Acquisition and simulation of deformable objects

vorgelegt von

Rüdiger Schmedding

Dissertation zur Erlangung des Doktorgrades
der Technischen Fakultät
Albert-Ludwigs-Universität Freiburg im Breisgau

Juni 2012

1. Gutachter	Prof. Dr. Matthias Teschner
2. Gutachter	Prof. Dr. Jan Bender
Dekan	Prof. Dr. Bernd Becker
Datum der Disputation	29.6.2012

Abstract

Virtual environments and simulations became increasingly popular in a variety of applications during the last decades. Therefore, approaches for model acquisition and deformable modeling have been extensively researched. Among others, these topics include 3D range scans, robust registration techniques, mesh construction, parameter estimation, deformation models, and stability of simulation systems.

In this thesis, several contributions to the field of simulation of deformable objects are introduced. In the context of 3D data acquisition, a global registration approach is proposed which computes an initial alignment of surface scans independent from the relative positions. The approach is based on geometric surface features which are compared over different objects to find matching point pairs.

To obtain realistic deformation parameters, an estimation approach based on indentation tests is introduced. It measures the exerted force and employs it in a simulation environment. The deformed surface is scanned with a range scanner and is compared to the simulated surface using registration algorithms. The parameters are computed with a gradient descent scheme by minimizing the difference between the simulated and measured displacements.

For the simulation of deformable objects, two approaches are illustrated that improve the stability of simulation environments. In a widely used deformation model, the simulation can fail if elements of the object discretization are inverted. Thus, an approach for accurate inversion handling is proposed to account for this problem. It heuristically assumes that elements are "as uninverted as possible" to locate the most probable inversion direction which is used to resolve the inversion. Further, a velocity-dependent damping approach is proposed that reduces local oscillations and improves both visual appearance and stability of simulations. At the same time, it simplifies the parameter setting compared to existing methods. The approach can also be used for the propagation of external forces, which otherwise proceeds slowly due to the discrete time integration and causes an implausibly weak impression of simulated objects.

Finally, the application of deformable simulations in an approach for path planning in mobile robotics is illustrated. The planning approach considers environment containing deformable objects and uses the simulation environment for the precomputation of deformation energies. These can be incorporated in the planning process, which allows to compute a trade-off between deformation and travel cost. This avoids expensive detours or enables to plan paths in environments where rigid planning approaches fail. During path execution, a learned sensor model allows to distinguish between deformable and unforeseen rigid obstacles to avoid collisions.

Zusammenfassung

Virtuelle Umgebungen und Simulationen werden seit einigen Jahrzehnten in einer wachsenden Zahl von Anwendungen verwendet. Ansätze zur Erstellung von dreidimensionalen Repräsentationen und Modelle für verformbare Objekte wurden deshalb intensiv untersucht. Dazu gehören unter anderem Methoden zur Erstellung von 3D Scans, robuste Registrierung, Netzgenerierung, Verformungsmodelle und Stabilität von Simulationsumgebungen.

In dieser Arbeit werden verschiedene Beiträge zur Simulation von verformbaren Objekten vorgestellt. Zur Registrierung von Oberflächenscans wird ein Algorithmus eingeführt, der unabhängig von der gegebenen relativen Lage der Scans eine passende Ausrichtung findet. Der Ansatz basiert auf geometrischen Oberflächenmerkmalen, die auf verschiedenen Objekten miteinander verglichen werden, um mögliche Korrespondenzen zu finden.

Um realistische Verformungsparameter zu erhalten, wird ein Ansatz zur Schätzung der Parameter vorgeschlagen, der auf gemessenen Kraft-Verformungs-Relationen beruht. Auf ein reelles Objekt werden Kräfte ausgeübt und gemessen, um sie in einer Simulationsumgebung auf ein entsprechendes Modell anwenden zu können. Die verformte Oberfläche wird mit Hilfe eines Tiefenscanners gemessen und mit der simulierten Verformung verglichen. Dafür wird ein Registrierungsverfahren verwendet. Die Parameter werden mit einem Gradientenverfahren berechnet, das die Differenz zwischen simulierter und gemessener Verformung minimiert.

