves the inversion of a Jacobian matrix, causing numerical problems for ill-shaped elements. This affects the integration of basis functions over the element as well as other uses of basis functions such as the interpolation of nodal quantities.

This particular problem can be avoided in DG FEM, as basis functions are defined in global coordinates. However, elements of small volume still cause problems in DG FEM. As stated in Shewchuck [2002], the condition number of the stiffness matrix of a tetrahedral mesh is related to the ratio between the volume of the largest and the smallest element. A similar behavior can be observed in DG FEM, as shown in Fig. 4.11.

On the other hand, we note that in DG FEM elements with *locally* small features do not cause problems, as long as the total volume of the element stays reasonably large (Fig. 4.12). Note that in order to compute the condition number of the stiffness matrices in those examples, appropriate boundary constraints were introduced in order to get a unique solution to the static problem.

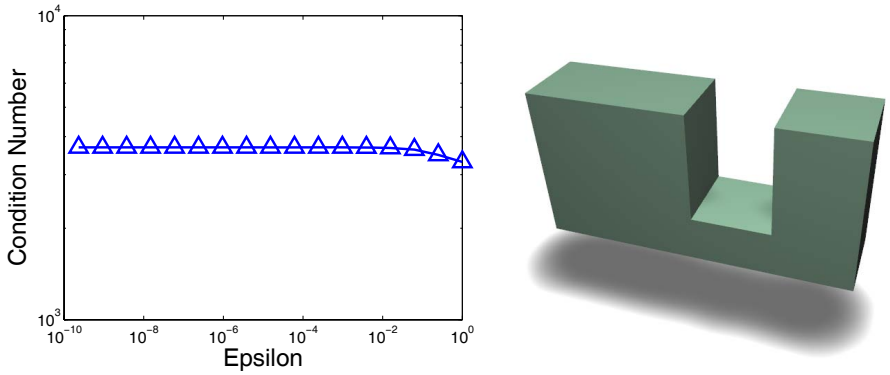

Figure 4.12: *Single non-convex element with a locally thin feature. Even as the height ε of the narrowed middle section approaches zero, the condition number of the stiffness matrix stays finite.*

4.8 Discussion and Outlook

In this chapter, a novel simulation technique for deformable models based on discontinuous Galerkin FEM was presented. The main advantage of DG FEM is the flexibility to use discontinuous shape functions, which was exploited for the efficient simulation of arbitrary polyhedral elements. The presented generalization of stiffness warping enables physically plausible large-scale deformations, and the novel MLS-based surface embedding allows to simulate complex models in the DG framework.

The versatility of this approach was presented on conceptually simple, efficient, and robust techniques for mesh generation, adaptive refinement, and cutting. While there are successful methods for each individual problem, the presented approach provides an interesting alternative that handles all problems in a single, consistent DG FEM framework.

Promising directions for future work include nonlinear elasticity simulations of both solids and shells, which would benefit even more from the flexibility offered by DG FEM. In particular, the next chapter shows how DG FEM can reduce the continuity requirements for thin shell simulations and allow

for the representation of small features and changing topology through the means of *enrichment textures*.

CHAPTER 5

Enrichment Textures for Shells

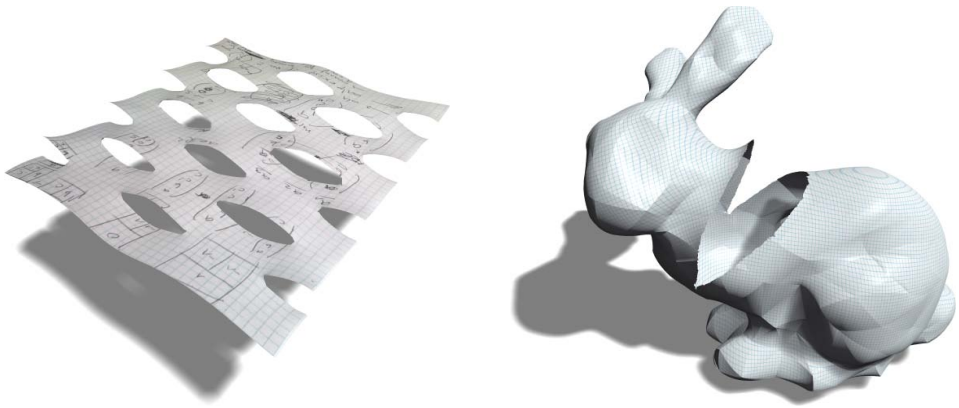

Figure 5.1: *Starting from a single shell element, the proposed method is able to capture complex deformations and topological changes by simply enriching the element's basis functions through texture maps (left). Fracturing a bunny triangle mesh (right).*

After having applied the discontinuous Galerkin finite element method to the simulation of deformable solids as presented in the previous chapter, it only seems natural to think about how the simulation of thin objects, i.e. shells, can benefit from the reduced continuity requirements offered by DG FEM. With that in mind, this chapter presents a method for simulating highly detailed cutting and fracturing of thin shells using low-resolution simulation meshes.

Instead of refining or remeshing the underlying simulation domain to resolve complex cut paths, custom-designed basis functions are introduced that enrich the approximation by accurately representing these detailed features.

The proposed approach generalizes the extended finite element method (XFEM) by storing the enrichment basis functions in *enrichment textures*, which allows for fracture and cutting discontinuities at a resolution much finer than the underlying mesh, similar to image textures for increased visual resolution. Furthermore, *harmonic enrichment functions* can handle multiple, intersecting, arbitrarily shaped, progressive cuts per element in a simple and unified framework.

By using an underlying shell simulation based on discontinuous Galerkin FEM, the restrictive requirement of C^1 continuous shell basis functions is relaxed and thus simpler, C^0 continuous XFEM enrichment functions can be used. The DG shell method is further extended by a corotational formulation that allows for the plausible simulation of large deformations while keeping the computational cost low.

5.1 Overview

In order to simulate the deformation behavior of thin shells, it is not sufficient to formulate their mechanical behavior as described in Section 3.5. We also need to provide an appropriate discretization, which due to the curvilinear coordinates and the involved differential equation (a fourth-order elliptic PDE) is non-trivial. Discontinuous Galerkin FEM provides a rather elegant way of solving this problem, by reducing the continuity requirements between elements from C^1 to C^0. After presenting the DG FEM-based discretization and basis functions for quad and triangle elements, corotation will be introduced such that large deformations can be simulated without the artifacts typically observed for geometrically linear models.

The proposed method is then presented as a texture-based variant of the XFEM which is able to handle multiple cuts and partial cuts in a natural way, whereas traditional XFEM approaches need to handle those as special cases. The method is also able to cut through meshes consisting of several elements while keeping the support of the new basis functions local to the elements intersected by the cut.

Next to shell simulation and topological changes, rendering is also an important aspect of the proposed method. The simulated features will be at the scale of a texture pixel rather than the size of a finite element, and simply

tessellating the elements at a high resolution for rendering is not the best possible approach. Instead, a renderer employing a geometry shader is proposed that is able to resolve details at the texture level even for coarse tessellations.

5.2 Discontinuous Galerkin FEM Thin Shells

This section describes an implementation of the discontinuous Galerkin finite element method for thin shells presented by Noels and Radovitzky [2008]. Building on the Kirchhoff-Love shell theory presented in Chapter 3, the DG weak form is reviewed and the assembly of the stiffness matrix is described in detail. Extending on this previous work, a corotational extension to the method is presented which allows for the simulation of large rotational deformations without the typical linearization artifacts of a linear shell model.

5.2.1 Shell Model

Eq. (5.1) shows again the weak form for the Kirchhoff-Love shell model as it has already been presented in Section 3.5.2. This weak form could be discretized using a standard finite element approach [Hughes, 2000; Zienkiewicz and Taylor, 2000].

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) \;:=\; & \int_\Omega \epsilon_{\alpha\beta}(\mathbf{v})\mathcal{H}_n^{\alpha\beta\gamma\delta}\epsilon_{\gamma\delta}(\mathbf{u})\,\mathrm{d}\mathcal{S} \\ & + \int_\Omega \rho_{\alpha\beta}(\mathbf{v})\mathcal{H}_m^{\alpha\beta\gamma\delta}\rho_{\gamma\delta}(\mathbf{u})\,\mathrm{d}\mathcal{S} \end{aligned} \tag{5.1}$$

However, a closer look at the weak form reveals that due to the second derivatives in the bending strain it represents a fourth order elliptic PDE, which implies that appropriate basis functions must lie in the Sobolev space $H^2(\Omega)$, i.e. have square-integrable first and second order derivatives [Hughes, 2000]. Unfortunately, constructing suitable shell functions on irregular meshes is a rather complex task. A variety of approaches are in use, such as ignoring the continuity requirements or introducing additional variables, e.g. derivatives at edge mid-points [Zienkiewicz and Taylor, 2000]. Other approaches use non-local interpolation schemes [Cirak et al., 2000].

A DG FEM approach for thin shells avoids the problem by only requiring basis functions that are C^0 across elements and enforcing C^1 continuity in a weak sense. The formulation results in additional integrals over element edges, which effectively introduce a deformation energy term which penalizes discontinuities in the derivatives.

In the case of DG FEM as described in Noels and Radovitzky [2008], the weak form becomes

$$a^{\mathrm{DG}}(\mathbf{u},\mathbf{v}) := \sum_K a^K(\mathbf{u},\mathbf{v}) + \sum_e a^e(\mathbf{u},\mathbf{v}) \tag{5.2}$$

for elements K and interior edges e (edges with neighboring elements on both sides). a^K corresponds to the weak form (3.36) evaluated on element K. a^e is defined on interior edges as follows:

$$\begin{aligned} &a^e(\mathbf{u},\mathbf{v}) := \\ &\int_e [\![\Delta\mathbf{t}(\mathbf{v})]\!] \cdot \boldsymbol{\varphi}_{0,\gamma} \nu_\delta^- \{\mathcal{H}_m\} [\![\Delta\mathbf{t}(\mathbf{u})]\!] \cdot \boldsymbol{\varphi}_{0,\alpha} \nu_\beta^- \eta^e \,\mathrm{d}\Gamma \\ &+ \int_e \left\{\rho_{\alpha\beta}(\mathbf{v}) \mathcal{H}_m \boldsymbol{\varphi}_{0,\gamma}\right\} \nu_\delta^- \cdot [\![\Delta\mathbf{t}(\mathbf{u})]\!] \,\mathrm{d}\Gamma \\ &+ \int_e [\![\Delta\mathbf{t}(\mathbf{v})]\!] \cdot \left\{\rho_{\alpha\beta}(\mathbf{u}) \mathcal{H}_m \boldsymbol{\varphi}_{0,\gamma}\right\} \nu_\delta^- \,\mathrm{d}\Gamma. \end{aligned} \tag{5.3}$$

Considering an edge e, element K^+ lies on the left-hand side of the directed edge and element K^- on the right-hand side. Quantities associated with either element are superscripted with a $-$ or $+$, respectively. $[\![\cdot]\!]$ is the jump operator, defined in this context as

$$[\![\mathbf{v}]\!] := \mathbf{v}^+ - \mathbf{v}^-$$

and $\{\cdot\}$ the average operator, with its usual definition

$$\begin{aligned} \{\mathbf{v}\} &:= \frac{1}{2}(\mathbf{v}^+ + \mathbf{v}^-), \\ \{v\} &:= \frac{1}{2}(v^+ + v^-) \end{aligned}$$

for $\mathbf{v} \in \mathbb{R}^3$ and $v \in \mathbb{R}$. Note that the definitions of the jump and average operators used in the context of shells are different from the ones used in Section 3.1.2 for solid FEM. η^e is a penalty factor for edge e which depends on a global penalty parameter η and the local element size as follows:

$$\eta^e = \frac{\eta}{h^e}$$

where h^e is the characteristic size of the edge. The characteristic size is computed from the areas and circumferences of the two adjacent elements using

$$h^e = \min\left(\frac{|A^+|}{|\partial A^+|}, \frac{|A^-|}{|\partial A^-|}\right).$$

In the above weak form, $\Delta\mathbf{t}(\mathbf{u})$ is the change of the normal vector, which can be computed as

$$\begin{aligned}\Delta\mathbf{t}(\mathbf{u}) \;=\; & \frac{1}{\bar{j}_0}\big(\boldsymbol{\varphi}_{0,1}\times\mathbf{u}_{,2}-\boldsymbol{\varphi}_{0,2}\times\mathbf{u}_{,1}\\ & +\mathbf{t}_0\,\mathbf{u}_{,1}\cdot(\mathbf{t}_0\times\boldsymbol{\varphi}_{0,2})\\ & -\mathbf{t}_0\,\mathbf{u}_{,2}\cdot(\mathbf{t}_0\times\boldsymbol{\varphi}_{0,1})\big).\end{aligned} \tag{5.4}$$

$\boldsymbol{\nu}^- = (\nu_1^-,\nu_2^-)^T$ is the outer unit normal of K^- represented in the *conjugate basis* $\boldsymbol{\varphi}_0^\alpha$, see (3.39).

Note that the first term in (5.3) penalizes the *jump* of the *change* of the normal vector $\mathbf{t}$ on the edge. In other words, $[\![\Delta\mathbf{t}(\mathbf{u})]\!]$ is zero if the displacement field $\mathbf{u}$ changes the normals $\mathbf{t}^-$ and $\mathbf{t}^+$ on either side of the edge in the same way. The other two terms in (5.3) are responsible for making the formulation consistent and symmetric, see Arnold et al. [2001].

5.2.2 Stiffness Matrix Assembly

This section describes in detail how the stiffness matrix $\mathbf{K}$ can be computed from element and edge contributions. Instead of following the formulation of Noels and Radovitzky [2008], Voigt notation will be employed here in order to avoid tensor notation and provide a more tangible implementation.

Basis Function Discretization. The undeformed shell surface is defined in terms of basis functions $N^a : \Omega \to \mathbb{R}$ as

$$\boldsymbol{\varphi}_0 = \sum_a N^a \mathbf{X}_0^a,$$

where $\mathbf{X}_0^a \in \mathbb{R}^3$ is the initial position of node a. It follows that the undeformed surface basis vectors can be computed as

$$\boldsymbol{\varphi}_{0,\alpha} = \sum_a N^a_{,\alpha}\mathbf{X}_0^a,$$

by differentiating the basis functions N^a with respect to ξ^1 and ξ^2. Similarly, the displacement field $\mathbf{u}$ is discretized as

$$\mathbf{u} = \sum_a N^a \mathbf{u}^a$$

with nodal displacements $\mathbf{u}^a$. Representing the solution $\mathbf{u}$ as well as the test function $\mathbf{v}$ in terms of the basis functions N^a results in a linear system

$$\mathbf{KU} = \mathbf{F}, \text{ with } \begin{cases} \mathbf{K}_{ij} \;=\; a^{\mathrm{DG}}(\mathbf{I}_3 N^i, \mathbf{I}_3 N^j) \\ \mathbf{F}_i \;=\; \int_\Omega \mathbf{f}\, N^i \end{cases}, \tag{5.5}$$

which is solved for the degrees of freedom $\mathbf{U} = ({\mathbf{u}^1}^T, \ldots, {\mathbf{u}^n}^T)^T$, given external forces $\mathbf{f}$.

Quad and Triangle Elements. As proposed in Noels and Radovitzky [2008], quadrangular elements with 8 nodes and bi-quadratic basis functions can be employed. The basis functions are:

$$\begin{aligned}
N^1 &= -\frac{1}{4}(-1+\xi^2)(-1+\xi^1)(\xi^1+\xi^2+1)\\
N^2 &= -\frac{1}{4}(-1+\xi^2)(1+\xi^1)(\xi^1-\xi^2-1)\\
N^3 &= \frac{1}{4}(1+\xi^2)(1+\xi^1)(\xi^1+\xi^2-1)\\
N^4 &= \frac{1}{4}(1+\xi^2)(-1+\xi^1)(\xi^1-\xi^2+1)\\
N^5 &= \frac{1}{2}(1-\xi^1\xi^1)(1-\xi^2)\\
N^6 &= \frac{1}{2}(1-\xi^2\xi^2)(1+\xi^1)\\
N^7 &= \frac{1}{2}(1-\xi^1\xi^1)(1+\xi^2)\\
N^8 &= \frac{1}{2}(1-\xi^2\xi^2)(1-\xi^1)
\end{aligned}$$

for element coordinates (ξ^1, ξ^2) with $-1 \leq \xi^1, \xi^2 \leq 1$.

When working with triangular 6-node elements, the following quadratic basis functions can be used

$$\begin{aligned}
N^1 &= 1-3\xi^1-3\xi^2+2\xi^1\xi^1+4\xi^1\xi^2+2\xi^2\xi^2\\
N^2 &= \xi^1(2\xi^1-1)\\
N^3 &= \xi^2(2\xi^2-1)\\
N^4 &= -4\xi^1(-1+\xi^1+\xi^2)\\
N^5 &= 4\xi^1\xi^2\\
N^6 &= -4\xi^2(-1+\xi^1+\xi^2)
\end{aligned}$$

where $\xi^1, \xi^2 \geq 0$ and $\xi^1 + \xi^2 \leq 1$. The node numbering for the two elements is depicted in Fig. 5.2.

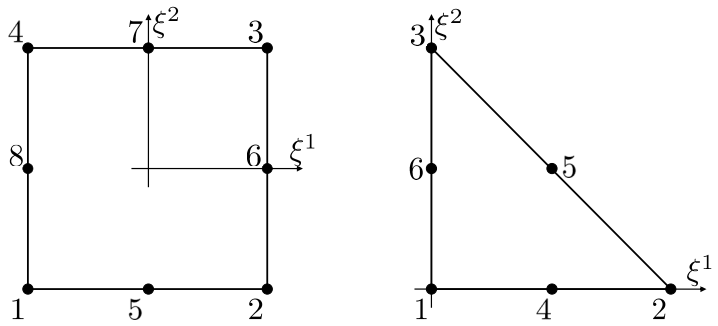

Figure 5.2: *Node numbering for the quadrangular (left) and triangular element (right).*

Element Contributions. Similar to Cirak et al. [2000], the membrane strain two-tensor can be represented in Voigt notation as

$$\hat{\varepsilon} := \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} + \varepsilon_{21} \end{pmatrix}.$$

The membrane strain in Voigt notation can then be computed as

$$\hat{\varepsilon} = \sum_a \hat{\mathbf{B}}_n^a \mathbf{u}^a$$

where the 3×3 matrix $\hat{\mathbf{B}}_n^a$ is made up of three row vectors:

$$\hat{\mathbf{B}}_n^a := \begin{pmatrix} {\mathbf{b}_{n11}^a}^T \\ {\mathbf{b}_{n22}^a}^T \\ {\mathbf{b}_{n12}^a}^T + {\mathbf{b}_{n21}^a}^T \end{pmatrix}_{3\times 3}$$

with

$$\mathbf{b}_{n\alpha\beta}^a := \boldsymbol{\varphi}_{0,\alpha} N_{,\beta}^a.$$

Similarly, the bending strain can be written in Voigt notation as

$$\hat{\rho} := \begin{pmatrix} \rho_{11} \\ \rho_{22} \\ 2\rho_{12} \end{pmatrix},$$

which can be computed as

$$\hat{\rho} = \sum_a \hat{\mathbf{B}}_m^a \mathbf{u}^a$$

where the 3×3 matrix $\hat{\mathbf{B}}_m^a$ is made up of three row vectors:

$$\hat{\mathbf{B}}_n^a := \begin{pmatrix} \mathbf{b}_{m11}^{a\,T} \\ \mathbf{b}_{m22}^{a\,T} \\ 2\mathbf{b}_{m12}^{a\,T} \end{pmatrix}_{3\times 3}$$

with

$$\begin{aligned} \mathbf{b}_{m\alpha\beta}^a \;:=\; & \boldsymbol{\varphi}_{0,\alpha\beta} \cdot \mathbf{t}_0 \frac{1}{\bar{j}_0} \left(N_{,1}^a (\boldsymbol{\varphi}_{0,2} \times \mathbf{t}_0) - N_{,2}^a (\boldsymbol{\varphi}_{0,1} \times \mathbf{t}_0) \right) \\ & + \frac{1}{\bar{j}_0} \left(N_{,1}^a (\boldsymbol{\varphi}_{0,\alpha\beta} \times \boldsymbol{\varphi}_{0,2}) - N_{,2}^a (\boldsymbol{\varphi}_{0,\alpha\beta} \times \boldsymbol{\varphi}_{0,1}) \right) \\ & - N_{,\alpha\beta}^a \mathbf{t}_0. \end{aligned} \tag{5.6}$$

Next, we define the Voigt notation matrix $\hat{\mathbf{H}}$ as

$$\hat{\mathbf{H}} := \begin{pmatrix} \mathcal{H}^{1111} & \mathcal{H}^{1122} & \mathcal{H}^{1112} \\ \mathcal{H}^{2211} & \mathcal{H}^{2222} & \mathcal{H}^{2212} \\ \mathcal{H}^{1211} & \mathcal{H}^{1222} & \mathcal{H}^{1212} \end{pmatrix},$$

where $\mathcal{H}$ is the constitutive tensor defined in (3.39). The constitutive tensors for membrane and bending stresses (in Voigt notation) can thus be defined as

$$\hat{\mathbf{H}}_n := \frac{Eh}{1-\nu^2} \hat{\mathbf{H}}$$

and

$$\hat{\mathbf{H}}_m := \frac{Eh^3}{12(1-\nu)^2} \hat{\mathbf{H}}.$$

The assembly of an element's membrane stiffness into the global stiffness matrix $\mathbf{K}$ can finally be written as

$$\mathbf{K}^{ab} += \int \hat{\mathbf{B}}_n^{a\,T} \hat{\mathbf{H}}_n \hat{\mathbf{B}}_n^b \, \mathrm{d}\mathcal{S}, \tag{5.7}$$

where $\mathbf{K}^{ab}$ is the 3×3 block of the stiffness matrix at position (a, b). Similarly, bending stiffness contributions are assembled using

$$\mathbf{K}^{ab} += \int \hat{\mathbf{B}}_m^{a\,T} \hat{\mathbf{H}}_m \hat{\mathbf{B}}_m^b \, \mathrm{d}\mathcal{S}. \tag{5.8}$$

During the element assembly, (5.7) and (5.8) are computed for all pairs of element basis functions a, b for all elements and added to the global stiffness matrix $\mathbf{K}$ as described above.