Für die Simulation von verformbaren Objekten werden zwei Ansätze vorgestellt, die die Stabilität verbessern. In einem weit verbreiteten Verformungsmodell ergeben sich Probleme, wenn Elemente der Objektdiskretisierung invertiert sind, was zu einem Zusammenbruch der Simulation führt. Es wird ein Ansatz vorgestellt, der dieses Problem behebt. Dafür wird angenommen, dass Elemente stets „möglichst uninvertiert" sind, um die wahrscheinlichste Richtung herauszufinden, die zur Invertierung geführt hat. Entlang dieser Richtung soll die Invertierung aufgelöst werden. Weiterhin wird ein geschwindigkeitsabhängiger Dämpfungsansatz vorgestellt, der lokale Oszillationen verringert und dadurch sowohl das visuelle Erscheinungsbild als auch die Stabilität verbessert. Gleichzeitig vereinfacht der Ansatz die Wahl der Dämpfungsparameter im Vergleich zu existierenden Methoden. Der Ansatz kann auch zur Propagierung externer Kräfte verwendet werden, was ansonsten durch die diskrete Zeitintegration langsam geschieht und einen unglaubwürdig weichen Eindruck der simulierten Objekte erzeugt.

Schließlich wird die Anwendung von verformbaren Objekten in der Pfadplanung für autonome Roboter gezeigt. Der Planungsansatz erlaubt Umgebungen, die verformbare Objekte enthalten. In einem Vorverarbeitungsschritt werden Verformungsenergien berechnet, die in der Planung berücksichtigt werden. Dadurch kann ein Ausgleich zwischen Verformungs- und Pfadkosten erreicht werden, was lange Umwege vermeidet und Pfade in Umgebungen finden kann, in denen klassische Ansätze versagen. Während der Ausführung des Pfades wird aufgrund eines gelernten Sensormodells zwischen verformbaren und unvorhergesehenen Hindernissen unterschieden, um Kollisionen zu vermeiden.

Acknowledgements

First of all, I want to thank Prof. Dr. Matthias Teschner for excellently supervising my thesis and for giving me the opportunity to pursue my Ph. D. in the Computer Graphics group at the University of Freiburg. The following work would not have been possible without his consistent support and expertise during the last years. I would like to thank him for the excellent conditions in the group and for his interest in my work.

Second, I would like to thank Prof. Dr. Jan Bender from the TU Darmstadt for reviewing my thesis. I would also like to thank the members of the Ph. D. committee at the University of Freiburg.

During the last years, I benefited a lot from my great colleagues in the Computer Graphics group, namely Jonas Spillmann, Markus Becker, Marc Gißler, Markus Ihmsen, Gizem Akinci and Nadir Akinci. They enhanced my work by their valuable ideas and feedback in numerous discussions, and it was always a pleasure to work with them.

Furthermore, I would like to thank my project partner Barbara Frank and the principal investigator Prof. Dr. Wolfram Burgard of project A2 of the collaborative research center SFB/TR-8 from the German Research Foundation for the collaboration on the topic of path planning and the benefit I had from their knowledge.

Thanks also goes to Cynthia Findlay and Veria Popa for handling the administrative tasks and Stefan Teister who cared about all the technical issues.

Finally, I am deeply grateful for the steady support of my family, and I would like to thank my wife Franzi for her love during every period of my life.

This thesis has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A2). Its support is gratefully acknowledged.

Contents

1 Introduction **1**
- 1.1 Contributions . . . 2
- 1.2 Publications . . . 4
- 1.3 Collaborations . . . 5
- 1.4 Thesis outline . . . 5

2 Related work **7**
- 2.1 Object acquisition . . . 7
- 2.2 Parameter estimation . . . 10
- 2.3 Deformable objects . . . 12

3 Simulation framework **15**
- 3.1 Object representation . . . 15
- 3.2 Deformation model . . . 17
- 3.3 Time integration . . . 19

4 Object acquisition **23**
- 4.1 Overview . . . 26
- 4.2 Transformed polynomials . . . 27
- 4.3 Global registration . . . 32
- 4.4 Results . . . 33
- 4.5 Conclusion . . . 37