Edge Contributions. For the computation of edge contributions, a local coordinate system on the edge e is defined as follows: the undeformed normal vector $\mathbf{t}_0^e$ at a point on the edge is set to the normalized average of the adjacent elements' normal vectors at that point, i.e.

$$\mathbf{t}_0^e := \frac{\mathbf{t}_0^+ + \mathbf{t}_0^-}{\|\mathbf{t}_0^+ + \mathbf{t}_0^-\|}. \tag{5.9}$$

The directed edge is parameterized with $t \in [-1,1]$. A function $\boldsymbol{\xi}_e^+(t) = (\xi_e^{1+}(t), \xi_e^{2+}(t))^T$ determines the position of the edge in local coordinates of element K^+. The first basis vector (along the edge) can thus be defined as

$$\boldsymbol{\varphi}_{0,1}^e := \frac{\partial \boldsymbol{\varphi}_0^+(\boldsymbol{\xi}_e^+(t))}{\partial t} = \boldsymbol{\varphi}_{0,1}^+ \frac{\partial \xi_e^{1+}(t)}{\partial t} + \boldsymbol{\varphi}_{0,2}^+ \frac{\partial \xi_e^{2+}(t)}{\partial t}. \tag{5.10}$$

Finally, define

$$\boldsymbol{\varphi}_{0,2}^e := \frac{\mathbf{t}_0^e \times \boldsymbol{\varphi}_{0,1}^e}{\|\mathbf{t}_0^e \times \boldsymbol{\varphi}_{0,1}^e\|}. \tag{5.11}$$

Note that $\boldsymbol{\varphi}_{0,2}^e$ is equal to the outward unit normal $\boldsymbol{\nu}^-$ of element K^-. Computing $\bar{j}_0$ in this local coordinate system according to (3.35), it becomes the curve length Jacobian.

The assembly of edge e results in the following contributions to the global stiffness matrix $\mathbf{K}$:

$$\begin{aligned} \mathbf{K}^{ab} \quad += \quad & \int (\tilde{\Delta}\mathbf{t}^{a\pm})^T \hat{\mathbf{N}}^T \hat{\mathbf{H}}_m \hat{\mathbf{N}} \tilde{\Delta}\mathbf{t}^{b\pm} \eta^e \, s^a s^b \, \mathrm{d}\Gamma \\ & + \frac{1}{2} \int (\hat{\mathbf{B}}_m^{a\pm})^T (\hat{\mathbf{H}}_P^{\pm})^T \hat{\mathbf{N}} \tilde{\Delta}\mathbf{t}^{b\pm} s^b \, \mathrm{d}\Gamma \\ & + \frac{1}{2} \int (\tilde{\Delta}\mathbf{t}^{a\pm})^T \hat{\mathbf{N}} \hat{\mathbf{H}}_P^{\pm} \hat{\mathbf{B}}_m^{b\pm} s^a \, \mathrm{d}\Gamma. \end{aligned} \tag{5.12}$$

Note that $a, b \in B^+ \cup B^-$, where B^+ and B^- are the sets of basis functions used by elements K^+ and K^-, respectively. Depending on which element the basis function is taken from, quantities are either evaluated in K^+ or K^-. $\hat{\mathbf{H}}_P^{\pm}$ is evaluated in the same element as the matrix $\hat{\mathbf{B}}_m^{\pm}$ related to the bending strain. $s^a \in \{-1,1\}$ assumes a value of 1 if the basis function a is taken from element K^+ and -1 if taken from element K^-.

Note that $\hat{\mathbf{H}}_m$ is evaluated in the local edge coordinate system. $\tilde{\Delta}\mathbf{t}^a$ is related to the change of surface normal and is defined as

$$\begin{aligned} \tilde{\Delta}\mathbf{t}^a \quad := \quad & \frac{1}{\bar{j}_0} ([\boldsymbol{\varphi}_{0,1}]_\times - \mathbf{t}_0 (\mathbf{t}_0 \times \varphi_{0,1})^T) N_{,2}^a \\ & - \frac{1}{\bar{j}_0} ([\boldsymbol{\varphi}_{0,2}]_\times - \mathbf{t}_0 (\mathbf{t}_0 \times \varphi_{0,2})^T) N_{,1}^a \end{aligned} \tag{5.13}$$

with

$$[\boldsymbol{\varphi}_{0,\alpha}]_\times := \begin{pmatrix} 0 & -\varphi^3_{0,\alpha} & \varphi^2_{0,\alpha} \\ \varphi^3_{0,\alpha} & 0 & -\varphi^1_{0,\alpha} \\ -\varphi^2_{0,\alpha} & \varphi^1_{0,\alpha} & 0 \end{pmatrix}. \tag{5.14}$$

As the local edge coordinate system is orthogonal and $\boldsymbol{\varphi}^e_{0,2} = \boldsymbol{\nu}^-$ and $\|\boldsymbol{\varphi}^e_{0,2}\| = 1$, it follows that $\nu^-_1 = 0$ and $\nu^-_2 = 1$. This fact can be used to construct the matrix $\hat{\mathbf{N}}$ as follows:

$$\begin{aligned} \hat{\mathbf{N}} &:= \begin{pmatrix} \boldsymbol{\varphi}_{0,1}{}^T \nu^-_1 \\ \boldsymbol{\varphi}_{0,2}{}^T \nu^-_2 \\ \boldsymbol{\varphi}_{0,1}{}^T \nu^-_2 + \boldsymbol{\varphi}_{0,2}{}^T \nu^-_1 \end{pmatrix}_{3\times 3} \\ &= \begin{pmatrix} \mathbf{0}^T \\ \boldsymbol{\varphi}_{0,2}{}^T \\ \boldsymbol{\varphi}_{0,1}{}^T \end{pmatrix}_{3\times 3}. \end{aligned} \tag{5.15}$$

$\hat{\mathbf{H}}^\pm_P$ represents the push-forward tensors and inverse transformation tensors combined with the constitutive tensor and formulated as a 3×3 matrix in Voigt notation:

$$\hat{\mathbf{H}}^\pm_P := (\hat{\mathbf{P}}^\pm)^T \hat{\mathbf{H}}^\pm_m \hat{\mathbf{P}}^\pm \hat{\mathbf{p}}^\pm, \tag{5.16}$$

where

$$\hat{\mathbf{P}}^\pm := \begin{pmatrix} P^\pm_{1111} & P^\pm_{1122} & P^\pm_{1112} \\ P^\pm_{2211} & P^\pm_{2222} & P^\pm_{2212} \\ 2P^\pm_{1211} & 2P^\pm_{1222} & P^\pm_{1212} + P^\pm_{1221} \end{pmatrix} \tag{5.17}$$

with

$$P^\pm_{\alpha\beta\gamma\delta} := (\boldsymbol{\varphi}^{,\gamma}_0 \cdot \boldsymbol{\varphi}^\pm_{0,\alpha})(\boldsymbol{\varphi}^{,\delta}_0 \cdot \boldsymbol{\varphi}^\pm_{0,\beta}) \tag{5.18}$$

and

$$\hat{\mathbf{p}}^\pm := \begin{pmatrix} p^\pm_{1111} & p^\pm_{1122} & p^\pm_{1112} \\ p^\pm_{2211} & p^\pm_{2222} & p^\pm_{2212} \\ 2p^\pm_{1211} & 2p^\pm_{1222} & p^\pm_{1212} + p^\pm_{1221} \end{pmatrix} \tag{5.19}$$

with

$$p^\pm_{\alpha\beta\gamma\delta} := (\boldsymbol{\varphi}^{\pm,\gamma}_0 \cdot \boldsymbol{\varphi}_{0,\alpha})(\boldsymbol{\varphi}^{\pm,\delta}_0 \cdot \boldsymbol{\varphi}_{0,\beta}). \tag{5.20}$$

Assembly Algorithm. The three main equations for the assembly of the global stiffness matrix are (5.7), (5.8), and (5.12). Algorithm 5.1 shows the main assembly steps in pseudocode.

```
1  Initialize global stiffness matrix K with zero
2  for all elements K:
3    for all a ∈ B^K:
4      for all b ∈ B^K:
5        Add membrane contribution (5.7) to K^ab
6        Add bending contribution (5.8) to K^ab
7      end
8    end
9  end
10 for all interior edges e:
11   // Note: Elements adjacent to e are K^+ and K^-
12   for all a ∈ B^+ ∪ B^-:
13     for all b ∈ B^+ ∪ B^-:
14       Add edge contribution (5.12) to K^ab
15     end
16   end
17 end
```

Algorithm 5.1: Summary of stiffness matrix assembly.

5.2.3 Dynamic Simulation

Given the stiffness matrix $\mathbf{K}$, a dynamic simulation of time-varying forces and displacements is governed by the equations

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{D}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}, \tag{5.21}$$

which are solved by semi-implicit Euler integration. Note that instead of lumping element masses to nodes, the full mass matrix $\mathbf{M}_{ij} = \int_\Omega N^i N^j$ is used, such that nodal basis functions and XFEM enrichment functions can later be handled in a uniform way. Moreover, if an element will be split into multiple parts, these will not only have the correct mass, but also the correct center of mass and moment of inertia.

5.2.4 Corotational DG FEM Shells

In its original formulation, the DG FEM treatment of Kirchhoff-Love shells of Noels and Radovitzky [2008] is not suitable for most graphics applications, because it uses a geometrically linear strain, which results in artifacts in case of large rotational deformations. We therefore propose a simple corotational

extension that allows arbitrary rotational deformations while still keeping the basic model linear in the displacements.

Following Müller and Gross [2004] and Thomaszewski et al. [2006], the strain is computed in an un-rotated coordinate frame and the resulting force rotated back to its original orientation. As components of the same element can deform significantly differently as a shell element is bent, we apply the corotational formulation to each quadrature point i, similar to Mezger et al. [2008]. For the assembly of element contributions into the global stiffness matrix $\mathbf{K}$ (Equations (5.7), (5.8)), numerical quadrature turns the integrals over the elements into per-quadrature point contributions $\mathbf{K}_K^{abi}$:

$$\mathbf{K}^{ab} += \mathbf{K}_K^{abi}. \tag{5.22}$$

Computing the strain in an un-rotated coordinate frame and rotating the resulting force back to its original orientation, the new contributions become

$$\mathbf{K}^{ab} += \mathbf{R}^i \mathbf{K}_K^{abi} \mathbf{R}^{i^T}, \tag{5.23}$$

$$\mathbf{F}^a += \mathbf{R}^i \mathbf{K}_K^{abi} (\mathbf{I} - \mathbf{R}^{i^T}) \mathbf{X}_0^b. \tag{5.24}$$

Note that a corrective term is added to the right-hand side $\mathbf{F}$. $\mathbf{X}_0^b$ is the value for basis function b that reproduces the undeformed configuration. For nodal degrees of freedom, this corresponds to the undeformed position of node b. When combining this model with enrichment functions, non-nodal degrees of freedom will be introduced and appropriate values for $\mathbf{X}_0^b$ must be defined for them. This can be achieved by applying the same idea as described in Section 4.4.2 for the case of non-nodal basis functions for solid FEM. $\mathbf{R}_i$ is a 3×3 rotation matrix that describes the local rotation at quadrature point i. It can be computed by polar decomposition of the deformation gradient at the quadrature point [Thomaszewski et al., 2006].

The same steps are performed for the edge contributions $\mathbf{K}_e^{abi}$, i.e., for the penalty term that weakly enforces C^1 continuity across edge e. This time, however, the rotation matrix is computed from the deformation gradient in the edge coordinate frame.

Qualitative Evaluation. Fig. 5.3 shows an example of a nonlinear deformation that would not be possible to simulate using a pure linear deformation model. The cylinder is fixed at both ends and slightly compressed, resulting in a typical buckling deformation. The mesh consists of 961 quad elements and 2945 nodes, with a global stiffness matrix of size 8835×8835.

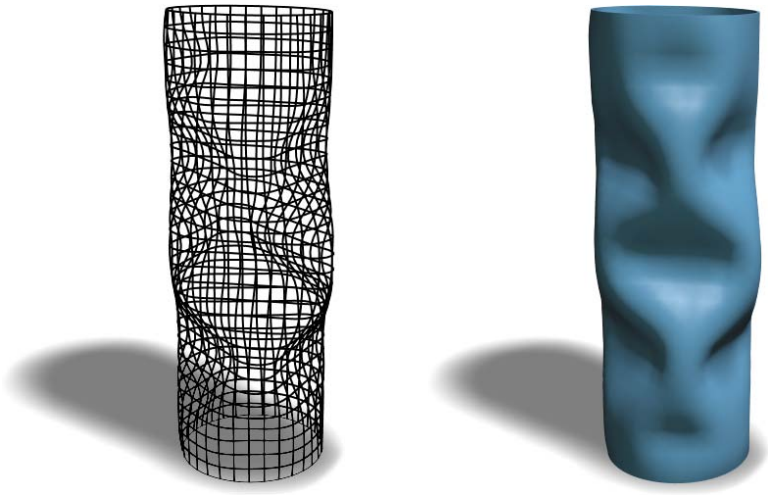

Figure 5.3: *Corotational DG FEM shells allow for the simulation of geometrically-nonlinear phenomena such as buckling (961 quad elements, 2945 nodes).*

The DG FEM discretization of Kirchhoff-Love shells described so far allows for simple C^0 basis functions and shows a plausible, geometrically nonlinear deformation behavior. However, to enable the simulation of highly detailed cuts or creases, as shown in Fig. 5.1, without excessively refining the underlying simulation mesh, we propose an XFEM approach that is described in the following sections.

5.3 XFEM Basics

Introduced in Belytschko and Black [1999], the extended finite element method (XFEM) builds up on the partition of unity concept [Melenk and Babuska, 1996]. The basic idea of XFEM consists in *enriching* an element by splitting its basis functions along a desired discontinuity. This effectively doubles the element's degrees of freedom and decouples the solutions on either side of the discontinuity. In an elasticity simulation, this allows a single element to be cut or fractured into two independent parts.

Splitting the original basis functions along the discontinuity results in the *canonical basis*, as shown for a 1D example in Fig. 5.4. We can define the enriched basis functions as the product of an original basis function $N^a(\boldsymbol{\xi})$ and a so-called *enrichment function* $\psi^a(\boldsymbol{\xi})$. Arguably the simplest choice for the enrichment functions $\psi^a(\boldsymbol{\xi})$ is the Heaviside function $H_s(\boldsymbol{\xi})$, which assumes the value of 0 on one side of the cut and 1 on the other side.

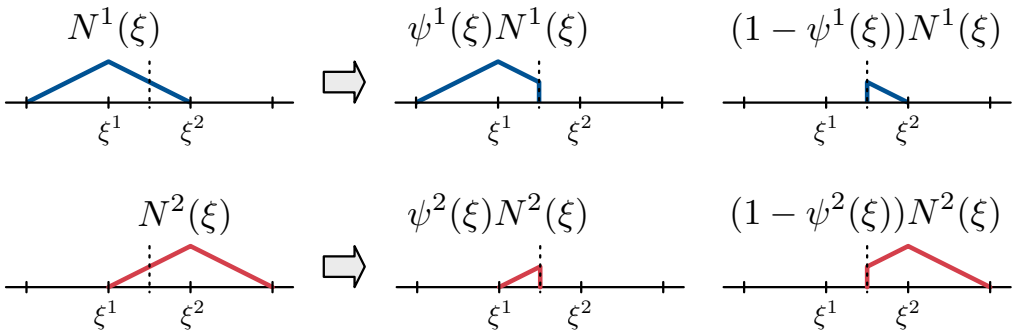

Figure 5.4: *The canonical basis: basis functions N^a are split along the discontinuity to handle both parts separately.*

Note that the function space spanned by $\psi^a N^a$ and $(1-\psi^a)N^a$ is the same as the one spanned by N^a and $\psi^a N^a$, and it does not depend on which side of the discontinuity we choose the Heaviside function to vanish. The splitting of the basis functions can therefore be realized by adding to the original basis functions N^a the *enrichment basis functions* $\psi^a N^a$ with their corresponding degrees of freedom $\mathbf{a}^a$. This yields the enriched displacement field

$$\mathbf{u}(\boldsymbol{\xi}) = \sum_{a=1}^{n} N^a(\boldsymbol{\xi})\mathbf{u}^a + \sum_{a=1}^{n} \psi^a(\boldsymbol{\xi})N^a(\boldsymbol{\xi})\mathbf{a}^a. \tag{5.25}$$

XFEM therefore modifies the functional representation on a sub-element level instead of changing the topology of the element mesh. While traditional methods would have to divide existing elements into smaller elements in order to resolve the geometry of the cut, XFEM achieves the same goal without complex remeshing.

Instead of the simple Heaviside function $H_s(\boldsymbol{\xi})$, Zi and Belytschko [2003] proposed to use *shifted enrichment functions*

$$\psi^a(\boldsymbol{\xi}) = H_g(\boldsymbol{\xi}) - H_g(\boldsymbol{\xi}^a), \tag{5.26}$$

where $H_g(\boldsymbol{\xi}) = 2H_s(\boldsymbol{\xi}) - 1$ is the generalized Heaviside function (signum function), and $\boldsymbol{\xi}^a \in \Omega$ is the position of node a in parameter space. See Fig. 5.5 for a 1D illustration. The shifted enrichment basis functions together with the original basis functions span the same space as in the case of the unshifted enrichment functions. They are thus equivalent to the canonical basis.

Defining the enrichment functions this way has two desirable properties. First, it keeps the enrichment local, in the sense that the enrichment basis functions are only supported within the cut element. Compare this to Fig. 5.4, where the enrichment basis functions extend to all elements incident to the

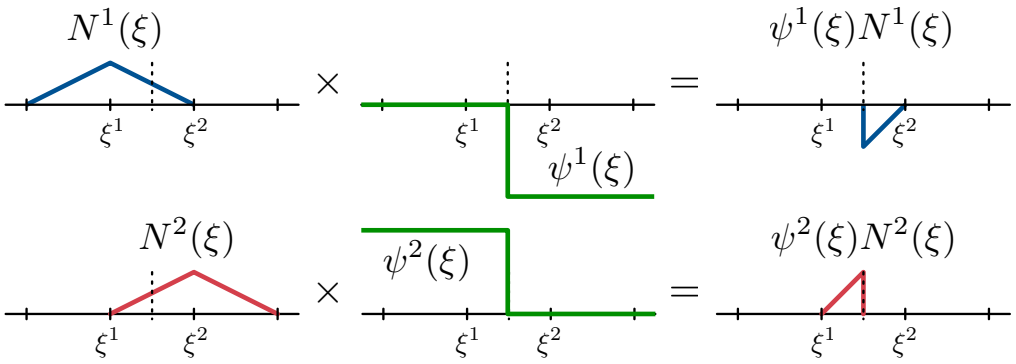

Figure 5.5: *Enrichment basis functions are obtained by multiplying the original basis functions by shifted Heaviside functions.*

enriched node. This locality simplifies implementation and, according to Jerabkova and Kuhlen [2009], also improves the stability of the simulation.

Second, shifting the enrichment functions recovers the Kronecker-delta property. The shifted enrichment function ψ^a vanishes at its associated node a, and as the nodal basis function N^a vanishes at all nodes other than a, the enrichment basis functions $\psi^a N^a$ vanish at *all* nodal positions $\boldsymbol{\xi}^a$. As a consequence, the displacement at each node a is fully defined by $\mathbf{u}^a$ alone. This simplifies the implementation of Dirichlet boundary conditions, as the displacement of a node can be fixed to a specific value by constraining a single degree of freedom.

5.4 Enrichment Textures

Many researchers have considered various ways to generalize the Heaviside function H_g and the shifted enrichment functions ψ^a from 1D to 2D or 3D. In most approaches, discontinuities are represented by simple analytical functions, such as cutting planes for 3D elements and linear or quadratic cutting paths for 2D elements. This representation, however, is too restrictive for graphics simulations, as it prevents us from having highly detailed, complex cuts through rather coarse elements of the simulation mesh.

To overcome this limitation, we represent the cut path as well as the generalized Heaviside function as piecewise linear functions on a regular 2D grid, which will be referred to as *enrichment texture* in the following. Discontinuities will be represented as an edge path between texture pixels (*texels*). The texture values are interpreted as (samples of) the generalized Heaviside function $H(\boldsymbol{\xi})$, and replaces $H_g(\boldsymbol{\xi})$ in (5.26).

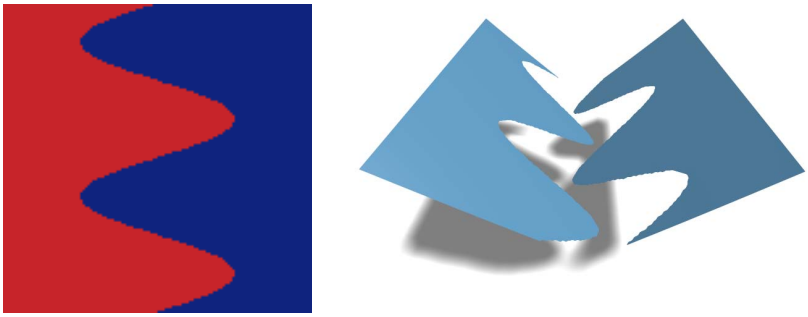

Figure 5.6: *A complete cut through a* single *element. The enrichment texture* (128×128) *is a generalized Heaviside (left), taking on values of* -1 *(red) or* $+1$ *(blue) on each side of the cut. A simulation of this enriched element is shown on the right.*

Defining the enrichment function via a texture map has two important advantages. First, compared to analytic descriptions, it is better suited for modeling complex cutting paths, since the complexity of cuts is only limited by the texture resolution. Second, when dealing with progressive cuts, partial cuts, or multiple, intersecting cuts—required for cutting and fracturing simulations—it allows to handle the generation of the associated enrichment functions in a straightforward and uniform manner.

In this section we will focus on *complete cuts*, i.e., cuts that completely split an element into two pieces, or into several pieces in case of multiple cuts. The more complex problem of partial and progressive cuts will be discussed in Section 5.5.

5.4.1 Single Cut

In the simplest case, for a single, complete cut through an element, the enrichment texture stores a value of -1 or $+1$ for each texel, indicating which side of the cut the texel lies on (see Fig. 5.6, left). This defines the Heaviside $H(\boldsymbol{\xi})$, which we use to construct the enrichment functions $\psi^a(\boldsymbol{\xi})$ as in (5.26). Multiplication with the original basis functions yields the enriched basis functions $\psi^a(\boldsymbol{\xi})N^a(\boldsymbol{\xi})$. Their partial derivatives can easily be computed by finite differences on the regular texture grid. The functions and their gradients are then integrated over the element to construct $\mathbf{K}$ and $\mathbf{F}$, as shown in (5.5). To this end, the numerical integrator can simply sample the texture values at the quadrature points. These must be chosen densely enough so that all features of the enrichment texture are captured. In the limit, this corresponds to one quadrature point per texel.

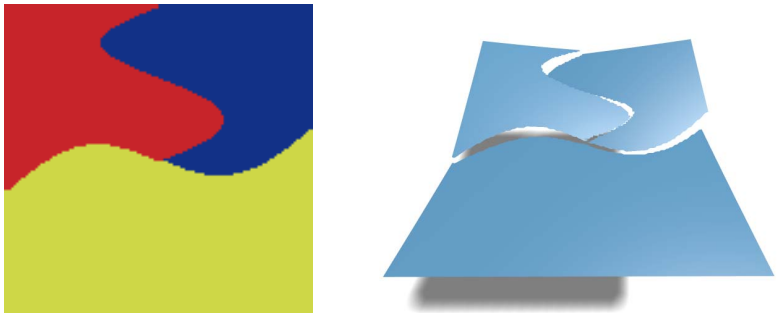

Figure 5.7: *Two complete, intersecting cuts through a single element, resulting in three separate connected components.*

5.4.2 Multiple Cuts

When considering *multiple* cuts within a single element, a cut is called complete if it starts and ends either at the element's boundary or at a junction with another cut. Multiple complete cuts therefore decompose the element into several connected components C_i.

The straightforward generalization of the canonical basis (see Fig. 5.4) is defined in terms of *canonical Heaviside* functions $H_c^i(\boldsymbol{\xi})$, which are 1 within their corresponding component C_i and 0 everywhere else. From these functions one can construct the shifted enrichment functions and from there on proceed like in the case of a single cut. Fig. 5.7 shows an example of two intersecting complete cuts, resulting in three physically independent surface components.

Note that thanks to our discrete texture representation the identification of connected components can be done by a trivial flood fill algorithm. In contrast, when representing cuts analytically, this task is more complex, such that often a hierarchical representation and classification of cuts has to be used.

5.4.3 Rendering

During a dynamic simulation, (4.11) is integrated in time, which in each time step yields the displacement $\mathbf{u}(\boldsymbol{\xi})$ at the spatial resolution of the enrichment textures. For rendering the deformed and cut shell surface, we tessellate each element of the simulation mesh into a fine triangulation, in order to more accurately approximate the quadratic displacement function $\mathbf{u}(\boldsymbol{\xi})$ as well as the high-resolution cuts. Note, however, that each element uses the same *static* triangulation, thereby minimizing storage and overhead.

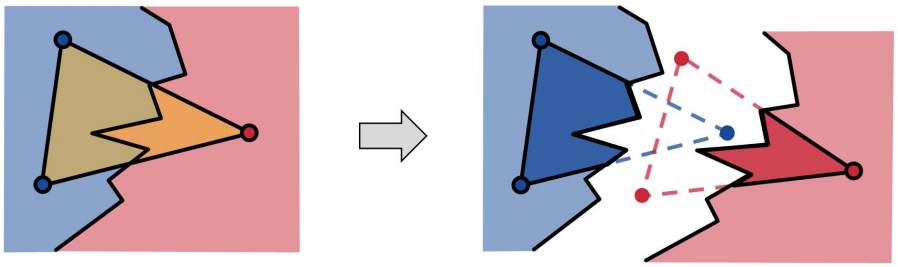

Figure 5.8: *A cut triangle (left) is duplicated and rendered once for each side of the cut (right)*

One option would be to generate a vertex for each texel, and have the triangulation implicitly defined by the regular texture grid, which would be conceptually similar to geometry images [Gu et al., 2002]. However, in order to reduce the number of triangles, while still being able to resolve features at the texel level, we employ a geometry shader. This shader duplicates triangles that have been cut: triangle vertices that lie on the opposite side of the cut are linearly extrapolated, and the triangle is then rendered with a fragment shader that masks out texels on the opposite side of the cut (see Fig. 5.8). An edge of a triangle is considered to be cut if two criteria are

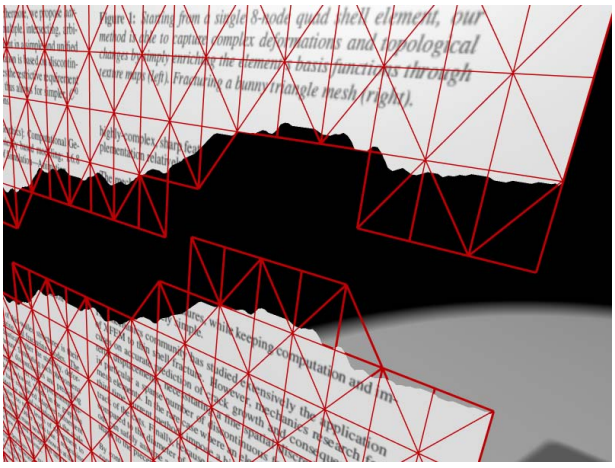

Figure 5.9: *Visualization of the underlying triangle mesh (red wireframe). Note the duplication of triangles that are intersected by the cut.*

met. First, the magnitude of the difference between the enrichment function values at the edge's vertices must exceed a certain threshold, and second, the sign of the enrichment function must be different at these two vertices. Fig. 5.6 shows an example of a complete cut through a single element, using an enrichment texture of resolution 128×128. Note that this approach limits the resolution of the cut to the texture resolution. Using texture filtering, the

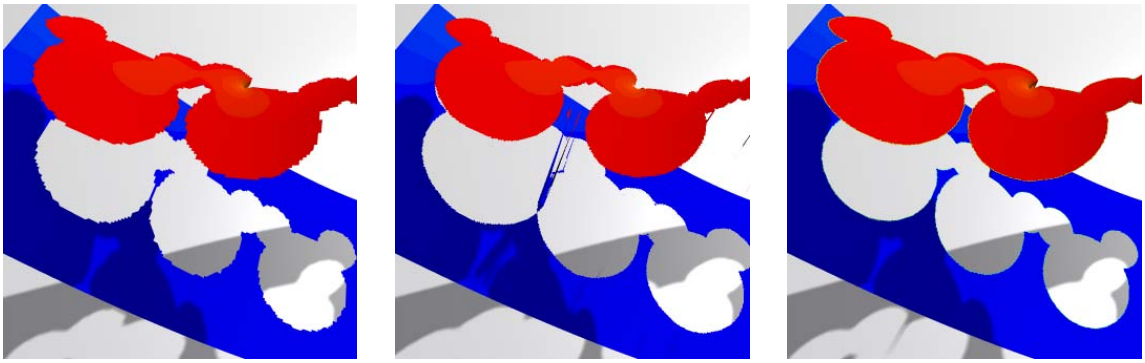

Figure 5.10: *Comparison of rendering techniques. Removing triangles intersected by the cut (left), killing pixels close to the cut curve (middle), and duplicating cut triangles while employing a geometry shader (right).*

resulting aliasing artifacts can be mitigated to some degree. Fig. 5.9 visualizes the underlying triangle mesh used to render an object consisting of a single element. To demonstrate the benefits of geometry shader based rendering, Fig. 5.10 compares simple triangle or pixel removal to the proposed approach, which is free of artifacts while keeping the width of the cut below the size of one texture pixel.

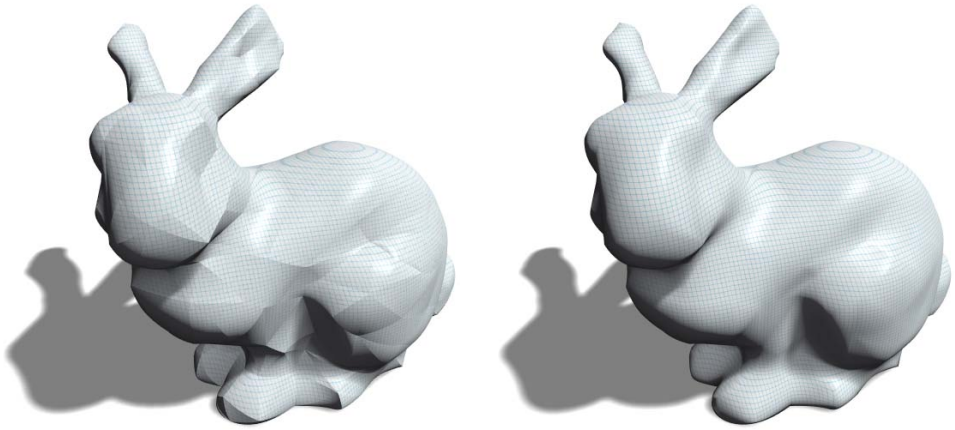

Figure 5.11: *Rendering each element independently highlights the discretization and C^0 continuity (left), while a simple averaging of normals at element nodes can result in a visually more pleasing representation (right).*

Visual Surface Smoothness. As the initial surface as well as the deformed mesh are only required to be C^0 continuous, a direct rendering of the physical simulation mesh can result in shading artifacts such as the creases in Fig. 5.11, left. If visual quality is more important than an accurate representation of the

simulation geometry, vertex normals can be averaged between elements and interpolated inside elements to get the shading of a C^1 continuous surface as shown in Fig. 5.11, right.

5.5 Progressive Cutting

After showing the simplicity and flexibility of enrichment textures for *complete* cuts, we now discuss the more interesting case of *partial*, *progressive* cuts. In this case, a cut does not (yet) fully separate the element into two independent components.

For such configurations, works in the mechanics literature typically require to classify a discontinuity into either a *tip* (partial cut), a *complete* cut, or a *junction* (intersecting cuts), and generate special enrichment functions depending on the type of discontinuity. When furthermore allowing for multiple cuts within a single element, with cuts being either partial or complete, the combinatorics of handling these different cases in a hierarchical representation inevitably leads to rather complex implementations.

In this section we propose our *harmonic enrichment* approach, which is conceptually simpler, in that it uses only one kind of unified enrichment function that does not depend on the type of cut, and that generalizes to multiple, partial, progressive, and complete cuts in a natural and canonical manner. We again start with a single cut, then discuss multiple cuts within one element, and describe how to handle cuts that intersect multiple elements.

5.5.1 Single Cut

Since a partial cut does not fully split the element into two components, separation of material is only allowed along the curve covered by the cut so far. However, as soon as a progressive cut has traversed the whole element and hence became a complete cut, we want the parts on either side of the cut to be independent of each other, as shown in the previous section. This translates into two conditions for the generalized Heaviside function:

- For a partial cut, the enrichment (and the Heaviside) should be discontinuous along the cut but continuous everywhere else, allowing material to separate at the cut only.
- If the progressive cut becomes a complete cut, the value of the enrichment function must be constant on either side of the cut, replicating the behavior of the Heaviside functions of the previous section.

Figure 5.12: *A* partial *cut in a single element: The corresponding Laplace problem (left), the resulting harmonic enrichment texture (center), and the element behavior in a simulation (right).*

In mechanics, one of the classical crack tip functions is $\sqrt{r}\sin(\theta/2)$ [Belytschko and Black, 1999], where (θ, r) are polar coordinates centered at the crack tip. This function takes on values of $\pm\sqrt{r}$ on either side of the crack. However, this function does not easily generalize to complex cut shapes and multiple cuts within a single element. We therefore generalize the classical enrichment function by observing that it is a harmonic function and defining *harmonic enrichments* as the solution to the Laplace equation

$$\Delta H(\boldsymbol{\xi}) = 0, \tag{5.27}$$

subject to suitable boundary conditions along the crack tip and the element boundary. On the element boundary, we prescribe vanishing Neumann constraints $\nabla H(\boldsymbol{\xi}) \cdot \mathbf{n}(\boldsymbol{\xi}) = 0$ (see Fig. 5.12).

Along the crack, prescribing the Dirichlet conditions $H(\boldsymbol{\xi}) = \pm\sqrt{r}$ recovers the classical harmonic enrichment function $\sqrt{r}\sin(\theta/2)$ in the vicinity of the crack tip as demonstrated in Fig. 5.13. In particular, since a harmonic reconstruction reproduces given boundary conditions (tautologically), it follows that the influence of any other finitely-distant boundary can be ignored in a sufficiently small neighborhood of the tip. As a simplification, we assume that the $\sqrt{r}$-decay is concentrated within the crack tip texel; thus, we prescribe the Dirichlet conditions $H(\boldsymbol{\xi}) = \pm 1$ on the texels incident to the cut. The simplified condition works well in our experiments; however, if adaptive refinement of the enrichment texture is used, the $\sqrt{r}$-decay should be explicitly considered, since the Dirichlet conditions ± 1 are ill-posed in the smooth limit.

Thanks to our texture representation, (5.27) can be discretized by a simple finite difference scheme. As the cut corresponds to a series of connected edges between texels, the Dirichlet conditions set the texels incident to a cut edge to -1 or 1, respectively. The resulting sparse linear system has to be

Figure 5.13: *Comparison between the function $\sqrt{r}\sin(\theta/2)$ (left) and the finite difference solution of $\Delta H(\boldsymbol{\xi}) = 0$ with Dirichlet boundary conditions $H(\boldsymbol{\xi}) = \pm\sqrt{r}$ on the cut (right). The two solutions are identical up to discretization errors.*

solved for the function values of $H(\boldsymbol{\xi})$ at each texel, which we do using either a multigrid solver or a sparse Cholesky factorization [Toledo et al., 2003].

As the solution of a Laplace equation, $H(\boldsymbol{\xi})$ will be harmonic, i.e., continuous and smooth, everywhere except at the Dirichlet constraints, where it shows the desired discontinuity. Hence, the first requirement is fulfilled. In the case of a complete cut, the vanishing Neumann boundary conditions cause $H(\boldsymbol{\xi})$ to replicate a Heaviside with constant ± 1 on either side of the cut. Furthermore, as the cut approaches the element boundary, the enrichment function converges to this Heaviside in a temporally smooth manner. As a consequence, the second criterion is satisfied. The enrichment texture of the partial cut in Fig. 5.12 would eventually converge to the one of the complete cut shown in Fig. 5.6.

5.5.2 Multiple Cuts

With just a slight modification of the boundary constraints, the harmonic enrichment approach can be generalized from single cuts to multiple cuts within an element. Consider n (partial or complete) cuts $\mathbf{c}_i(\xi)$, $1 \leq i \leq n$, in an element. We have to construct an enrichment function for each cut, i.e., we have to find a generalized Heaviside $H^i(\boldsymbol{\xi})$ for each cut $\mathbf{c}_i(\xi)$. The two requirements formulated for single cuts can be extended to multiple cuts as follows:

- The enrichment function $H^i(\boldsymbol{\xi})$ should be discontinuous across its cut $\mathbf{c}_i(\xi)$, and continuous for unconstrained texels. $H^i(\boldsymbol{\xi})$ might be discontinuous across $\mathbf{c}_j(\xi)$ for $j \neq i$.
- If all cuts are *complete* cuts, they partition the element into $n+1$ separate components $C_1, \ldots, C_{n+1}$. The original basis functions and their

Figure 5.14: Multiple progressive *cuts within an element: The harmonic enrichment textures $H^i(\boldsymbol{\xi})$ for each of the cuts (left, center), and the resulting simulation behavior (right).*

n enriched versions should reproduce the behavior of the canonical basis (see Fig. 5.7), i.e., they should span the same function space.

We extend the definition of harmonic enrichment functions for multiple cuts as follows: each enrichment function $H^i(\boldsymbol{\xi})$ is again defined as the solution of

$$\Delta H^i(\boldsymbol{\xi}) = 0. \tag{5.28}$$

As in the single cut case, the discontinuity across its corresponding cut $\mathbf{c}_i(\xi)$ is enforced through Dirichlet conditions of ± 1. However, now we enforce vanishing Neumann conditions not only on the element boundary, but also on all other cuts $\mathbf{c}_j(\xi)$, $j \neq i$, i.e.

$$\nabla H^i(\boldsymbol{\xi}) \cdot \mathbf{n}(\boldsymbol{\xi}) = 0, \tag{5.29}$$

where $\mathbf{n}$ is the normal of either the domain boundary or another cut $\mathbf{c}_j(\xi)$ respectively. Note that there might be components with Neumann conditions only. These components are independent of the cut $\mathbf{c}_i$ under consideration, and we therefore explicitly set the function values in these regions to zero. The resulting Heaviside functions $H^i(\boldsymbol{\xi})$ again trivially satisfy the first condition by construction. A simple proof showing that the second condition is also fulfilled is given in Appendix A.2. See Fig. 5.14 for an example of two partial cuts, which would eventually converge to the situation depicted in Fig. 5.7.

Harmonic enrichment textures therefore allow us to handle both single and multiple cuts, as well as partial, progressive, and complete cuts in a simple, unified, and canonical manner. In contrast to existing work, we do not require cuts to be classified into crack tips, joints, or complete cuts. Neither do we need a complex hierarchical representation to keep track of multiple cuts and the resulting connected components.

5.5.3 Multiple Elements

So far, we described how to handle single or multiple, partial or complete cuts within a *single element*. For realistic simulations, however, a simulation mesh of more than one element is of course required. In this section we extend our enrichment approach to multiple elements partitioning the simulation domain Ω. A 2D illustration of a multi-element mesh with two complete cuts $\mathbf{c}_1$, $\mathbf{c}_2$ and one partial cut $\mathbf{c}_3$ is shown in Fig. 5.15.

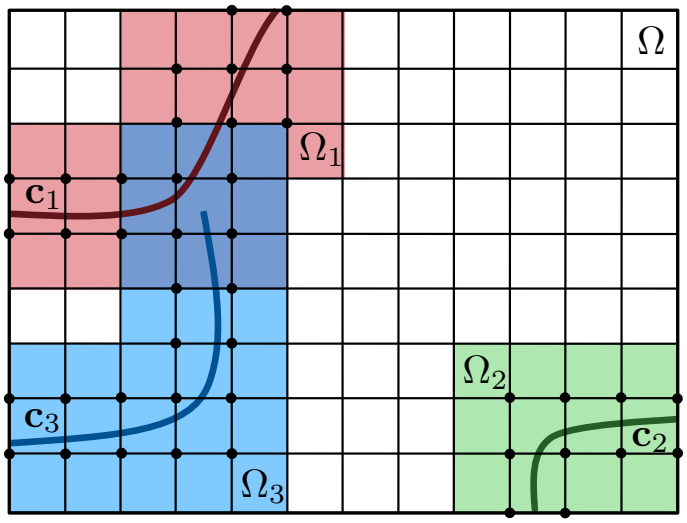

Figure 5.15: *Three cuts define nodes to be enriched (dots) and associated domains Ω_i on which the H^i are defined. Nodes in overlapping domains are enriched multiple times.*

In order to model the discontinuities caused by a cut, all elements intersected by this cut are enriched. This is achieved by enriching the nodal basis functions $N^a(\boldsymbol{\xi})$ of each of the element's vertices a, resulting in $\psi^a(\boldsymbol{\xi})N^a(\boldsymbol{\xi})$. This expression requires the enrichment function ψ^a, and therefore the Heaviside H (see (5.26)), to be defined on the whole support of N^a, which in our case is just the set of elements incident to vertex a. The (overlapping) colored regions Ω_i in Fig. 5.15 highlight the elements that are enriched due to the respective cuts $\mathbf{c}_i$, the dots represent the vertices to be enriched.

The main difference to the single element case is that the enrichment functions of neighboring elements have to be C^0 continuous across their shared edge, in order to guarantee the enrichment basis functions $\psi^a N^a$, and thus the enriched displacement $\mathbf{u}$ (see (5.25)), to be C^0, as required by our DG FEM formulation.

For *complete cuts*, such as $\mathbf{c}_1$ and $\mathbf{c}_2$ in Fig. 5.15, we can use the simple enrichment functions shown in Figs. 5.6 and 5.7 for the intersected elements,

and constant enrichments of $+1$ or -1 for their neighboring elements. The resulting enrichment functions will be C^0 across mesh edges by construction.