5 Deformation model **39**
- 5.1 Linear elasticity theory . . . 40
- 5.2 Linear Finite Element Method . . . 49

6 Inversion handling for dynamic simulations **57**
- 6.1 Problem description . . . 59
- 6.2 Approach . . . 60
- 6.3 Results . . . 66
- 6.4 Conclusion . . . 70

7 Parameter estimation **71**
- 7.1 Deformation parameters . . . 72
- 7.2 Illumination parameters . . . 80
- 7.3 Conclusion . . . 85

8 Stability in dynamic simulations **87**
- 8.1 Motivation . . . 88
- 8.2 Damping in dynamic simulations . . . 91
- 8.3 Optimized damping . . . 94
- 8.4 Results . . . 105
- 8.5 Conclusion . . . 107

9 Application **113**
9.1 Path planning in environments containing deformable objects . . 114
9.2 Collision avoidance in environments containing deformable objects 115
9.3 Conclusion 117

10 Conclusion **119**
10.1 Outlook 121

1 Introduction

Today, virtual environments and simulations are used in a great variety of applications, reaching from industrial design over entertainment industry and psychological experiments up to medical applications. The user benefits in various ways. Appropriate simulations allow to check mechanical properties of the planned materials before production, and improvements can be included in the planning process. Thus, they enhance quality and security, and additionally, development costs are reduced. Virtual environments also allow psychologists to analyze human behavior at comparably low costs, as measurement units can be used at fixed locations. In the entertainment industry, they enhance the realism in movies and computer games. Finally, in medical applications, simulators can be used as training centers during the education and support the preoperative planning process.

Consequently, realistic simulations are of great importance in various fields. Especially the simulation of deformable objects is required in many of these applications. Therefore, the realistic model acquisition as well as physically realistic simulations have become popular in the last decades. This comprises topics such as 3D scanning, robust registration, mesh generation and mesh smoothing, parameter estimation, and deformable modeling.

For the construction of spatial representations, 3D point clouds of real objects could be obtained within short time using a range scanner. However, several scans from different directions are needed for a complete model. These have to be aligned in a common coordinate system, which is done by registration algorithms. The Iterative Closest Point algorithm obtains optimal alignments if the relative positions are not too far from the optimum. Thus, it needs user-defined input. To overcome this limit and to achieve a fully-automatic object reconstruction, global registration algorithms have been developed which aim at a coarse initial alignment. They usually compare some kind of surface features like color or geometry to find matching points in different scans. Further, the point clouds are converted to appropriate structures that are needed in the respective environment. In the context of dynamic simulations, volumetric meshes are frequently used.

To provide physically realistic behavior, appropriate deformation parameters have to be acquired if the utilized materials are unknown. This could be done experimentally by exerting forces on the real object and on the reconstructed

model in the simulation environment, which allows to compare the obtained displacements. Minimization strategies then lead to an estimation of the deformation parameters.

Physical realism and efficiency are the main tasks in the simulation environment. Thus, appropriate deformation models have to be used based on physical laws which at the same time allow for an efficient computation of internal forces. Meeting these requirements, the so called co-rotational Finite Element Method is widely used. Moreover, stability is a fundamental issue as it allows to choose larger time steps and, thus, leads to a faster simulation. For example, velocity-dependent damping approaches reduce oscillations and allow the simulation of badly shaped meshes, which otherwise heavily influence the stability of the simulation. Therefore, they are widely used as they enhance both the visual plausibility and the stability of deformable objects.

1.1 Contributions

This thesis presents various contributions to the simulation of deformable objects in Computer Graphics and its applications. It addresses the fields of automatic object reconstruction concerning spatial representations as well as parameter estimation, and the field of stability in physically-based animation.