The handling of *partial cuts*, such as $\mathbf{c}_3$ in Fig. 5.15, is slightly more involved. For the element containing the cut tip a harmonic enrichment is computed by solving (5.27) or (5.28), as discussed in Section 5.5.2. The vanishing Neumann boundary conditions for this system can, however, not guarantee C^0 continuity to its neighboring elements (as Dirichlet boundary constraints could do).

In order to ensure continuity, we additionally have to take the incident elements into account and solve the Laplace equation on this extended one-ring neighborhood (3×3 region around the tip of $\mathbf{c}_3$ in Fig. 5.15). On the boundary edges of this region we prescribe the values of neighboring enrichment functions, if they exist, as C^0 Dirichlet constraints, or vanishing Neumann conditions otherwise. As the neighboring enrichment function values were computed in previous simulation steps, this construction is free of cycles.

In Fig. 5.15 we would therefore compute the enrichment function(s) H^3 for the nine elements around the cut tip by a single Laplace system, prescribing ± 1 Dirichlet constraints along $\mathbf{c}_3$, vanishing Neumann constraints along $\mathbf{c}_1$, C^0 Dirichlet constraints on the boundary to the blue elements, and vanishing Neumann constraints on the other boundaries. In this 3×3 region the enrichment function H^1 for cut $\mathbf{c}_1$ also has to be computed through this system, by exchanging the roles of $\mathbf{c}_1$ and $\mathbf{c}_3$. With this slight modification of the domain and the constraints of the Laplace systems (5.27), (5.28), we are able to generalize the harmonic enrichment textures from single elements to arbitrary simulation meshes, including quad and triangle meshes through the definition of appropriate texel neighborhoods at the element boundaries.

Note that our DG FEM formulation considerably simplifies this generalization as it only requires C^0 continuity on element edges. On the other hand, conforming thin shell discretizations using C^1 basis functions [Cirak et al., 2000; Thomaszewski et al., 2006] would require C^1 continuous enrichment functions, which cannot be computed through simple Laplace systems.

5.6 Results

We demonstrate our enrichment method on a variety of examples, including single elements, more complex element meshes, single progressive cuts, cuts consisting of multiple enrichments, and multiple cuts per element. The simulation meshes used consist of either 8-node bi-quadratic quadrilateral

elements or 6-node quadratic triangular elements. All discontinuities were prescribed as polylines and then rasterized to texel edges.

Complex Cut Lines. Our method allows us to represent discontinuities on a fine sub-element level without the need to remesh. Fracture lines are a special instance of such discontinuity lines and are usually computed using local strain or stress measurements to predict their evolution. We do not directly simulate a complex fracturing model but generate the highly-detailed cut boundaries procedurally. Our approach is able to resolve the shells' reaction to the fracture discontinuity at the sub-element level, as shown in Fig. 5.16.

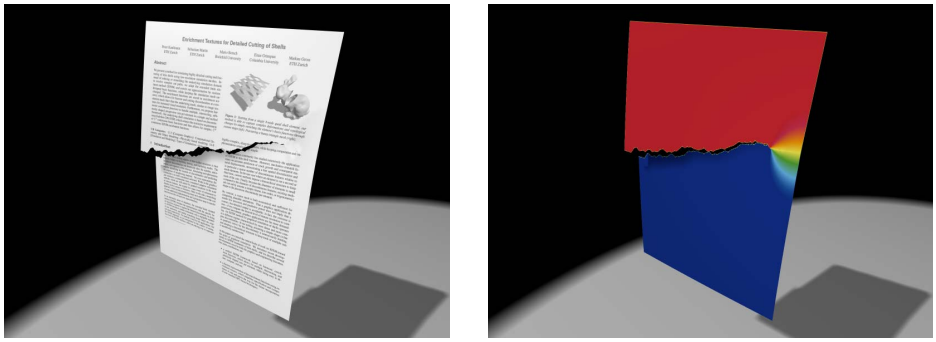

Figure 5.16: *A single textured quad element being fractured (left), and the underlying discontinuous enrichment function (right).*

Fig. 5.1, right, shows the same model applied to a non-trivial mesh with triangular quadratic elements. Note that the elements are shaded independently of each other in order to show the coarse resolution of the simulation mesh. To avoid these shading discontinuities, normals can be averaged at nodes and interpolated over the elements as described in Section 5.4.3.

Small Features. The presented approach allows us to simulate small scale details down to the resolution of the texture. In Fig. 5.17, left, a circular cut is applied to a planar shell mesh consisting of four quads. The method can robustly handle the situation where the connection between the "hanging chad" and the rest of the element is only a few texels wide. Without requiring any special handling, the basis functions resulting from the harmonic enrichments introduce a specific additional mode of deformation, leading to the expected result. Fig. 5.17, right, shows an example of a highly-detailed cut line demonstrating the capability to resolve small features.

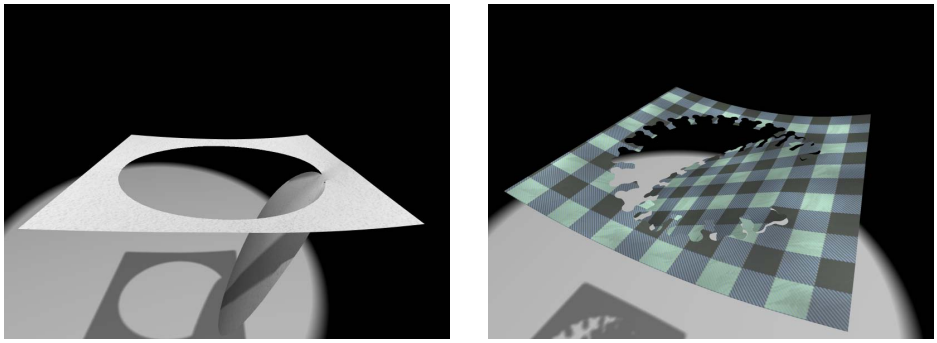

Figure 5.17: *Simulation of a chad failing to completely separate from a punchcard (left). Highly detailed cut on a tissue (right).*

Multiple Cuts. As described in Section 5.5.2, our method is also applicable to the case of multiple cuts per element. Fig. 5.1, left, shows a single quad element that has been enriched 18 times, replicating a "paper accordion". In this example, the boundary conditions on the cuts are not purely ± 1 on either side of the cuts, but instead are blended to zero towards both ends of each cut to get an enrichment that better captures the desired deformation.

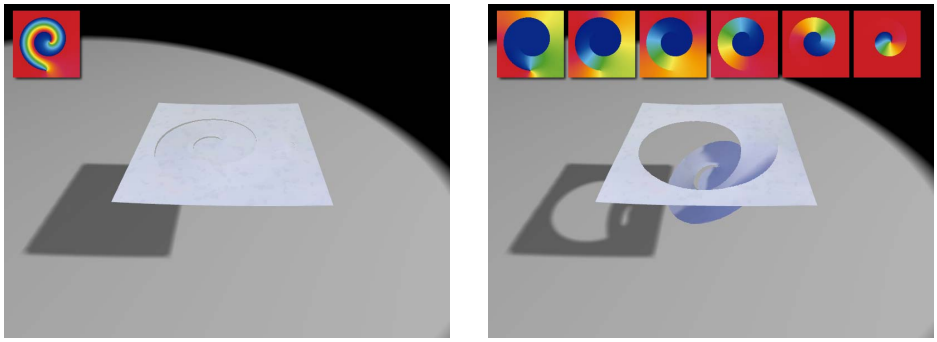

Figure 5.18: *A quad element is cut by a helical line, using a single enrichment (left) and six cut segments (right). The insets show the enrichment functions used.*

For long progressive cuts with complex trajectories, a single enrichment will not provide sufficient additional degrees of freedom to allow for the desired deformation. Such cases can be handled consistently by tracking the length or integral angle of the cut and by initiating a new cut once a predefined threshold is exceeded. While this criterion is mathematically not rigorous, it works very well in practice, and we employed it for our experiments.

This results in a series of connected cuts that together make up the desired discontinuity and add the required local degrees of freedom. Fig. 5.18 shows an example of a spiral cut enriched by a single enrichment texture (left), lacking the desired deformation, and the same curve divided into 6 cuts

(right), resulting in a more plausible simulation. Applying the same principle to a less trivial mesh, a slinky is cut out of a cylinder in Fig. 5.19, left. Fig. 5.19, right, additionally shows an example of intersecting cuts. Note that once a cut crosses another cut, it becomes a complete cut that ends at the point of intersection and a new cut is started.

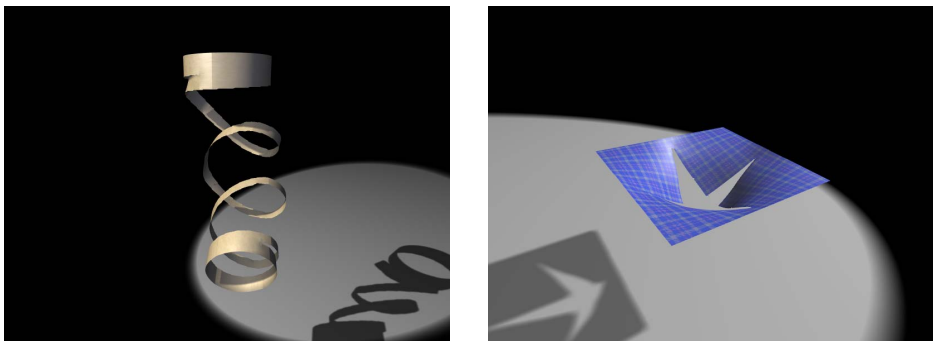

Figure 5.19: *A cylinder represented by four quad elements is cut into a slinky, deforms under gravity and uncoils (left). Simulation of intersecting cuts in a single element (right).*

Material Textures. As an alternative application of textures besides cutting, we can define material properties at the texel level. In the case of Kirchhoff-Love shells, properties that can be modulated by such a *material texture* include the material's Young's modulus, Poisson's ratio, as well as the local thickness of the shell. During the integration of an element's stiffness matrix, the material texture is evaluated at the quadrature points. Fig. 5.20 shows an example of a single shell element with locally reduced thickness, causing it to weaken and exhibit a stronger deformation under gravity. In this example, 12^2 quadrature points were used to integrate the element's stiffness matrix.

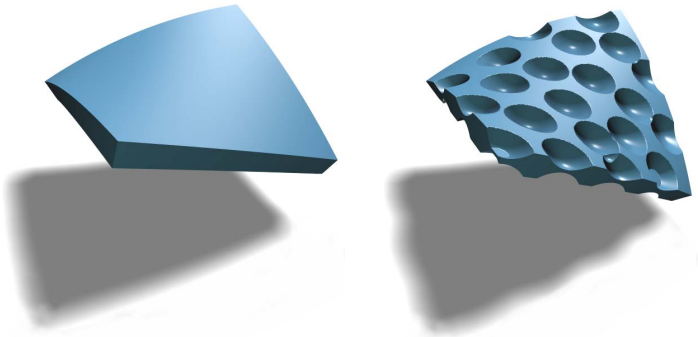

Figure 5.20: *The thickness of a shell element (left) is modulated by a thickness texture and deforms accordingly (right).*

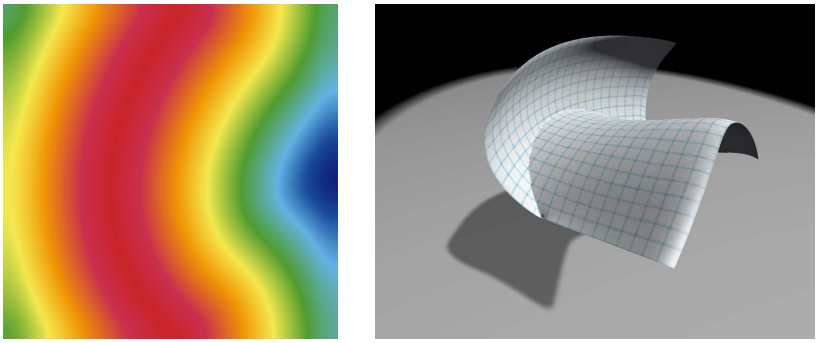

Figure 5.21: *A C^0 continuous enrichment texture with discontinuous first derivatives (left) allows adding a crease to an element (right).*

Creases. So far, we have only considered enrichment textures with discontinuities, allowing for cutting and separating shells into multiple independent components. However, we note that the concept of enrichment textures is more general. In particular, it can also be used to model sharp creases, at which the shell is free to bend. Creases can be modeled by keeping the enrichment textures C^0 continuous, but introducing discontinuities in the first derivatives. Fig. 5.21, left, shows an example of an enrichment texture for a curved crease line. The enrichment function is computed as the geodesic distance from the crease using a fast marching method [Sethian, 1999]. Applying this enrichment texture to a shell element allows the element to bend around the crease, resulting in interesting states of minimal energy when external forces are applied (see Fig. 5.21, right).

Model	#Els.	#Nodes	#Enr.	Tex.	t_{enr}	t_{asm}	t_{sol}
Chad	4	21	1	256	1433	16	3
Spiral 1	1	8	1	256	274	1.8	0.1
Spiral 6	1	8	6	256	1701	38	0.7
Accordion	1	8	18	256	4785	313	8
Bunny	552	1106	1	32	283	452	917
Slinky	4	20	12	128	5207	513	22

Table 5.1: *Comparison of timings for the computation of the enrichment functions (t_{enr}), assembly of the global stiffness matrix (t_{asm}), and solving the resulting linear system (t_{sol}). Timings are in milliseconds and were taken on an Intel Core2 Duo 2.4 GHz.*

Timings. Table 5.1 shows the computation times per simulation step for a representative selection of examples. The times for computing the enrichments are peak values, while the assembly and solve times were averaged

over the whole simulation. For solving the Laplace problem during the computation of the enrichment functions, we employ a sparse Cholesky solver [Toledo et al., 2003].

Comparison to Discrete Shells. For typical meshes the cost of cutting and enrichment is only a small percentage of the total simulation time, e.g., 7% for the Bunny example. This percentage vanishes as mesh resolution increases, since the cost of dynamics increases while the texture area intersected by the cut decreases.

Focusing on the bulk of computational cost—simulating dynamics after cutting—Table 5.2 compares our method to Discrete Shells [Grinspun et al., 2003] in terms of computational effort. We use the number of non-zero entries in the stiffness matrix (NNZ) as a measure of complexity for solving the linear system during time integration. For Discrete Shells, the mesh resolutions were chosen to match the NNZs of our enriched simulations.

For the Discrete Shells simulations, the mesh resolution might be sufficient to simulate the elastic behavior, but it is not enough to resolve the cuts in sufficient detail, especially for highly detailed cuts. By contrast, our enrichment uses the same number of additional DOFs, whether the cut is highly jagged or straight. The comparison shows that our method excels in settings characterized by a higher ratio of cut to deformation complexity.

Mesh (Method)	#Els.	#Nodes	#DOFs	#NNZ
Bunny (our method)	552	1106	3318	243954
Bunny (Discrete Shells)	4200	2102	6306	245934
Spiral (our method)	1	8	168	28224
Spiral (Discrete Shells)	385	274	822	32058
Chad (our method)	1	8	48	2304
Chad (Discrete Shells)	72	62	186	7254

Table 5.2: *Comparison of our method to Discrete Shells with a comparable computational budget (non-zero stiffness matrix entries).*

Limitations. The previous examples show that our method is able to represent complex cut lines and small features in single elements and element meshes, and that multiple cuts per element can be handled consistently.

However, we note that our current implementation of the method lacks a proper projection step. When the basis functions change due to enrichment, we simply apply the previous solution to the new basis. In most situations

the introduced error is unnoticeable, since the enrichments change gradually over time. However, in some scenarios it leads to obvious "popping" artifacts, e.g., in the multiply enriched spiral of Fig. 5.18, right. These artifacts may be eliminated by projecting the displacements and velocities onto the new basis, as proposed, e.g., in Réthoré et al. [2005].

5.7 Discussion and Outlook

We presented a novel method for handling highly detailed discontinuity features such as cutting or fracture lines using a versatile, texture-based basis enrichment approach. The method spends new degrees of freedom in an economical manner and supports a uniform and general treatment of multiple progressive or complete discontinuities. While the proposed method focuses on introducing material discontinuities, we hope that the presented way of combining texture concepts with physical simulations opens exciting new areas for future work.

A promising direction for future work is to generalize the modeling of creases, building on the preliminary results presented in this work. We would also like to explore the synergies of textures representing both geometric and physical material properties. For example, a displacement texture map could represent both the physical rest state of a shell as well as allow for high surface detail while still only requiring a coarse simulation mesh.

We further note that most of the steps in the simulation pipeline are very amenable to parallelization, making them ideal candidates for computation on the GPU. In particular, the computation of the enrichment functions can be performed efficiently using a multigrid solver, whose pre- and post-smoothing steps can be interpreted as simple texture filtering operations [Bolz et al., 2003]. The corotated stiffness matrix integration could be computed on the GPU, since it parallelizes trivially, leading to a highly accurate integration at the texel level. At the rendering stage, the deformed geometry could easily be constructed on the GPU as well, by computing the displacement field from the enrichment textures and the solution to the dynamic simulation.

The presented corotational extension of the linear discontinuous Galerkin shell method of Noels and Radovitzky [2008] allows for the simulation of nonlinear phenomena while still being efficient to compute. The use of DG FEM in this context not only provides a practical solution to the C^1 continuity problem of thin shells, it also simplifies the computation of enrichment textures for multi-element cuts.

So far, we have only applied DG FEM to linear problems (with corotation). The next chapter will present an application where the method is used to solve non-linear problems in the context of image warping.

CHAPTER 6

Non-Linear Image Warping

a b c d e

Figure 6.1: *Our unifying representation of image warping supports efficient adaptive meshing, high order basis functions, and more. a) Original image, b) automatic saliency map, c) aspect ratio change by uniform scaling, d) retargeting using uniform grid (6767 vertices), e) similar quality results using an adaptive mesh (1325 vertices).*

After having applied the DG FEM to the simulation of deformable solids and shells in the previous two chapters, we now show that the method can also be used to solve content-aware image warping problems in a single unifying framework that encompasses a wide range of image and video editing tasks. In this field, even standard FEM is not widely used yet. Instead, existing approaches define error terms over mesh edges or vertex finite differences. As will be shown, these can be expressed as a special case of the proposed FEM model.

We exploit the full generality of FEMs, gaining important advantages, such as: arbitrary mesh connectivity in the image domain, a well defined, continuous problem formulation, and well behaved convergence properties. These advantages are then used to demonstrate support for adaptive meshing, allowing for efficient temporally stable solutions, and robust and simple constraint substitution, allowing for fine level control over warping results. An FEM representation also naturally supports higher order basis functions, allowing for smoother warps even with a small number of elements.

Through the use of DG FEM, per-element basis functions of varying degree and complex mesh connectivity with hanging nodes can be used, which further increases the space of possibilities and simplifies the problem setup. The continuous problem formulation allows us to clearly analyze existing image warping error functions, and propose improved ones. The utility of the proposed method is demonstrated by showing examples in adaptive, temporally consistent content-aware video resizing and camera stabilization applications, and by comparing our results to previous state of the art methods.

6.1 Overview

In order to formulate image warping problems in a generic 2D FEM framework, we first need to understand their underlying continuum equations. It turns out that for most of the energy terms and constraints that have previously been used in the image warping literature, this is a pretty straight forward task. While the FEM is more general in theory, we limit ourselves to warp energies that are defined in terms of an energy density that only depends on position as well as first spatial derivatives of the warp. This allows us to directly apply the non-linear FEM theory presented in Section 3.2 and Section 3.3.

Formulating all warp energies as energy density functions will allow us to compare and classify them, and in some cases even to simplify them by coming up with new energies that have similar properties but a simpler mathematical formulation. The benefits of DG FEM will become obvious once we look at the actual discretization of the problem: one of the general advantages of FEM over finite difference methods is that FEM allows for adaptivity, so degrees of freedom can be spent where they are most needed, i.e. in areas where the solution changes rapidly. With DG FEM, one can even go one step further as the method naturally supports hanging nodes, which makes adaptive meshing (h-refinement) almost trivial. It additionally allows for easily combining elements with different polynomial degrees

(p-refinement) to further adapt the distribution of DOFs to the underlying problem.

Finally, FEM image warping is demonstrated in the domain of media retargeting, where the aspect ratio of images or videos is modified such that the shape of important content is preserved, as well as video stabilization, where a smooth reconstruction of hand-held shaky videos is computed. For video retargeting, adaptive solutions will be shown that allow us to solve a whole sequence of frames at once; a task that previously would be either computationally intractable, or have only low resolution control due to excessive subsampling.

6.2 FEM for Image Warping

In order to derive an FEM for image warping, we begin by formulating the general image warping problem in the continuous case, and then in Section 6.3 we will discuss application specific decisions. While previous approaches first set up the discretization using finite differences, regularly spaced grids, or triangle meshes, computing ad-hoc energy functions from the resulting primitives (vertices, edges), a continuous formulation allows us to study and compare the properties of various image warping energies independently of their discretization. We *then* perform the actual discretization by means of finite elements as a second, independent step, where we are presented with a multitude of choices, allowing us for example to trade accuracy (how well the continuous solution is approximated) for performance.