1.1.1 Global registration of point clouds

As outlined above, the registration of point clouds is an essential step within the object reconstruction pipeline. Towards the automatic reconstruction of object models, the automatic composition of different depth images obtained by range scanners is required. As the Iterative Closest Point algorithm aligns objects locally, a coarse initial alignment has to be provided as input data. Therefore, a novel global registration approach for partially overlapping point clouds is introduced to overcome this limitation. The approach identifies feature points of matching objects based on surface-approximating polynomials and finds an initial transformation depending on these polynomials. In contrast to purely feature-based approaches, the aligning transformation is not solely based on the invariant properties of polynomials. Instead, the polynomials are transformed into a common coordinate system to compare the transformed coefficients. This results in an improved correspondence analysis of local surfaces. Hence, the transformed polynomials provide more discriminating information about different structures. Therefore, the approach can handle partial scans of different objects simultaneously, which are assigned and registered accordingly.

1.1.2 Parameter estimation

In order to acquire realistic models for deformable objects, an approach for the estimation of deformation parameters for the co-rotational Finite Element Method is presented. It exerts forces on the objects that should be estimated, and obtains the deformation by a range scanner. Using a force-feedback sensor,

the measured forces can also be applied to a constructed model in the simulation environment. The deformed real and simulated object surfaces are registered using the Iterative Closest Point algorithm in order to compare the resulting displacements. Assuming that smaller displacements correspond to better chosen parameters allows to compute the deformation parameters by a gradient descent scheme. The approach has been realized on a mobile system and has proven to be able to acquire parameters of real-world objects. Although the approach is formulated for the co-rotational Finite Element Method, the idea could be applied to any deformation model.

1.1.3 Deformable modeling

In the context of deformable modeling, several contributions are presented that improve the stability of dynamic simulations. Therefore, they allow for larger time steps and, thus, for a more efficient simulation.

Inversion handling. The linear Finite Element Method is a widely used deformation model, which achieves physically realistic simulations and computational efficiency. However, it is valid only for small deformations, and especially, rotations are not handled correctly. Therefore, the co-rotational Finite Element Method [HS04, MG04] introduces a separate rotation handling to improve the usability of the linear Finite Element Method. However, this imposes a new issue if an element gets inverted, i. e. if a vertex of a tetrahedron crosses its opposite face. The calculated forces keep the tetrahedron inverted and do not restore the correct resting state. This causes the simulation to fail. As the inversion of elements cannot be safely avoided, there is a strong need for an appropriate handling of inverted elements. Therefore, a novel approach to resolve inverted elements is introduced that is based on the heuristic assumption that a tetrahedron is "as uninverted as possible" [ITF04]. Thus, it chooses the shortest distance between a vertex and its opposite face as the most intuitive direction that caused the inversion, and resolves the inverted state in this direction. In combination with an efficient handling of degenerated elements, the approach yields a stable simulation of arbitrary deformations. Although the approach is formulated for the co-rotational formulation of the linear Finite Element Method, the method can be implemented within a wide range of constitutive models.

Damping. As the equation of motion is integrated numerically, dynamic simulations frequently suffer from oscillations. Therefore, usually a velocity-dependent damping force is introduced which suppresses the oscillations and greatly improves the stability and visual appearance of simulations. Furthermore, also badly shaped meshes containing sharp angles or slivers can be handled if a proper damping approach is used. At the same time, however, it must be ensured that the damping forces do not change the global movement of objects, i. e. the linear and angular momentum have to be conserved. The so called spring damping approach fulfills this property. On this basis, a novel damping approach is proposed that employs iterative spring damping to further improve the stability. It is shown that the resulting iterative forces can be com-

puted directly without actually performing the iterations. The approach does not require any connectivity information about the object and therefore, it can be used for arbitrary object representations. Further, it is independent of the integration scheme and the chosen deformation model. The approach provides a simple parameter setting and guarantees that the damping forces do not overshoot. It is shown that the approach allows for larger time steps than existing damping models.

Force propagation. Due to the object discretization and discrete time integration, external forces are propagated only one element per time step. This behavior is independent of the object's stiffness and causes an implausibly weak impression, which contradicts the natural experience. Further, it causes large displacements of those vertices exposed to external forces, which results in large strain values that could affect the object's stability. It is illustrated that the damping approach can be used for a fast propagation of external forces caused by constraints or collisions, for example. Therefore, the simulation gets less sensitive to large external forces, which again improves the stability of dynamic simulations.