6.2.1 Continuous Warping

Consider the rectangular domain $\Omega = [0,1] \times [0,h] \subset \mathbb{R}^2$ of an undeformed image. The warping function $\boldsymbol{\varphi} : \Omega \to \mathbb{R}^2$ maps a point $\mathbf{X} = (X_1, X_2)^T \in \Omega$ to a warped point $\mathbf{x} = (x_1, x_2)^T = (\varphi_1(X_1, X_2), \varphi_2(X_1, X_2))^T = \boldsymbol{\varphi}(\mathbf{X})$ (see Figure 6.2). Following the notation introduced in Section 3.3, let the 2-dimensional *deformation gradient* be defined as the 2×2 matrix $\mathbf{F}(\mathbf{X})$ with entries:

$$F_{ij}(\mathbf{X}) = \left.\frac{\partial \varphi_i}{\partial X_j}\right|_{\mathbf{X}}$$

In order to define the cost of performing a certain warp $\boldsymbol{\varphi}$, we again make use of a function Ψ that computes the *deformation energy density* (per undeformed

area in 2D) at any point $\mathbf{X} \in \Omega$. The total deformation energy of a warp $\boldsymbol{\varphi}$ can then be computed as:

$$E[\boldsymbol{\varphi}] = \int_{\Omega} \Psi(\mathbf{F}(\mathbf{X}))$$

The optimal warp is the one that minimizes E, and respects a number of problem-specific boundary constraints defined in Section 6.3.

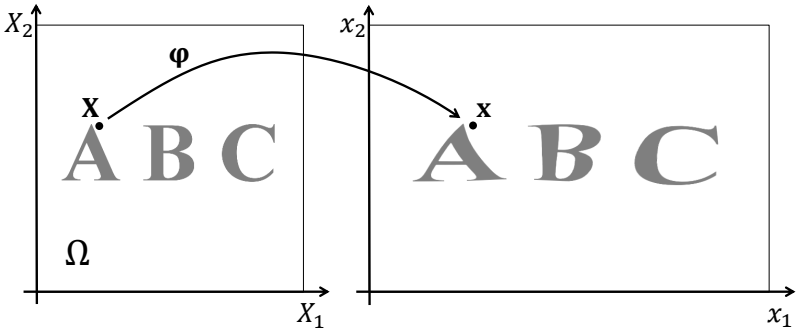

Figure 6.2: *Continuous mapping from the undeformed domain to the warped image using a simple nonuniform scale warp.*

6.2.2 FEM Discretization

Now that we have defined the continuous image warping problem, we discretize the problem using the FEM in order to numerically compute the optimal warp.

Basis Functions. As a first step, we proceed as described in Section 3.3.2 and discretize the warping function $\boldsymbol{\varphi}$ into a linear combination of n *basis functions* $N_i : \Omega \to \mathbb{R}$ with associated weights $\mathbf{x}_i = (x_1^i, x_2^i)^T \in \mathbb{R}^2$:

$$\boldsymbol{\varphi}(\mathbf{X}) \approx \sum_{i=1}^{n} \mathbf{x}_i N_i(\mathbf{X}) \tag{6.1}$$

Once the shapes of the individual basis functions have been defined, the warping function $\boldsymbol{\varphi}$ is fully determined through the values $\mathbf{x}_i$. The continuous problem of finding a function $\boldsymbol{\varphi}$ reduces to finding the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ that minimize E. The $\mathbf{x}_i$ are thus the degrees of freedom (DOF) of our optimization problem.

Mesh Representation. The domain Ω is represented as a mesh consisting of m elements $K^1, \ldots, K^m$, and q nodes at positions $\mathbf{X}_1, \ldots, \mathbf{X}_q$. Computing the derivative of Eq. (6.1) with respect to $\mathbf{X}$ gives us $\mathbf{F}$ as a linear combination of $\mathbf{x}_i$. The energy E can now be formulated as a sum over element integrals:

$$E = \sum_{k=1}^{m} \int_{K^k} \Psi \left(\sum_{i=1}^{n} \mathbf{x}_i \frac{\partial N_i}{\partial \mathbf{X}} \bigg|_{\mathbf{X}}^{T} \right) d\mathbf{X} \tag{6.2}$$

The per-element integrals can be approximated using a numerical quadrature rule. For implementation details, we refer to Hughes [2000]. What makes this computation efficient is the fact that basis functions N_i are *local*: inside any element K^k, only a constant number of basis functions can be non-zero, and the sum over i can be reduced to a sum over those basis functions.

If we treat each pixel of an image as an element with a single constant basis function, and introduce some additional terms to handle the resulting discontinuities between elements, we are in fact able to recover the standard finite difference scheme that is used in existing image warping methods.

Numerical Minimization. For general $\Psi(\mathbf{F})$, Eq. (6.2) is a non-linear equation in $\mathbf{x}_i$, and we can minimize it using a Newton method [Nocedal and Wright, 2000]. The Newton-Raphson method presented in Section 3.3.2 worked well in our experiments. For this, we need the first and second derivatives of E with respect to the $\mathbf{x}_i$, which we compute using either a code generation tool or automatic differentiation (AD). The second derivatives of E result in a sparse, symmetric matrix $\mathbf{H}$, which may not be positive-definite. Merging all the vector DOFs $\mathbf{x}_i$ into one big vector $\mathbf{d}$ of length $2n$ and denoting the first derivative of E by $\mathbf{f}$, a single Newton step computes the increment $\Delta\mathbf{d}$ of $\mathbf{d}$ by solving the linear system

$$\mathbf{H}\Delta\mathbf{d} = -\mathbf{f} \tag{6.3}$$

using a direct solver for sparse positive definite systems [Schenk et al., 2001].

The pseudocode function *ComputeE* defined in Algorithm 6.1 computes the energy E for given DOF values $\mathbf{d}$. It calls *ComputePsi* to evaluate the energy density Ψ for the given deformation gradient $\mathbf{F}$. *ComputeBFunDeriv* computes the derivative of a basis function at a given position. E_C computes the energy of constraint C for the given DOF values.

```
1  ComputeE( d = (x^1T, ..., x^nT)^T )
2    E ← 0
3    for each element K^k:
4      for each quadrature point q_j with weight w_j of K^k:
5        F ← 0
6        for each basis function N_i of K^k:
7          b ← ComputeBFunDeriv(i, q_j)
8          F += x_i b^T
9        end
10       E += ComputePsi(F) w_j
11     end
12   end
13   for each soft constraint C:
14     E += E_C(d)
15   end
16   return E
17 end
```

Algorithm 6.1: Computation of the energy E.

6.3 Application Specifics

We have now defined image warping in the continuous sense and presented the generic framework of non-linear FEM. At this point we still have to make a choice about the basis functions N_i, the mesh connectivity $K^1, \ldots, K^m$, and the energy density function Ψ. Now we discuss how we may want to pick these for different image warping applications.

6.3.1 Deformation Energy Densities

One of the main application-specific questions is what kind of energy density functions Ψ we want. We look at several different possible formulations, showing how our continuous formulation allows us to not only reproduce existing warping energies used in earlier work, but also more clearly understand their limitations and design new constraints. We start by defining some important quantities from continuum mechanics [Bonet and Wood, 1997].

Most of the commonly used deformation energy densities for image warping can be computed from a combination of quantities derived from the

deformation gradient $\mathbf{F}(\mathbf{X})$. $\mathbf{F}$ itself can also be useful here, as it tells us how an infinitesimal line segment $d\mathbf{X}$ at position $\mathbf{X}$ gets deformed under $\boldsymbol{\varphi}$. The deformed line segment $d\mathbf{x}$ can be computed as $d\mathbf{x} = \mathbf{F}d\mathbf{X}$. Another useful quantity is the Jacobian determinant J defined as:

$$J(\mathbf{X}) = \det(\mathbf{F}(\mathbf{X})) \tag{6.4}$$

The Jacobian determinant tells us how an infinitesimal area changes under the deformation $\boldsymbol{\varphi}$. We will make frequent use of the right Cauchy-Green tensor $\mathbf{C}$ defined as

$$\mathbf{C}(\mathbf{X}) = \mathbf{F}(\mathbf{X})^T\mathbf{F}(\mathbf{X}), \tag{6.5}$$

which is invariant under rotation and thus any energy density that can be expressed in terms of $\mathbf{C}$ will inherit its rotational invariance. Analyzing the eigenvalues λ_1, λ_2 of $\mathbf{C}(\mathbf{X})$, one can see that $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$ are the *principal stretches* of the deformation at $\mathbf{X}$, i.e. the minimum and maximum values for the stretch that can be achieved for any direction $d\mathbf{X}$ at position $\mathbf{X}$ [Sander et al., 2001].

Translational Invariance. As shown in Shamir and Sorkine [2009], one simple energy density for image warping is the one that penalizes all deformations except for translations

$$\Psi_F = \|\mathbf{F} - \mathbf{I}\|_F^2, \tag{6.6}$$

where $\mathbf{I}$ denotes the 2×2 identity matrix and $\|\cdot\|_F$ denotes the Frobenius norm of a matrix (see Appendix B). Ψ_F is only quadratic in derivatives of $\boldsymbol{\varphi}$, so when using this energy density, the minimum of a discretized deformation energy E can by found in a single Newton step.

Scaling Invariance. For energies that result in more complex optimization problems, existing methods usually employ alternating iterative methods [Shamir and Sorkine, 2009] that solve a linear least-squares problem while fixing the values of some quantities, like a uniform scale. On the other hand, once we have found the corresponding continuous formulation of such energies, our FEM framework can then solve the resulting non-linear problem in a consistent way.

For example, Wang et al. [2008] and Laffont et al. [2010] penalize all transformations other than translation and uniform scaling. The corresponding continuous energy density can be written as:

$$\Psi_{\bar{F}} = ||\mathbf{F} - J^{\frac{1}{2}}\mathbf{I}||^2 \tag{6.7}$$

Rotation Invariance. The right Cauchy-Green tensor $\mathbf{C}$ is equal to the identity matrix for pure rotations, a fact we can use to find an energy density penalizing only translations and rotations [Wang et al., 2010]:

$$\Psi_C = ||\mathbf{C} - \mathbf{I}||^2 \tag{6.8}$$

Similarity Invariance. The distortion energy used in Zhang et al. [2009] also allows for scaling invariance, permitting elements to undergo a similarity transform (quadratic in derivatives of $\boldsymbol{\varphi}$). Its corresponding energy density is:

$$\Psi_S = \mathrm{tr}(\mathbf{C}) - 2J \tag{6.9}$$

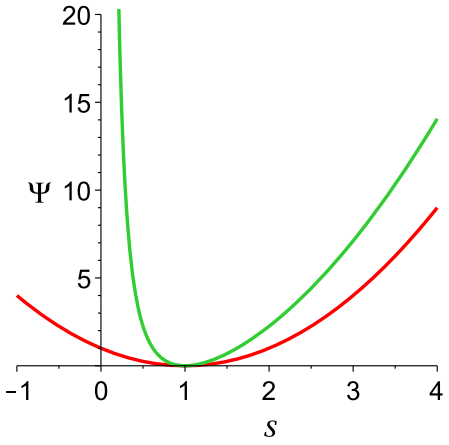

Figure 6.3: *Comparison between the energy densities $\Psi_{\bar{C}} - 2$ (green) and Ψ_S (red) for non-uniform scaling with $\mathbf{F} = \left(\begin{smallmatrix} s & 0 \\ 0 & 1 \end{smallmatrix}\right)$.*

While this allows for an efficient minimization of E in a single Newton step, this energy density has the drawback of tolerating inversions. As can be seen in Figure 6.3, Ψ_S assumes a finite value when the stretch s in direction X_1 becomes negative. Our continuous formulation allows us to design an alternative energy density $\Psi_{\bar{C}}$ that possesses the same invariance to scaling and rotation but increases to infinity as s approaches zero, thus preventing inversions:

$$\Psi_{\bar{C}} = J^{-2}||\mathbf{C}||^2 \tag{6.10}$$

For practical considerations, it helps to leave this energy density undefined for $J \leq 0$. This will prevent the Newton solver from ever taking a step that would invert an element, and instead results in the line search finding an increment $\Delta\mathbf{d}$ such that J remains positive at every quadrature point.

Figure 6.4: *Comparison between different deformation energy density functions in an image retargeting application, with indicated invariance to translation (T), scaling (S), and rotation (R). Note the self-intersection in the second right image, which is eliminated by our proposed energy density $\Psi_{\bar{C}}$.*

One can show that $||\mathbf{C}||^2 = \lambda_1^2 + \lambda_2^2$ and $J^2 = \lambda_1\lambda_2$, from which we derive an alternative representation of $\Psi_{\bar{C}}$,

$$\Psi_{\bar{C}} = \frac{\lambda_1^2 + \lambda_2^2}{\lambda_1\lambda_2} = \frac{(\lambda_1 - \lambda_2)^2}{\lambda_1\lambda_2} + \kappa, \tag{6.11}$$

where the constant $\kappa = 2$ does not depend on the warp and can thus be dropped from the energy density function. The last equations show that $\Psi_{\bar{C}}$ essentially measures the difference between the squares of the minimum and maximum stretches $\sqrt{\lambda_1}, \sqrt{\lambda_2}$ of the deformation, divided by J to achieve invariance to uniform scaling. It thus measures the 'non-uniformity' of the scaling, while being invariant to rotations and uniform scaling.

Line Bending. One common energy term that comes up in image warping methods prevents bending of mesh edges [Wang et al., 2008]. In the undeformed state, these edges are exactly horizontal and vertical, meaning that this constraint is anisotropic and only works to preserve horizontal and vertical lines in an image. A straightforward application of the deformation gradient to the line segments $(\mathrm{d}S, 0)^T$ and $(0, \mathrm{d}S)^T$ leads to the corresponding continuous energy density:

$$\Psi = C_{11} - F_{11}\sqrt{C_{11}} + C_{22} - F_{22}\sqrt{C_{22}} \tag{6.12}$$

However, the same effect can be achieved by using the simpler

$$\Psi_L = F_{12}^2 + F_{21}^2, \tag{6.13}$$

which effectively penalizes shearing deformations along the main axes. This is identical to the 'edge preservation' constraint used in Krähenbühl et al. [2009] and Wolf et al. [2007].

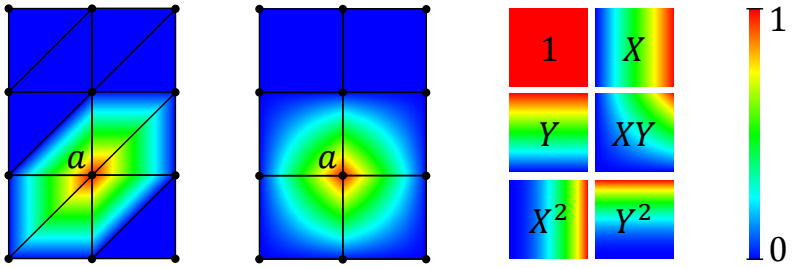

Figure 6.5: *Values of nodal basis function N_i in a triangle mesh (left) and a quad mesh (middle), and the six basis functions used for quadratic DG FEM elements (right).*

Visual Saliency. A key component of many content-aware image warping methods is the inclusion of a visual saliency term. The higher the saliency of a region, the better should it be 'preserved' in the warp. This can be easily included in our formulation by multiplying the energy density by some spatially varying saliency function $\gamma(\mathbf{X}) > 0$. For the results shown here, γ was computed using a recent state-of-the art method from Goferman et al. [2010].

We have presented a small selection of possible energy density terms in order to demonstrate how such terms can be created, and what their effects are on image warping applications. By representing energy densities in these general terms, not only can we replicate existing solutions, but it also allows us to discover new insights and provide improvements, such as $\Psi_{\bar{C}}$, an energy that allows similarity transforms while explicitly preventing inversions. Figure 6.4 shows a collection of these energy densities used in an image retargeting application.

6.3.2 Basis Functions

In standard FEM, nodal DOFs are used, meaning that $q = n$ and every DOF is associated with a node: the DOF $\mathbf{x}_i$ directly represents the solution of the warp at node i. This implies that for all i, the basis function $N_i(\mathbf{X})$ assumes a value of 1 at position $\mathbf{X}_i$, and a value of 0 at all $\mathbf{X}_j$ with $i \neq j$. The natural choice for basis functions for simple triangle and quad elements are the linear and bi-linear basis functions, respectively (see Figure 6.5). These basis functions are used in most prior work, and restrict the methods to averaging content (such as image saliency) within elements. With the FEM, saliency information is sampled at multiple quadrature points during the numerical integration allowing more accurate deformation penalties.

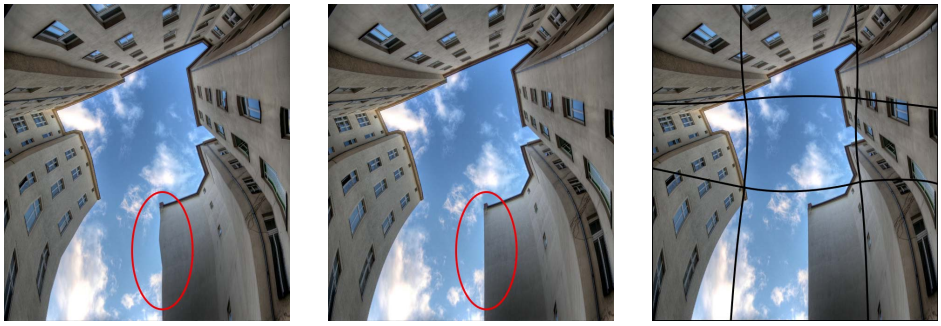

Figure 6.6: *Retargeted image using 81 bilinear quads (left, 100 DOFs) vs. 9 cubic quads (middle, 90 DOFs) and the corresponding deformed mesh (right). Using higher order basis functions yields smoother results.*

DG FEM. In discontinuous Galerkin FEM, each element is endowed with its own set of basis functions. Using nodal basis functions is still possible in this context, but in contrast to standard FEM, each element using the corresponding node will have its own DOF associated with that node. The big advantage of the method is that we are no longer restricted to nodal basis function but we can use simple polynomials of arbitrary order, where the order of polynomials can be chosen independently for each element. For example, the basis functions $1, X_1, X_2, X_1^2, X_1X_2, X_2^2$ allow an element to approximate the solution quadratically (see Figure 6.5).

To restore the coupling between elements, some additional terms are necessary. There are several possible ways to achieve this using DG FEM, but it all comes down to how the bi-valued function $\boldsymbol{\varphi}$ is replaced by the so-called *numerical flux* $\hat{\boldsymbol{\varphi}}$ on the edges between elements. We proceed exactly as described in Section 3.2, using the numerical flux of Bassi and Rebay [1997].

For rendering elements with higher-order basis functions, we triangulate each element using a fine triangulation and compute the new positions of the triangle mesh vertices by evaluating $\boldsymbol{\varphi}$ inside the element. A penalty term is used to reduce the size of gaps between elements. These gaps are then small enough (sub-pixel size) to not cause artifacts in the warp and for rendering we additionally average the values of $\boldsymbol{\varphi}$ on edges between elements.

Figure 6.6 shows where higher order basis functions are beneficial. Linear elements can result in visible 'kinks' between elements because they cannot represent bending, while higher order basis functions result in smoother transitions between elements.

Figure 6.7: *Error-based adaptivity for DG FEM with linear, quadratic and cubic quad elements (approx. 1024 DOFs each)*

6.3.3 Mesh Construction

We take advantage of well behaved convergence properties of the FEM (see Figure 6.8), and our mesh-independent formulation of the continuous image warping problem to propose an adaptive mesh. We can refine the mesh in important areas to achieve a high quality solution while drastically reducing the number of DOFs, allowing for temporally stable solutions with a tractable problem size.

While we can use any mesh subdivision technique, we demonstrate a couple of simple choices. One method that we use is computed from a Delaunay triangulation [Shewchuk, 1996] of a point set distributed according to *variance* in saliency, which allows for increased resolution in areas that are most likely to contain changes in local deformation.

When using DG elements, we gain some additional freedom for mesh construction. In particular, the edges of neighboring elements are not required to coincide, creating hanging nodes. This allows for the use of adaptive quadtree meshes. Additionally, DG FEM gives rise to an error-based refinement scheme. As the amount of discontinuities between elements is a direct indicator of the local error of the solution, refining the elements with the highest discontinuities will reduce the global error in a greedy fashion and also reduce the amount of visible gaps, resulting in a mesh where the function values on edges can safely be averaged for rendering. See Figure 6.7 for an example.

6.3.4 Additional Constraints

Boundary Conditions. Depending on the particular application, we may want to constrain all nodes on the boundary of Ω such that $(x_1^a, x_2^a)^T = (X_1^a s_1, X_2^a s_2)^T$ if the image is stretched by $(s_1, s_2)^T$, or allow

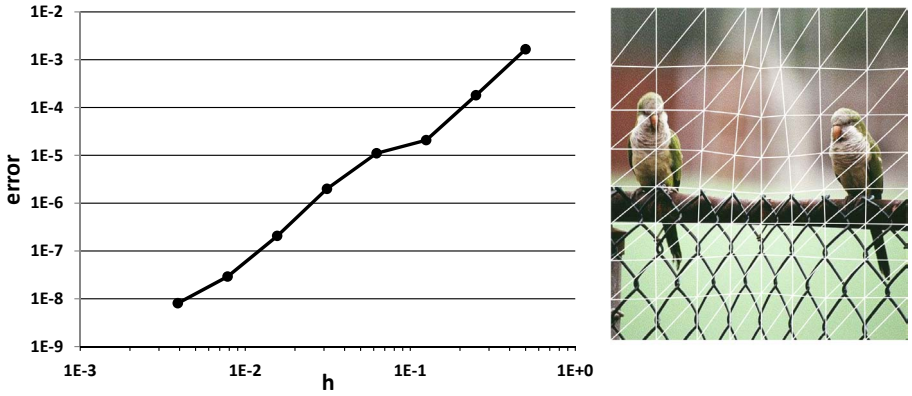

Figure 6.8: *Error $\int_\Omega ||\boldsymbol{\varphi} - \bar{\boldsymbol{\varphi}}||^2$ with respect to a reference solution $\bar{\boldsymbol{\varphi}}$ for meshes with different element size h, showing nice convergence properties. The image on the right shows the solution for $h = 1/8$. $\Psi_{\bar{F}}$ was used as the energy density.*

nodes to slide on the boundary, by constraining $x_1^a = X_1^a s_1$ for nodes with $X_1^a = 0$ or $X_1^a = 1$ and similarly for the X_2 direction. These constraints can be implemented as hard constraints by modifying the first and second derivatives of E such that the constrained DOFs do not get modified during a Newton step.

Weak Point Constraints. For certain applications like camera stabilization [Liu et al., 2009] and stereoscopic disparity editing [Lang et al., 2010], we want a weakly enforced point constraint such that $\mathbf{P}_W$ gets warped to a specific position $\mathbf{p}_W$. Such constraints can be realized by adding an energy term for each point constraint:

$$E_W = \gamma_w \frac{1}{2} ||\boldsymbol{\varphi}(\mathbf{P}_W) - \mathbf{p}_W||^2 \tag{6.14}$$

Line Constraints. We can also impose line constraints on the warp to weakly enforce straight lines to remain straight after the warp [Krähenbühl et al., 2009], or to warp initially curved lines to straight ones [Carroll et al., 2009]. Similar to previous approaches, we parameterize the best-fitting straight line as $\sin(\alpha)x_1 + \cos(\alpha)x_2 + b = 0$. For each line constraint, an additional energy term is added to our minimization problem, computed as the integral of the squared distance between the warped curve and the closest point on the fitted straight line, weighted by a penalty γ_L:

$$E_L = \frac{1}{2}\gamma_L \int_0^l (\sin(\alpha)p_1(s) + \cos(\alpha)p_2(s) + b)^2 ds \tag{6.15}$$

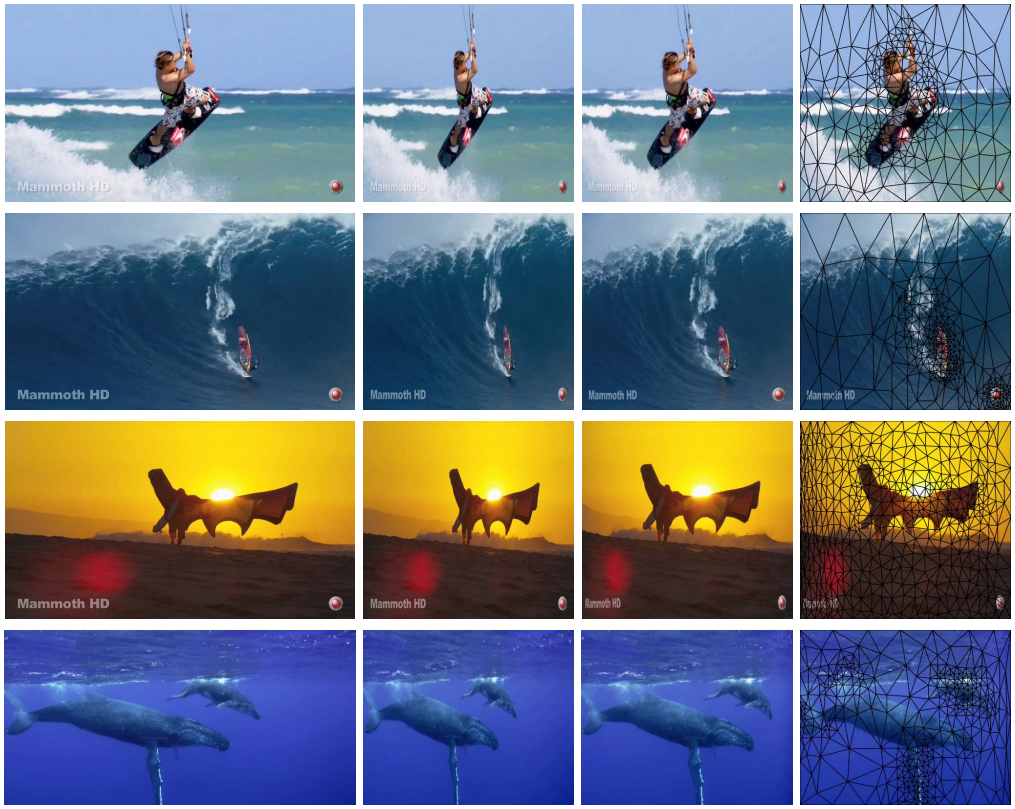

Figure 6.9: *Frames from retargeted video sequences. Showing the input frame, linear scaling, FEM warping, and the corresponding mesh.*

We use per-element quadrature to evaluate this integral. These kind of constraints can be incorporated seamlessly into our method: the two new unknowns a and b parameterizing the fitted straight line simply become two new scalar DOFs of our non-linear problem, and no further special treatment is necessary.

Temporal Stability. When dealing with video sequences, consistency of output over time is important. Working with per-frame warps, we can formulate this as weak constraints that force an object to deform in a similar way in each frame of the video. Given a position $\mathbf{X}^t$ in frame t and a corresponding position $\mathbf{X}^{t+1}$ in frame $t+1$, our constraint tries to minimize the difference between the deformation gradients at these points, taking into account object and camera rotation and scaling represented as the matrix $\mathbf{G}_t$. The corresponding energy is:

$$E_T = \gamma_T \frac{1}{2} ||\mathbf{G}_t \mathbf{F}_t(\mathbf{X}^t)\mathbf{G}_t^{-1} - \mathbf{F}_{t+1}(\mathbf{X}^{t+1})||^2 \tag{6.16}$$

a b c

Figure 6.10: *Comparison of temporal stability in video retargeting on a synthetic dataset. Cropped results showing: a) original image, b) two frames from Krähenbühl et al. [2009] (note the difference in shape of the object), and c) same two frames from our method.*

This constraint couples the otherwise independent DOFs of frames t and $t+1$. One possibility of retrieving the correspondence between $\mathbf{X}^t$ and $\mathbf{X}^{t+1}$ and the transformation $\mathbf{G}_t$ is by using SIFT feature tracking.

6.4 Results

While the presented framework itself is more general and could be applied to any warping problem, this section shows results for two image warping applications.

Video Retargeting. For different video retargeting applications, we can choose various energy density functions. Simple ones like Ψ_F, Ψ_S are attractive for performance reasons, because the solution can be found in a single Newton step. For difficult cases and extreme aspect ratio changes, one that prevents inversions such as $\Psi_{\bar{C}}$ is preferable. To solve over a whole sequence of frames and still keep the problem at a tractable size, we use an adaptive triangle mesh and standard FEM. See Figure 6.9 for examples.

We compare the temporal stability of our method to prior work by Krähenbühl et al. [2009] in Figure 6.10.

Video Stabilization. Warping-based video stabilization methods [Liu et al., 2009] consist of tracking feature points over time, reconstructing their positions and the camera in 3D space, then reprojecting feature points into a new, stabilized camera path. The input to the image warp consists of a set of weighted feature points with source and (reprojected) target positions, which guide the warp as weak point constraints. Additionally, we select an energy

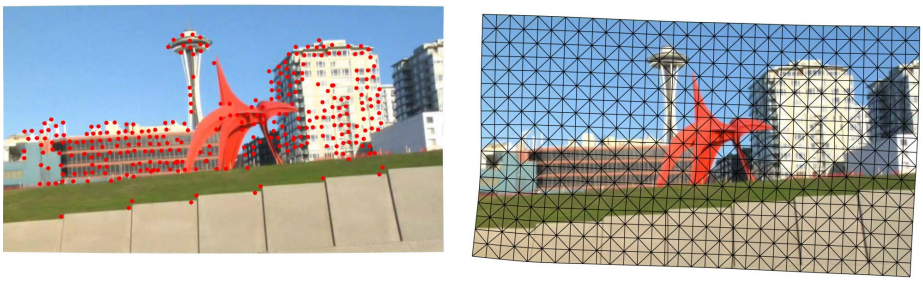

Figure 6.11: *Our image warping method applied to camera stabilization. Input frame with detected features (left), stabilized frame with FEM mesh (right).*

Figure 6.12: *Line constraints controlling the appearance of objects in a modified perspective. Input image with triangle mesh and constrained lines shown in red (left), warp without line constraints (middle), warp with line constraints (right).*

density function that is invariant to rotation. Good choices are Ψ_C or $\Psi_{\bar{C}}$, but Ψ_S works well, too, because the amount of deformation is typically low for this particular application. There is no direct influence between the warps computed for individual frames, so each frame can be warped independently and a regular mesh of size 64×36 as used in Liu et al. [2009] provides a good quality-vs-performance trade-off. See Figure 6.11 for examples.

Other Applications. There are other methods that while we have not fully reimplemented them, follow from our earlier definitions. For example, Figure 6.12 shows an example of user line constraints, which are used as a means of creating artistic perspective corrections [Carroll et al., 2010]. In addition, by using our weak point constraints on stereo correspondences, we can perform stereoscopic disparity editing [Lang et al., 2010].

Timing. We show the timing of our method on several examples in Table 6.1. These results were generated on an Intel Core i7 3.2 GHz, with a single-threaded application.

Example	Ψ	#Els	#Frames	#Newton	t_{pre}	t_{Hf}	t_{solve}
Fig. 6.1, right	$\Psi_{\bar{C}}$	2612	1	7	60	2	19
Fig. 6.9, wakeboarder	Ψ_F	111124	234	1	27820	600	6311
Fig. 6.9, wind surfer	Ψ_F	54249	117	1	14634	273	2537
Fig. 6.11	$\Psi_{\bar{C}}$	1152	1	2	31	2	4

Table 6.1: *Problem complexity and timings (in ms) for precomputation (including mesh generation), computation of* $\mathbf{H}$, $\mathbf{f}$, *and solving the linear system*

Limitations. One limitation to our FEM approach is that it can be more complex to implement than traditional methods, especially when using DG FEM. However, this is a one-time cost, and when completed, the framework is very flexible, making application to novel problems and domains a much simpler task than before.

Our method supports high order basis functions. However, there is a trade-off with using these. While fewer elements are needed, there are more degrees of freedom per element. In addition, it also results in denser systems to solve. However, the decisions to what kind of basis function should be used can be made by the specific applications, weighting advantages and disadvantages of either.

In addition, with non-linear image warping problems, it is difficult to prove convergence guarantees, and it is possible with extreme warps the optimization method may result in a local minimum that does not correspond to our desired solution. However, this is true with existing methods as well, and we did not observe this behavior in any of our examples. Furthermore, our representation allows for easy substitution of energy densities, mesh formulation, and minimization methods, potentially making dealing with local minima a simpler problem to resolve.

6.5 Discussion and Outlook

In conclusion, we have presented a novel, general representation for image warping that unifies a wide range of existing solutions. Our approach provides a well defined continuous mathematical formulation that has multiple

real-world advantages. For one, a mathematical basis allows for energy densities to be clearly defined and analyzed, allowing for improved understanding and design. Our representation is independent of the mesh formulation, which allows for simple extensions for adaptive meshing and temporally stable solutions, and naturally permits higher order basis functions, which allow smoother warps using a small number of elements. However, probably most significantly, there is a wide range of literature and ongoing research about FEM techniques in the mechanics and geometry communities, and by phrasing the image warping problem in the same context, both areas of research have the potential to benefit tremendously from their combined research efforts.

At this point, we have only scratched the surface of what the FEM could be used for in the context of image editing applications. Further insight into FEM and related methods could provide additional capabilities, such as using XFEM for discontinuous warping methods. Our representation is also not restricted to 2D elements, and one extension could be a mesh subdivision with 3D elements, such as a video cube oct-tree, that may provide stability for temporal solutions. In addition, it is possible that energies with higher order derivatives could be useful, e.g. for the spherical distortion application.

CHAPTER 7

Conclusion

This chapter discusses the main contributions of this thesis and looks at potential future work.

7.1 Discussion

While focusing on the simulation of deformable solids and shells as well as image warping, this thesis has shown that DG FEM approaches can compete with, or even outperform, traditional finite element methods in certain computer graphics applications. Overall, the benefits of DG FEM did not so much turn out to be of a quantitative nature (resulting in significantly better performance or faster convergence). Instead, the application of DG FEM lead to elegant solutions to existing problems, with one method solving several issues at once, and in some cases it turned out to allow for results that simply were not possible using existing methods. As a by-product, some of the contributions made by this thesis go beyond, or are even orthogonal to DG FEM and may find broader applications in other areas of computer graphics. This includes for example the exact integration of polynomials over polyhedra, the concept of enrichment textures, or the analysis of image warping energy terms using tools from material science.

Chapter 4 presented a simulation technique for deformable solids based on DG FEM. The discontinuous shape functions of DG FEM allowed the simulation of arbitrary polyhedral elements using simple polynomial basis functions that can even be integrated exactly (up to machine precision) within

elements. A corotational adaption of DG FEM allowed for large rotational deformations while still using a fast linear deformation model for elasticity. Smooth deformation fields computed using MLS allow for the embedding of high resolution surface geometry, even in the presence of element separation and overlaps due to the discontinuous nature of DG FEM. As a consequence, mesh modifications become easier to handle in the proposed DG FEM framework, which was demonstrated through the simulation of progressive cutting, adaptive element refinement, and a novel mesh generation approach that can represent complex surface geometries with only a few elements.

In Chapter 5, a corotational extension of a linear DG FEM shell model was presented, which allows for the simulation of nonlinear phenomena such as buckling while still being efficient to compute. Making use of the fact that DG FEM can deal with C^0 continuous basis functions for shell models, a novel basis function enrichment method has been presented that allows for the simulation of very detailed cut and fracture curves even on coarse simulation meshes. These discontinuities are represented in what we termed *enrichment textures*, discretely sampled functions that modify the existing set of basis functions. In this framework, incomplete cuts, progressive cuts, the interplay of multiple cuts, and cuts extending over several elements can be handled in a uniform way by solving a Laplace problem with suitable boundary conditions. To visualize the results of such simulations in real time, a novel rendering method based on a geometry shader was presented.

Applying DG FEM to image warping, Chapter 6 presented a unified framework that encompasses a wide range of existing warping approaches. Thanks to the underlying continuous mathematical formulation, existing energy densities can be analyzed and improved, and new energies with desirable properties designed from scratch. The main benefit of FEM over alternative approaches is that it supports irregular domains and flexible meshing. This feature allows for adaptively sampling the warp to concentrate DOFs where they are most needed, which in turn reduces the total number of DOFs without significantly reducing the accuracy of the computed solution. Thanks to this, large problems where a solve is computed over several images, such as temporally stable video retargeting, now become tractable. The use of DG FEM allowed for even more aggressive adaptivity by choosing higher polynomial degrees in elements where the solution is expected to show a more non-linear behavior. Also, DG FEM simplifies the process of generating an adaptive mesh. While a simple quad-tree would not be a valid mesh for CG FEM due to hanging nodes, DG FEM can handle such meshes without problems. This is computationally easier and less error prone than computing a “nice” adaptive triangulation of the image domain.

7.2 Future Work

While the focus of this work was on applying discontinuous Galerkin methods to elliptic problems, there are certainly other problem types with relevant applications in graphics, fluid simulations being the prime example. In fact, these were amongst the first problems to which DG FEM had been applied [Bassi and Rebay, 1997]. Fluid simulation applications could potentially benefit from the adaptive meshing, or dynamic meshing to handle changing boundary conditions in the simulation of fluid-solid interaction.

Another exciting aspect of DG FEM that has not been considered in this thesis is space-time adaptivity [Altmann et al., 2010]. Just as a spatial element mesh is allowed to have hanging nodes in DG FEM, a spatio-temporal mesh can combine elements that have shorter or longer lifespans, and the interaction between these elements is automatically handled by the numerical fluxes. In computer graphics, this concept could again be applied to the simulation of deformable objects or fluids, resulting in dynamic simulations that automatically, and locally, adapt the size of their time steps to reduce overall simulation times.

Deformable Solids. One obvious extension to the proposed method for the simulation of deformable solids is its application to nonlinear material models. The same benefits as for linear models can be expected here, and additionally, DG FEM may offer ways to handle degenerate cases more gracefully. Elements under stress that would otherwise get inverted could to some degree be compensated by the discontinuities between elements.

Collision handling has been limited to simple point-plane collisions, but full collision handling including self-collisions would be required for many practical applications. For linear elements, existing techniques for collision detection and handling on triangle meshes could be used. Higher order elements will bend under deformation and thus need special handling for collision detection on polynomial surfaces, which could be an interesting topic for future research.

One further advantage of DG FEM that has not been exploited in this work is that it can quite easily be parallelized [Devine et al., 1993; Noels and Radovitzky, 2006]. The simulation mesh can be partitioned in advance, and computations on each part can be performed in parallel. The only way these parts exchange information is through the numerical fluxes on the element boundaries. It could be interesting to investigate how well this idea would extend to the simulation of deformable objects, especially as today's multi

core machines and highly parallel GPUs are indicators for how computing resources will have to be used in the future.

Enrichment Textures for Shells. The representation of material discontinuities as texture maps can be seen as a first example that may motivate investigation in other areas where physical simulations can be enhanced by the use of texture concepts. Some preliminary work in this direction has already been shown: thickness textures can define shells with locally varying thicknesses. Borrowing from existing uses of textures, e.g. displacement maps or bump maps used in rendering, one could also think of displacing the undeformed surface of a shell according to a texture map. This approach would allow for coarse shell elements with highly detailed surfaces.

As shown in Section 5.6, enrichment textures can also be used to represent creases. The problem of how to compute physically motivated enrichment functions for creases, especially in the presence of other discontinuities, remains an open problem. Solving the Laplace problem is still a bottleneck when simulating progressive cuts in our implementation. However, this computation as well as the rest of the simulation pipeline can be expected to be parallelizable on the GPU without too much effort, which would open up the door to real-time applications such as games.

Image Warping. By formulating the warping problem in an FEM context, the doors are wide open to take advantage of the vast body of FEM research and to apply the latest research results to problems in image warping. Our work has really only scratched the surface of what is possible. One interesting approach could be to apply the XFEM and in particular enrichment textures to image warping, to allow for discontinuous warps with details at the pixel level, while still simulating only coarse elements. Additionally to what has been shown in this work, there may be other concepts from the mechanics and geometry communities that could be applied to image warping. After all, there is a close relationship between how real physical materials deform and how one expects a warped image to behave.

In this work, the warp energies and constraints have been limited to only involve first spatial derivatives. For some applications such as spherical distortion correction, the involved energies use higher order derivatives and it would be interesting to extend our method to support those as well. Finally, starting from the temporally stable video retargeting approach with adaptive per-frame meshes, one obvious direction for future work is to look at the

video cube as a three-dimensional domain and use 3D elements in an octtree fashion to approximate the solution. Elements would thus also have a temporal dimension, and the mesh would be adaptive in space as well as time, further reducing the total number of degrees of freedom of the problem.

APPENDIX A

Derivations and Proofs

This appendix shows the details of derivations and proofs mentioned in earlier chapters.

A.1 Derivation of IP DG Weak Form for Linear Elasticity

This section shows the detailed derivation of the DG weak form for the interior penalty method (see Equation (4.3)).

Notation. The average operator $\{\cdot\}$ has the same meaning for scalars and tensors as in Chapter 4. Additionally to the $[\![\cdot]\!]$ jump operator mapping vectors to matrices and vice versa, the jump operator $[\![\cdot]\!]_s$ maps vectors to scalars and vice versa as follows:

$$
\begin{aligned}
[\![\mathbf{u}]\!]_s &:= \mathbf{u}^- \cdot \mathbf{n}^- + \mathbf{u}^+ \cdot \mathbf{n}^+, \\
[\![s]\!]_s &:= s^- \mathbf{n}^- + s^+ \mathbf{n}^+.
\end{aligned}
$$

Identities. We will be making use of the following identities, where $\mathbf{v}$ and $\mathbf{n}$ represent vectors and σ represents a matrix.

$$
[\![\mathbf{v} \cdot \sigma]\!]_s = [\![\mathbf{v}]\!] : \{\sigma\} + \{\mathbf{v}\} \cdot [\![\sigma]\!], \tag{A.1}
$$

$$
[\![\{\cdot\}]\!] = [\![[\![\cdot]\!]]\!] = 0, \tag{A.2}
$$

$$[\![\{\cdot\}]\!]_s = [\![[\![\cdot]\!]]\!]_s = 0, \tag{A.3}$$

$$\{[\![\cdot]\!]\} = [\![\cdot]\!], \tag{A.4}$$

$$\{\{\cdot\}\} = \{\cdot\}, \tag{A.5}$$

$$\mathbf{v} \cdot \boldsymbol{\sigma} \cdot \mathbf{n} = (\mathbf{v} \otimes \mathbf{n}) : \boldsymbol{\sigma}. \tag{A.6}$$

A.1.1 Strong Form

Recall the strong form of the linear elasticity PDE, formulated as two first order PDEs:

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}), \tag{A.7}$$

$$\nabla \cdot \boldsymbol{\sigma} = -\mathbf{f}. \tag{A.8}$$

A.1.2 Local Weak Form

The domain Ω gets triangulated into elements K. The space of the functions $\mathbf{u}$ and $\boldsymbol{\sigma}$ is restricted to polynomials on each element ($P(K)$ and $\Sigma(K)$, respectively). Multiplying Equations (A.7) and (A.8) by a tensor test function $\boldsymbol{\tau}$ and and a vector test function $\mathbf{v}$, respectively, and integrating over element K, gives

$$\int_K \boldsymbol{\tau} : \boldsymbol{\sigma} = \int_K \boldsymbol{\tau} : C : \boldsymbol{\epsilon}(\mathbf{u}), \quad \forall \boldsymbol{\tau} \in \Sigma(K), \tag{A.9}$$

$$\int_K \nabla \cdot \boldsymbol{\sigma} \cdot \mathbf{v} = -\int_K \mathbf{f} \cdot \mathbf{v}, \quad \forall \mathbf{v} \in P(K). \tag{A.10}$$

Noting the identity

$$\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v}) = -\int_\Omega \nabla \cdot \boldsymbol{\sigma} \cdot \mathbf{v} + \int_{\partial\Omega} \mathbf{v} \cdot \boldsymbol{\sigma} \cdot \mathbf{n} \tag{A.11}$$

and applying this to the right-hand side of (A.9) and the left-hand side of (A.10) results in the local weak form of element K:

$$\int_K \boldsymbol{\tau} : \boldsymbol{\sigma} = -\int_K \nabla \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{u} + \int_{\partial K} \mathbf{u} \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{n}, \quad \forall \boldsymbol{\tau} \in \Sigma(K), \tag{A.12}$$

$$\int_K \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v}) = \int_K \mathbf{f} \cdot \mathbf{v} + \int_{\partial K} \mathbf{v} \cdot \boldsymbol{\sigma} \cdot \mathbf{n}, \quad \forall \mathbf{v} \in P(K). \tag{A.13}$$

Fluxes. Introducing fluxes $\hat{\mathbf{u}}$ and $\hat{\sigma}$ in (A.12) and (A.13) to define the bi-valued functions $\mathbf{u}$ and σ on ∂K gives

$$\int_K \boldsymbol{\tau} : \sigma = -\int_K \nabla \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{u} + \int_{\partial K} \hat{\mathbf{u}} \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{n}, \quad \forall \boldsymbol{\tau} \in \Sigma(K), \tag{A.14}$$

$$\int_K \sigma : \boldsymbol{\epsilon}(\mathbf{v}) = \int_K \mathbf{f} \cdot \mathbf{v} + \int_{\partial K} \mathbf{v} \cdot \hat{\sigma} \cdot \mathbf{n}, \quad \forall \mathbf{v} \in P(K). \tag{A.15}$$

A.1.3 Global Weak Form

Generalizing the flux definition from Arnold et al. [2000] to tensors, we get the following fluxes for the IP method:

$$\hat{\mathbf{u}} = \{\mathbf{u}\},$$
$$\hat{\sigma} = \{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!].$$

Sum Over Elements (First Part of Weak Form). The following equations are to hold $\forall \boldsymbol{\tau}$ and $\forall \mathbf{v}$. Summing (A.14) over all elements K and inserting the previously defined IP flux we get

$$\sum_K \int_K \boldsymbol{\tau} : \sigma = -\sum_K \int_K \nabla \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{u} + \sum_{e \in \varepsilon_h} \int_e [\![\{\mathbf{u}\} \cdot (\boldsymbol{\tau} : \mathbf{C})]\!]_s.$$

Note that ε_h denotes the set of all element edges, including $\partial\Omega$ and all edges between elements. Using (A.11) this becomes

$$\begin{aligned}\sum_K \int_K \boldsymbol{\tau} : \sigma &= \sum_K \int_K (\boldsymbol{\tau} : \mathbf{C}) : \boldsymbol{\epsilon}(\mathbf{u}) - \sum_K \int_{\partial K} \mathbf{u} \cdot (\boldsymbol{\tau} : \mathbf{C}) \cdot \mathbf{n} \\ &\quad + \sum_{e \in \varepsilon_h} \int_e [\![\{\mathbf{u}\} \cdot (\boldsymbol{\tau} : \mathbf{C})]\!]_s \\ &= \sum_K \int_K (\boldsymbol{\tau} : \mathbf{C}) : \boldsymbol{\epsilon}(\mathbf{u}) + \sum_{e \in \varepsilon_h} \int_e [\![(\{\mathbf{u}\} - \mathbf{u}) \cdot (\boldsymbol{\tau} : \mathbf{C})]\!]_s.\end{aligned}$$

Using (A.1) we get

$$\begin{aligned}\sum_K \int_K \boldsymbol{\tau} : \sigma &= \sum_K \int_K (\boldsymbol{\tau} : \mathbf{C}) : \boldsymbol{\epsilon}(\mathbf{u}) \\ &\quad + \sum_{e \in \varepsilon_h} \int_e [\![\{\mathbf{u}\} - \mathbf{u}]\!] : \{\boldsymbol{\tau} : \mathbf{C}\} + \{\{\mathbf{u}\} - \mathbf{u}\} \cdot [\![\boldsymbol{\tau} : \mathbf{C}]\!].\end{aligned}$$

Because of (A.5) it holds that $\{\{\mathbf{u}\} - \mathbf{u}\} = 0$ and the above can be simplified to

$$\sum_K \int_K \boldsymbol{\tau} : \sigma = \sum_K \int_K (\boldsymbol{\tau} : \mathbf{C}) : \boldsymbol{\epsilon}(\mathbf{u}) + \sum_{e \in \varepsilon_h} \int_e [\![\{\mathbf{u}\} - \mathbf{u}]\!] : \{\boldsymbol{\tau} : \mathbf{C}\}.$$

Using (A.3) we finally get

$$\sum_K \int_K \boldsymbol{\tau} : \boldsymbol{\sigma} = \sum_K \int_K (\boldsymbol{\tau} : \mathbf{C}) : \boldsymbol{\epsilon}(\mathbf{u}) - \sum_{e \in \varepsilon_h} \int_e [\![\mathbf{u}]\!] : \{\boldsymbol{\tau} : \mathbf{C}\}. \tag{A.16}$$

Sum Over Elements (Second Part of Weak Form). Summing (A.15) over all elements K and inserting the previously defined flux we get

$$\begin{aligned}\sum_K \int_K \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v}) &= \int_\Omega \mathbf{f} \cdot \mathbf{v} + \sum_K \int_{\partial K} \mathbf{v} \cdot (\{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!]) \cdot \mathbf{n} \\ &= \int_\Omega \mathbf{f} \cdot \mathbf{v} + \sum_{e \in \varepsilon_h} \int_e [\![\mathbf{v} \cdot (\{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!]]\!]_s.\end{aligned}$$

Using (A.1) we get

$$\begin{aligned}&\sum_K \int_K \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} + \\ &\quad \sum_{e \in \varepsilon_h} \int_e [\![\mathbf{v}]\!] : \{\{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!]\} + \{\mathbf{v}\} \cdot [\![\{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!]]\!].\end{aligned}$$

Because of (A.2) and (A.3), it holds that $[\![\{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} - \eta_f [\![\mathbf{u}]\!]]\!] = 0$, so after some reordering the above simplifies to

$$\begin{aligned}&\sum_K \int_K \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\mathbf{v}) \\ &- \sum_{e \in \varepsilon_h} \int_e [\![\mathbf{v}]\!] : \{\mathbf{C} : \varepsilon(\mathbf{u})\} + \sum_{e \in \varepsilon_h} \int_e \eta_f [\![\mathbf{v}]\!] : [\![\mathbf{u}]\!] = \int_\Omega \mathbf{f} \cdot \mathbf{v}.\end{aligned} \tag{A.17}$$

Eliminating $\boldsymbol{\sigma}$. Using $\boldsymbol{\tau} = \boldsymbol{\epsilon}(\mathbf{v})$ in (A.16) gives

$$\sum_K \int_K \boldsymbol{\epsilon}(\mathbf{v}) : \boldsymbol{\sigma} = \sum_K \int_K \boldsymbol{\epsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) - \sum_{e \in \varepsilon_h} \int_e [\![\mathbf{u}]\!] : \{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{v})\}. \tag{A.18}$$

We can now insert (A.18) into (A.17) to get the primal formulation which does not depend on $\boldsymbol{\sigma}$:

$$\begin{aligned}&\sum_K \int_K \boldsymbol{\epsilon}(\mathbf{v}) : \mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u}) \\ &- \sum_{e \in \varepsilon_h} \int_e \Big[[\![\mathbf{v}]\!] : \{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{u})\} + [\![\mathbf{u}]\!] : \{\mathbf{C} : \boldsymbol{\epsilon}(\mathbf{v})\} - \eta_f [\![\mathbf{v}]\!] : [\![\mathbf{u}]\!] \Big] = \int_\Omega \mathbf{f} \cdot \mathbf{v}.\end{aligned} \tag{A.19}$$

Note that the left-hand side of the above equation is symmetric in $\mathbf{u}$ and $\mathbf{v}$.

A.2 Basis for Multiple Complete Cuts

In this section we show for n complete cuts $\mathbf{c}_i$, the harmonic enrichment functions H^i, obtained by solving $\Delta H^i = 0$ for each cut $\mathbf{c}_i$ as described in Section 5.5.2, span the same function space as the canonical basis for the resulting $n+1$ components.

The *canonical Heaviside* functions H_c^i are the enrichment functions that produce the canonical basis. H_c^i is 1 in component C_i and 0 otherwise. The *canonical enrichment basis* of $n+1$ components is

$$\mathcal{C}_{n+1} = \{H_c^1, \ldots, H_c^{n+1}\}.$$

For an un-cut element, which is described by its original basis functions N^a only, the canonical enrichment function is $H_c^1 \equiv 1$. In order to proof that our enrichment basis functions span the same space as the canonical basis, it suffices to show that the space

$$\mathcal{H}_{n+1} = \{1, H^1, \ldots, H^n\},$$

spanned by our harmonic enrichment functions H^i and the constant 1 function corresponding to the original basis, is equivalent to the space spanned by the canonical enrichment functions $\mathcal{C}_{n+1}$.

This can be shown by induction as follows: For the case with $n = 1$ components, the constant 1 function is the only enrichment function and all enrichments are trivially equivalent to the canonical basis.

In the inductive step we assume that for n components the set of enrichment functions $\mathcal{H}_n$ spans the same space as the canonical enrichments $\mathcal{C}_n$. Hence, $\mathcal{H}_n$ spans an n-dimensional space and the H^i are linearly independent.

Then a new cut $\mathbf{c}_n$ splits a component, say C_n, into two parts C_n and C_{n+1}, thus leading to $n+1$ components. The space $\mathcal{H}_{n+1}$ is computed by solving (5.28) for each $H^1, \ldots, H^n$. Note that the i-th functions of $\mathcal{H}_n$ and $\mathcal{H}_{n+1}$ ($1 \leq i \leq n-1$) are not equal, since the new cut imposes additional Neumann conditions at $\mathbf{c}_n$ for (5.28). However, the additional Neumann conditions cannot turn the new functions $H^1, \ldots, H^n$ linearly dependent. Consequently, $\mathcal{H}_{n+1} \setminus H^n$ spans an n-dimensional subspace. The additional new function H^n is linearly independent of $\mathcal{H}_{n+1} \setminus H^n$, since it is the only Heaviside that can control both sides of the new cut $\mathbf{c}_n$ (i.e., C_n and C_{n+1}) independently. This means that $\mathcal{H}_{n+1}$ must be a complete basis for an $(n+1)$-dimensional space and thus is equivalent to the canonical basis $\mathcal{C}_{n+1}$.

APPENDIX B

Notation and Theorems

This appendix covers commonly used notation and summarizes relevant theorems.

B.1 Mathematical Notation

Matrix Entries. The 'bracket operator' with subscript ij returns the entry at the i-th row and j-th column of matrix $\mathbf{A}$:

$$(\mathbf{A})_{ij} := A_{ij}$$

for

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots \\ A_{21} & A_{22} & \cdots \\ \cdots & \cdots & \ddots \end{bmatrix}.$$

Kronecker Delta. The Kronecker delta is defined as

$$\delta_{ij} := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Scalar Product. The scalar product "$\cdot$" between two n-dimensional vectors $\mathbf{a} = (a_1, \ldots, a_n)^T$ and $\mathbf{b} = (b_1, \ldots, b_n)^T$ is defined as

$$\mathbf{a} \cdot \mathbf{b} := \sum_{i=1}^{n} a_i b_i.$$

Outer Product. The outer product "$\otimes$" between two n-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\mathbf{a} \otimes \mathbf{b} := \mathbf{a}\mathbf{b}^T.$$

Colon Operator. The colon operator ":" denotes the tensor product between two matrices $\mathbf{A}$ and $\mathbf{B}$ or between a matrix $\mathbf{A}$ and a 4-tensor $\mathbf{C}$ as

$$\begin{aligned}
\mathbf{A} : \mathbf{B} &:= \sum_{ij} \mathbf{A}_{ij} \mathbf{B}_{ij}, \\
\mathbf{A} : \mathbf{C} &:= \sum_{ij} \mathbf{A}_{ij} \mathbf{C}_{ijkl}, \\
\mathbf{C} : \mathbf{A} &:= \sum_{kl} \mathbf{C}_{ijkl} \mathbf{A}_{kl}.
\end{aligned}$$

Matrix Operators. The trace of an $n \times n$ matrix $\mathbf{A}$ is defined as

$$\mathrm{tr}(\mathbf{A}) := \sum_{i=1}^{n} (\mathbf{A})_{ii}$$

Norms. The 2-norm of a vector $\mathbf{v}$ is defined as

$$\|\mathbf{v}\| := \sqrt{\mathbf{v} \cdot \mathbf{v}}.$$

The Frobenius norm of a matrix $\mathbf{A}$ is defined as

$$\|\mathbf{A}\|_F := \sqrt{\mathbf{A} : \mathbf{A}}.$$

Derivative Operators. A coma denotes partial differentiation. For a scalar function $u : \mathbb{R}^n \to \mathbb{R}$:

$$u_{,i} := \frac{\partial u}{\partial x_i}.$$

The gradient of a scalar function $u : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\nabla u := (\partial u / \partial x_1, \ldots, \partial u / \partial x_n)^T.$$

The divergence of a vector function $\mathbf{v} : \mathbb{R}^n \to \mathbb{R}^n$ with $\mathbf{v} = (v_1, \ldots, v_n)^T$ is defined as

$$\nabla \cdot \mathbf{v} = \operatorname{div} \mathbf{v} := \sum_{i=1}^{n} \frac{\partial v_i}{\partial x_i}$$

The Laplacian of a scalar function $u : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\Delta u = \nabla^2 u := \nabla \cdot \nabla u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}.$$

DG Average and Jump. Unless specified otherwise, the average operator $\{\cdot\}$ and the jump operator $[\![\cdot]\!]$ for scalar-valued functions u and vector-valued functions $\mathbf{v}$ are defined as

$$\begin{aligned} \{u\} &:= \frac{1}{2}\left(u^- + u^+\right), & [\![u]\!] &:= u^-\mathbf{n}^- + u^+\mathbf{n}^+, \\ \{\mathbf{v}\} &:= \frac{1}{2}\left(\mathbf{v}^- + \mathbf{v}^+\right), & [\![\mathbf{v}]\!] &:= \mathbf{v}^- \cdot \mathbf{n}^- + \mathbf{v}^+ \cdot \mathbf{n}^+ \end{aligned} \tag{B.1}$$

where $\mathbf{n}^- = -\mathbf{n}^+$ is the outward normal vector of the element on the '$-$'-side of the edge.

B.2 Theorems

Integration by Parts (Higher Dimensions). For an open bounded subset Ω of $\mathbb{R}^n$ with a piecewise smooth boundary $\partial\Omega$ and continuously differentiable functions u and v, it holds that

$$\int_\Omega \frac{\partial u}{\partial x_i} v = \int_{\partial\Omega} u v n_i - \int_\Omega u \frac{\partial v}{\partial x_i} \tag{B.2}$$

where n_i is the i-th component of the outward unit normal vector $\mathbf{n}$ of Ω.

Integration by Parts (Vector Variant). Replacing v by v_i in Eq. (B.2) and summing over i results in the identity

$$\int_\Omega \nabla u \cdot \mathbf{v} = \int_{\partial\Omega} u\, \mathbf{v} \cdot \mathbf{n} - \int_\Omega u \nabla \cdot \mathbf{v}. \tag{B.3}$$

Divergence Theorem. Setting $u = 1$ in Eq. (B.3) gives

$$\int_{\partial\Omega} \mathbf{v} \cdot \mathbf{n} = \int_\Omega \nabla \cdot \mathbf{v}. \tag{B.4}$$

Variant of the Divergence Theorem. Assuming $\mathbf{v}$ to be constant in Eq. (B.3), i.e. $\partial v_i / \partial x_j = 0$, gives

$$\int_\Omega \nabla u \cdot \mathbf{v} = \int_{\partial\Omega} u \, \mathbf{v} \cdot \mathbf{n}. \tag{B.5}$$

Sum Over Edge Integrals. A common step in deriving DG FEM formulations is to rewrite sums over edge integrals using the average and jump operators. For a mesh consisting of elements K, a bi-valued scalar function u and a bi-valued vector function $\mathbf{v}$, and $\mathbf{n}$ denoting the outward normal of the element, the following identity holds:

$$\sum_K \int_{\partial K} u \, \mathbf{n} \cdot \mathbf{v} = \int_\Gamma [\![u]\!] \cdot \{\mathbf{v}\} + \int_\Gamma \{u\} \, [\![\mathbf{v}]\!]$$

where Γ is the set of all edges (including interior and exterior edges). For consistency, this assumes the following definitions of the outside ('+') values for u and $\mathbf{v}$ on exterior edges:

$$\begin{aligned} \mathbf{v}^+ &:= \mathbf{v}^-, \\ u^+ &:= 0 \end{aligned}$$

which causes the average and jump operators to evaluate to the following on exterior edges of element K^-:

$$\begin{aligned} [\![u]\!] &= u^- \mathbf{n}^-, \\ [\![\mathbf{v}]\!] &= 0, \\ \{\mathbf{v}\} &= \mathbf{v}^-. \end{aligned}$$

Bibliography

[Abdelaziz and Hamouine, 2008] Yazid Abdelaziz and Abdelmadjid Hamouine. A survey of the extended finite element. *Computers and Structures*, 86(11–12):1141–1151, 2008.

[Altmann et al., 2010] Christoph Altmann, Gregor Gassner, and Claus-Dieter Munz. An explicit space-time adaptive discontinuous galerkin scheme. In *Proceedings of ECCOMAS*, 2010.

[Arnold et al., 2000] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and Donatella Marini. Discontinuous Galerkin methods for elliptic problems. In *Discontinuous Galerkin methods (Newport, RI, 1999)*, volume 11 of *Lect. Notes Comput. Sci. Eng.*, pages 89–101. Springer, Berlin, 2000.

[Arnold et al., 2001] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2001.

[Arnold, 1990] Douglas N. Arnold. Mixed finite element methods for elliptic problems. *Comput. Meth. Appl. Mech. Engrg*, pages 281–300, 1990.

[Babuška and Zlámal, 1973] Ivo Babuška and Miloš Zlámal. Nonconforming elements in the finite element method with penalty. *SIAM J. Numer. Anal.*, 10:863–875, 1973.

[Bao et al., 2007] Zhaosheng Bao, Jeong mo Hong, J. Teran, and Ron Fedkiw. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):370–378, 2007.

[Baraff and Witkin, 1998] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1998.

[Bargteil et al., 2007] Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. A finite element method for animating large viscoplastic flow. *ACM Trans. on Graphics*, 26(3):16.1–16.8, 2007.

[Bassi and Rebay, 1997] Francesco Bassi and Stefano Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier-stokes equations. *J. Comput. Phys.*, 131:267–279, 1997.

[Bathe, 1995] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, 1995.

[Beier and Neely, 1992] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH*, pages 35–42, 1992.

[Belytschko and Black, 1999] Ted Belytschko and Tom Black. Elastic crack growth in finite elements with minimal remeshing. *Int. J. Numer. Methods Eng.*, 45(5):601–620, 1999.

[Bielser and Gross, 2000] Daniel Bielser and Markus Gross. Interactive simulation of surgical cuts. In *Proceedings of Pacific Graphics*, pages 116–125, 2000.

[Bielser et al., 1999] Daniel Bielser, Volker A. Maiwald, and Markus Gross. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum (Proceedings of Eurographics)*, 18(3):31–38, 1999.

[Bielser et al., 2003] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. In *Proceedings of Pacific Graphics*, pages 377–386, 2003.

[Bolz et al., 2003] Jeffrey Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 22(3):917–924, 2003.

[Bonet and Wood, 1997] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[Botsch et al., 2005] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for geometry processing. In *11th IMA conference on the Mathematics of Surfaces*, pages 62–83, 2005.

[Botsch et al., 2006] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Symp. on Geometry Processing*, pages 11–20, 2006.

[Botsch et al., 2007] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3):339–347, 2007.

[Brezzi et al., 2000] Franco Brezzi, Gianmarco Manzini, Donatella Marini, Paola Pietra, and Alessandro Russo. Discontinuous galerkin approximations for elliptic problems. *Numerical Methods for Partial Differential Equations*, 16:365–378, 2000.

[Bridson et al., 2002] Robert Bridson, Ron Fedkiw, and John Anderson. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Trans. on Graphics*, 21(3):594–603, 2002.

[Bridson et al., 2003] Robert Bridson, Sebastian Marino, and Ron Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. of Symp. on Computer Animation*, pages 28–36, 2003.

[Capell et al., 2002] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A multiresolution framework for dynamic deformations. In *Proc. of Symp. on Computer Animation'02*, pages 41–47, 2002.

[Carroll et al., 2009] Robert Carroll, Maneesh Agrawala, and Aseem Agarwala. Optimizing content-preserving projections for wide-angle images. *ACM Transactions on Graphics*, 28(3), 2009.

[Carroll et al., 2010] Robert Carroll, Aseem Agarwala, and Maneesh Agrawala. Image warps for artistic perspective manipulation. *ACM Transactions on Graphics*, 29(4), 2010.

[Chaurasia et al., 2011] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. Silhouette-aware warping for image-based rendering. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 30(4):1223–1232, 2011.

[Chen et al., 2008] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):1–14, 2008.

[Choi and Ko, 2002] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3):604–611, 2002.

[Cirak et al., 2000] Fehmi Cirak, Michael Ortiz, and Peter Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Int. J. Numer. Methods Eng.*, 47(12):2039–2072, 2000.

[Cockburn, 2003] Bernardo Cockburn. Discontinuous Galerkin methods. *Z. Angew. Math. Mech.*, 83(11):731–754, 2003.

[de Casson and Laugier, 2000] François Boux de Casson and Christian Laugier. Simulating 2D tearing phenomena for interactive medical surgery simulators. In *Computer Animation 2000*, pages 9–14, May 2000.

[Debunne et al., 2001] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of ACM SIGGRAPH*, pages 31–36, 2001.

[Desbenoit et al., 2005] Brett Desbenoit, Eric Galin, and Samir Akkouche. Modeling cracks and fractures. *The Visual Computer*, 21(8-10):717–726, 2005.

[Devine et al., 1993] Karen D. Devine, Joseph E. Flaherty, Stephen R. Wheat, and Arthur B. Maccabe. A massively parallel adaptive finite element method with dynamic load balancing. In *Supercomputing '93. Proceedings*, pages 2 – 11, November 1993.

[Douglas and Dupont, 1976] Jim Douglas and Todd Dupont. Interior penalty procedures for elliptic and parabolic Galerkin methods. *Computing Methods in Applied Science, Lecture Notes in Physics*, 58, 1976.

[English and Bridson, 2008] Elliot English and Robert Bridson. Animating developable surfaces using nonconforming elements. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):1–5, 2008.

[Faloutsos et al., 1997] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, 1997.

[Floater et al., 2005] Michael S. Floater, Geza Kos, and Martin Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623–631, 2005.

[Forest et al., 2002] Clément Forest, Hervé Delingette, and Nicholas Ayache. Removing tetrahedra from a manifold mesh. In *Proceedings of IEEE Computer Animation*, pages 225–229, 2002.

[Fries and Matthies, 2004] Thomas-Peter Fries and Hermann-Georg Matthies. Classification and overview of meshfree methods. Informatikbericht 2003-03, revised 2004, Institute of Scientific Computing, Technical University Braunschweig, 2004.

[Gee, 1994] James C. Gee. Finite element approach to warping of brain images. *Proceedings of SPIE*, pages 327–337, 1994.

[Gingold et al., 2004] Yotam Gingold, Adrian Secord, Jefferson Y. Han, Eitan Grinspun, and Denis Zorin. A discrete model for inelastic deformation of thin

shells. Technical report, Courant Institute of Mathematical Sciences, New York University, 2004.

[Goferman et al., 2010] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-aware saliency detection. In *CVPR*, pages 2376–2383, 2010.

[Goldenthal et al., 2007] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 26(3):49:1–49:7, 2007.

[Gracie et al., 2008] Robert Gracie, Hongwu Wang, and Ted Belytschko. Blending in the extended finite element method by discontinuous Galerkin and assumed strain methods. *Int. J. Numer. Methods Eng.*, 74(11):1645–1669, 2008.

[Greisen et al., 2012] Pierre Greisen, Manuel Lang, Simon Heinzle, and Aljoscha Smolic. Algorithm and vlsi architecture for real-time 1080p60 video retargeting. *Eurographics / ACM SIGGRAPH Symposium on High Performance Graphics*, pages 57–66, June 2012.

[Grinspun et al., 2002] Eitan Grinspun, Petr Krysl, and Peter Schröder. CHARMS: A simple framework for adaptive simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3):281–290, 2002.

[Grinspun et al., 2003] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proc. of Symp. on Computer Animation*, pages 62–67, 2003.

[Gu et al., 2002] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 21(3):355–361, 2002.

[Guo et al., 2006] Xiaohu Guo, Xin Li, Yunfan Bao, Xianfeng Gu, and Hong Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE Trans. on Visualization and Computer Graphics*, 12(3):375–385, 2006.

[Guttmann et al., 2009] Moshe Guttmann, Lior Wolf, and Daniel Cohen-Or. Semi-automatic stereo extraction from video footage. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 136 –142, Oct 2009.

[Hansbo and Larson, 2002] Peter Hansbo and Mats G. Larson. Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Comput. Methods Appl. Mech. Eng.*, 191(17):1895–1908, 2002.

[Hauth and Strasser, 2004] Michael Hauth and Wolfgang Strasser. Corotational simulation of deformable solids. In *Proceedings of WSCG*, pages 137–145, 2004.

[Huang et al., 2003] Rui Huang, N. Sukumar, and Jean Prévost. Modeling quasi-static crack growth with the extended finite element method – Part II. *Int. J. of Solids and Structures*, 40(26):7539–7552, 2003.

[Hughes, 2000] Thomas J. R. Hughes. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[Irving et al., 2007] Geoffrey Irving, Craig Schroeder, and Ron Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Trans. on Graphics*, 26(3):13.1–13.6, 2007.

[Jacobson et al., 2010] Alec Jacobson, Elif Tosun, Olga Sorkine, and Denis Zorin. Mixed finite elements for variational surface modeling. *Computer Graphics Forum*, 29(5):1467–8659, 2010.

[James et al., 2004] Doug James, Jernej Barbič, and Christopher Twigg. Squashing cubes: Automating deformable model construction for graphics. In *Proceedings of SIGGRAPH '04 Sketches and Applications*, 2004.

[Jerabkova and Kuhlen, 2009] Lenka Jerabkova and Torsten Kuhlen. Stable cutting of deformable objects in virtual environments using the XFEM. *IEEE Computer Graphics and Applications*, 29(2):61–71, 2009.

[Joshi et al., 2007] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 26(3), 2007.

[Ju et al., 2005] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):561–566, 2005.

[Krähenbühl et al., 2009] Philipp Krähenbühl, Manuel Lang, Alexander Hornung, and Markus Gross. A system for retargeting of streaming video. *ACM Transactions on Graphics*, 28(5), 2009.

[Laffont et al., 2010] Pierre-Yves Laffont, Jon Yun Jun, Christian Wolf, Yu-Wing Tai, Khalid Idrissi, George Drettakis, and Sung eui Yoon. Interactive content-aware zooming. In David Mould and Sylvie Noel, editors, *Proceedings of the Graphics Interface 2010 Conference*, pages 79–87, 2010.

[Lang et al., 2010] Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus Gross. Nonlinear disparity mapping for stereoscopic 3d. *ACM Transactions on Graphics*, 29(4), 2010.

[Lew et al., 2004] Adrian Lew, Patrizio Neff, Deborah Sulsky, and Michael Ortiz. Optimal BV estimates for a discontinuous Galerkin method in linear elasticity. *Applied Mathematics Research Express*, 2004(3):73–106, 2004.

[Liu et al., 2009] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics*, 28(3), 2009.

[Martin et al., 2008] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum (Proceedings of Symposium on Geometry Processing)*, 27(5):1521–1529, 2008.

[Melenk and Babuska, 1996] Jens Markus Melenk and Ivo Babuska. The partition of unity finite element method: Basic theory and applications. *Comput. Meth. Appl. Mech. Eng.*, 139:289–314, 1996.

[Mezger et al., 2008] Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Interactive physically-based shape editing. In *Proceedings of ACM Symposium on Solid and Physical Modeling*, pages 79–89, 2008.

[Mirtich, 1996] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.

[Moës et al., 1999] Nicolas Moës, John Dolbow, and Ted Belytschko. A finite element method for crack growth without remeshing. *Int. J. Numer. Methods Eng.*, 46(1):131–150, 1999.

[Moës et al., 2002] Nicolas Moës, Anthony Gravouil, and Ted Belytschko. Non-planar 3d crack growth by the extended finite element and level sets - Part I. *Int. J. Numer. Methods Eng.*, 53(11):2549–2568, 2002.

[Molino et al., 2004] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. on Graphics*, 23(3):385–392, 2004.

[Müller and Gross, 2004] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface*, pages 239–246, 2004.

[Müller et al., 2002] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proc. of Symp. on Computer Animation*, pages 163–170, 2002.

[Müller et al., 2004a] Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point-based animation of elastic, plastic and melting objects. In *Proc. of Symp. on Computer Animation*, pages 141–151, 2004.

[Müller et al., 2004b] Matthias Müller, Matthias Teschner, and Markus Gross. Physically based simulation of objects represented by surface meshes. In *Proceedings of Computer Graphics International*, pages 26–33, 2004.

[Müller, 2008] Matthias Müller. Hierarchical position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulations*, Grenoble, 2008.

[Nealen et al., 2006] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[Niu et al., 2010] Yuzhen Niu, Feng Liu, Xueqing Li, and Michael Gleicher. Warp propagation for video resizing. In *CVPR*, pages 537–544, 2010.

[Nocedal and Wright, 2000] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000.

[Noels and Radovitzky, 2006] Ludovic Noels and Raul Radovitzky. A general discontinuous galerkin method for finite hyperelasticity. formulation and numerical applications. *Int. J. Numer. Meth. Engng.*, 68:64–97, 2006.

[Noels and Radovitzky, 2008] Ludovic Noels and Raul Radovitzky. A new discontinuous Galerkin method for Kirchhoff-Love shells. *Computer Methods in Applied Mechanics and Engineering*, 197:2901–2929, 2008.

[Noels, 2009] Ludovic Noels. A discontinuous Galerkin formulation of non-linear Kirchhoff-Love shells. *Int. J. Numer. Methods Eng.*, 78(3):296–323, 2009.

[O'Brien and Hodgins, 1999] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH*, pages 137–146, 1999.

[O'Brien et al., 2002] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. on Graphics*, 21(3):291–294, 2002.

[Otaduy et al., 2007] Miguel A. Otaduy, Daniel Germann, Stephane Redon, and Markus Gross. Adaptive deformations with fast tight bounds. In *Proc. of Symp. on Computer Animation'07*, pages 181–190, 2007.

[Panozzo et al., 2012] Daniele Panozzo, Ofir Weber, and Olga Sorkine. Robust image retargeting via axis-aligned deformation. *Computer Graphics Forum (Proceedings of Eurographics)*, 31(2pt1):229–236, May 2012.

[Pauly et al., 2005] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutre, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. on Graphics*, 24(3):957–964, 2005.

[Reed and Hill, 1973] Wilmer H. Reed and Timothy R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[Réthoré et al., 2005] Julien Réthoré, Anthony Gravouil, and Alain Combescure. An energy-conserving scheme for dynamic crack growth using the extended finite element method. *Int. J. Numer. Methods Eng.*, 63(5):631–659, 2005.

[Saad and van der Vorst, 2000] Yousef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123(1-2):1–33, 2000.

[Sander et al., 2001] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 409–416. ACM, 2001.

[Schenk et al., 2001] Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, and Andreas Stricker. PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems*, 18(1):69–78, 2001.

[Sethian, 1999] James Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[Shamir and Sorkine, 2009] Ariel Shamir and Olga Sorkine. Visual media retargeting. In *SIGGRAPH ASIA Courses*, 2009.

[Sharf et al., 2007] Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics*, 26(3):43, 2007.

[Shewchuck, 2002] Jonathan Shewchuck. What is a good linear finite element? interpolation, conditioning, and quality measures. In *Proceedings of the 11th International Meshing Roundtable*, pages 115–126, 2002.

[Shewchuk, 1996] Jonathan Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148, pages 203–222. Springer-Verlag, May 1996.

[Sifakis et al., 2007a] Eftychios Sifakis, Kevin G. Der, and Ron Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of Symp. on Computer Animation*, pages 73–80, 2007.

[Sifakis et al., 2007b] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ron Fedkiw. Hybrid simulation of deformable solids. In *Proc. of Symp. on Computer Animation*, pages 81–90, 2007.

[Simo and Fox, 1989] Juan C. Simo and David D. Fox. On stress resultant geometrically exact shell model – Part I. *Comput. Meth. Appl. Mech. Eng.*, 72(3):267–304, 1989.

[Stam, 1999] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Stazi et al., 2003] Furio L. Stazi, Elisa Budyn, Jack Chessa, and Ted Belytschko. An extended finite element method with higher-order elements for curved cracks. *Comput. Mech.*, 31(1):38–48, 2003.

[Steinemann et al., 2006a] Denis Steinemann, Matthias Harders, Markus Gross, and Gabor Szekely. Hybrid cutting of deformable solids. In *Proceedings of IEEE VR*, pages 35–42, 2006.

[Steinemann et al., 2006b] Denis Steinemann, Miguel A. Otaduy, and Markus Gross. Fast arbitrary splitting of deforming objects. In *Proc. of Symp. on Computer Animation*, pages 63–72, 2006.

[Ten Eyck and Lew, 2006] Alex Ten Eyck and Adrian Lew. Discontinuous Galerkin methods for non-linear elasticity. *Int. J. Numer. Methods Eng.*, 67(9):1204–1243, 2006.

[Terzopoulos and Fleischer, 1988a] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4:306–331, 1988.

[Terzopoulos and Fleischer, 1988b] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proceedings of ACM SIGGRAPH*, pages 269–278, 1988.

[Terzopoulos et al., 1987] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of ACM SIGGRAPH*, pages 205–214, 1987.

[Thomaszewski et al., 2006] Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proc. of Symp. on Computer Animation*, pages 107–116, 2006.

[Toledo et al., 2003] Sivan Toledo, Doron Chen, and Vladimir Rotkin. Taucs: A library of sparse linear solvers. http://www.tau.ac.il/∼stoledo/taucs, 2003.

[Wang et al., 2008] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. Optimized scale-and-stretch for image resizing. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 118:1–118:8. ACM, 2008.

[Wang et al., 2009] Yu-Shuen Wang, Hongbo Fu, Olga Sorkine, Tong-Yee Lee, and Hans-Peter Seidel. Motion-aware temporal coherence for video resizing. *ACM Transactions on Graphics*, 28(5), 2009.

[Wang et al., 2010] Yu-Shuen Wang, Hui-Chih Lin, Olga Sorkine, and Tong-Yee Lee. Motion-based video retargeting with optimized crop-and-warp. *ACM Trans. Graph.*, 29(4), 2010.

[Wang et al., 2011] Yu-Shuen Wang, Jen-Hung Hsiao, Olga Sorkine, and Tong-Yee Lee. Scalable and coherent video resizing with per-frame optimization. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 30(4):88:1–88:8, 2011.

[Wempner and Talaslidis, 2003] Gerald Wempner and Demosthenes Talaslidis. *Mechanics of solids and shells: theories and approximations*. CRC Press, 2003.

[Wicke et al., 2005] Martin Wicke, Denis Steinemann, and Markus Gross. Efficient animation of point-sampled thin shells. *Computer Graphics Forum*, 24:667–676, 2005.

[Wicke et al., 2007] Martin Wicke, Mario Botsch, and Markus Gross. A finite element method on convex polyhedra. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3):355–364, 2007.

[Wihler, 2006] Thomas P. Wihler. Locking-free adaptive discontinuous Galerkin FEM for linear elasticity problems. *Mathematics of Computation*, 75(255):1087–1102, 2006.

[Wolf et al., 2007] Lior Wolf, Moshe Guttmann, and Daniel Cohen-Or. Non-homogeneous content-driven video-retargeting. In *ICCV 2007. IEEE 11th International Conference on Computer Vision*, pages 1–6, Oct. 2007.

[Zhang et al., 2009] Guo-Xin Zhang, Ming-Ming Cheng, Shi-Min Hu, and Ralph R. Martin. A shape-preserving approach to image resizing. *Computer Graphics Forum*, 28(7):1897–1906, 2009.

[Zi and Belytschko, 2003] Goangseup Zi and Ted Belytschko. New crack-tip elements for XFEM and applications to cohesive cracks. *Int. J. Numer. Methods Eng.*, 57(15):2221–2240, 2003.

[Zienkiewicz and Taylor, 2000] Olgierd C. Zienkiewicz and Robert L. Taylor. *The Finite Element Method*. Butterworth-Heinemann, 2000.

Curriculum Vitae

Peter Kaufmann

Personal Data

Feb. 13, 1980	Born in Baden, Switzerland
Nationality	Swiss

Education

Dec. 21, 2012	Ph.D. defense
May 2007 – April 2011	Research assistant and Ph.D. student at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology (ETH) Zurich, Prof. Markus Gross.
Oct. 2005	Diploma degree in Computer Science (*mit Auszeichnung*).
Oct. 2000 – Oct. 2005	Diploma studies of Computer Science, ETH Zurich, Switzerland. Specialization: Computational Sciences; Complementary studies: Signal Processing.

Awards

October 2005 — ETH Medal, Diploma Thesis "Holography-Pipeline: From the Wave Front Generation to an Image".

Scientific Publications

P. KAUFMANN, O. WANG, A. HORNUNG, O. SORKINE, A. SMOLIC, and M. GROSS. Finite Element Image Warping, *Proceedings of Eurographics (Girona, Spain, May 6-10, 2013), Computer Graphics Forum*, vol. 32, no. 2, 2013.

B. BICKEL, P. KAUFMANN, M. SKOURAS, B. THOMASZEWSKI, D. BRADLEY, T. BEELER, P. JACKSON, S. MARSCHNER, W. MATUSIK, and M. GROSS. Physical Face Cloning, *Proceedings of ACM SIGGRAPH (Los Angeles, USA, August 5-9, 2012), ACM Transactions on Graphics*, vol. 31, no. 3, pp. 118:1–118:10, 2012.

S. MARTIN, P. KAUFMANN, M. BOTSCH, E. GRINSPUN, and M. GROSS. Unified Simulation of Elastic Rods, Shells and Solids: Implementation Notes, Technical Report No. 721, Institute of Visual Computing, ETH Zurich, 2011.

S. MARTIN, P. KAUFMANN, M. BOTSCH, E. GRINSPUN, and M. GROSS. Unified Simulation of Elastic Rods, Shells, and Solids, *Proceedings of ACM SIGGRAPH (Los Angeles, USA, July 25-29, 2010), ACM Transactions on Graphics*, vol. 29, no. 3, pp. 39:1–39:10, 2010.

S. MARTIN, C. HUBER, P. KAUFMANN, and M. GROSS. Shape-Preserving Animation of Deformable Objects, *Proceedings of Vision, Modeling, and Visualization (VMV) (Braunschweig, Germany, November 16-18, 2009)*, pp. 65–72, 2009.

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS. Implementation of Discontinuous Galerkin Kirchhoff-Lovee Shells, Technical Report No. 622, Institute of Visual Computing, ETH Zurich, 2009.

P. KAUFMANN, S. MARTIN, M. BOTSCH, E. GRINSPUN, and M. GROSS. Enrichment Textures for Detailed Cutting of Shells, *Proceedings of ACM SIGGRAPH (New Orleans, USA, August 3-7, 2009), ACM Transactions on Graphics*, vol. 28, no. 3, pp. 50:1–50:10, 2009.

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM, *Journal of Graphical Models, 2009, Special Issue of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2008*, vol. 71, no. 4, pp. 153–167, 2009.

S. MARTIN, P. KAUFMANN, M. BOTSCH, and M. GROSS. Polyhedral Finite Elements Using Harmonic Basis Functions, *Proceedings of Eurographics Symposium on Geometry Processing 2008 (Copenhagen, Denmark, July 2-4, 2008), Computer Graphics Forum*, vol. 27, no. 5, pp. 1521–1529, 2008. (Best Student Paper Award)

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM, *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (Dublin, Ireland, July 7-9, 2008)*, pp. 105–115, 2008.

R. ZIEGLER, P. KAUFMANN, and M. GROSS. A Framework for Holographic Scene Representation and Image Synthesis, *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 403–415, 2007.

R. ZIEGLER, P. KAUFMANN, and M. GROSS. A Framework for Holographic Scene Representation and Image Synthesis, *In SIGGRAPH '06: Material presented at the ACM SIGGRAPH 2006 conference (New York, NY, USA, 2006), ACM Press*, p. 108, 2006.

S. WÜRMLIN, E. LAMBORAY, M. WASCHBÜSCH, P. KAUFMANN, A. SMOLIC, and M. GROSS. Image-space Free-viewpoint Video, *Proceedings of Vision, Modeling, Visualization (VMV) (Erlangen, Germany, November 16-18, 2005)*, pp. 453–460, 2005.

Patents

B. BICKEL, P. KAUFMANN, B. THOMASZEWSKI, D. BRADLEY, P. JACKSON, S. MARSCHNER, W. MATUSIK, M. GROSS, and T. BEELER. Physical Face Cloning, *U.S. Patent 2012/0185218 A1*, filed October 18, 2011

MPEG Contributions

S. WÜRMLIN, M. WASCHBÜSCH, E. LAMBORAY, P. KAUFMANN, A. SMOLIC, and M. GROSS. Image-space Free-viewpoint Video, *ISO/IEC JTC1/SC29/WG11, MPEG04/M10894, Redmond, WA, USA, July 2004)*.

Employment

From May 2011	Research scientist at The Walt Disney Company Schweiz AG, Disney Research, Zurich.
May 2007 – April 2011	Research assistant at ETH Zurich, Zurich, Switzerland.
Oct. 2005 – April 2007	Software engineer at Green Hippo Ltd, London, UK.
Oct. 2004 – Feb. 2005	Internship at Cyfex AG, Oerlikon, Switzerland.