1.2 Publications

This thesis is based on the following peer-reviewed publications in conference proceedings and journals:

[**ST08**] Ruediger Schmedding, Matthias Teschner, **Inversion Handling for Stable Deformable Modeling**, *The Visual Computer*, vol. 24, no. 7-9 (CGI 2008 Special Issue), pp. 625-633, 2008.

[**SGT09**] Ruediger Schmedding, Marc Gissler, Matthias Teschner, **Optimized Damping for Dynamic Simulations**, *Proc. Spring Conference on Computer Graphics*, Budmerice, Slovakia, pp. 205-212, April 23-25, 2009.

[**SGT10**] Ruediger Schmedding, Marc Gissler, Matthias Teschner, **Fast Force Propagation in Dynamic Simulations Using Optimized Damping**, *Journal of Computer Graphics & Geometry*, vol. 12, no.2, pp. 60-77, 2010.

[**SFBT11**] Ruediger Schmedding, Barbara Frank, Wolfram Burgard, Matthias Teschner, **Transformed Polynomials for the Global Registration of Point Clouds**, *Proc. Spring Conference on Computer Graphics*, Viničné, Slovakia, pp. 157-164, April 28-30, 2011.

[**GST09**] Marc Gissler, Ruediger Schmedding, Matthias Teschner, **Time-critical Collision Handling for Deformable Modeling**, *Journal of Computer Animation and Virtual Worlds (CAVW)*, vol. 20, no. 2-3, pp. 355-364, CASA 2009 Special Issue, 2009.

[**FSS*09**] Barbara Frank, Cyrill Stachniss, Ruediger Schmedding, Matthias Teschner, Wolfram Burgard, **Real-world Robot Navigation amongst**

Deformable Obstacles, *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 1649-1654, May 12-17, 2009.

[**FSS*10a**] Barbara Frank, Ruediger Schmedding, Cyrill Stachniss, Matthias Teschner, Wolfram Burgard, **Learning Deformable Object Models for Mobile Robot Path Planing using Depth Cameras**, *Proc. Workshop RGB-D: Science and Systems RSS*, 2010.

[**FSS*10b**] Barbara Frank, Ruediger Schmedding, Cyrill Stachniss, Matthias Teschner, Wolfram Burgard, **Learning the Elasticity Parameters of Deformable Objects with a Manipulation Robot**, *Proc. IEEE / RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1877-1883, 2010.

1.3 Collaborations

This thesis has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 (A2). The SFB/TR-8 is a special research area with a transregional collaboration between Bremen and Freiburg. It is concerned with topics from different research areas such as Artificial Intelligence, Robotics, Cognitive Science, Linguistics and Computer Graphics. Currently, the SFB/TR-8 is funded in the third phase from 2011-2014.

Project *A2-[3DSpace]* is a member of SFB/TR-8. During the second phase, it was concerned with mobile robot navigation in environments containing deformable objects. In the third phase, it is concerned with the automatic reconstruction of environments based on human navigation. The tight collaboration within project A2 resulted in the joint publications [FSS*09, FSS*10a, FSS*10b, SFBT11] with my research partner Barbara Frank and the principal investigators of project A2, Prof. Dr. Wolfram Burgard and Prof. Dr. Matthias Teschner.

1.4 Thesis outline

This thesis is arranged as follows. In Chapter 2, the related work on the topics of this thesis is discussed. It starts with a summary of object reconstruction, where the focus lies on global registration algorithms. Then, an overview of the field of parameter estimation is given. Then, the related work in deformable modeling is discussed.

In Chapter 3, a simulation framework for deformable objects is introduced which has been used in this thesis. The approaches proposed in Chapter 6 and Chapter 8 have been integrated within this framework. The chapter summarizes the object representation and the numerical computation of object dynamics.

The subsequent chapters are oriented at the steps of the model acquisition and simulation pipeline, and the contributions are assigned accordingly. Results of the developed methods are given within the according chapter.

In Chapter 4, the acquisition of 3D object representations is discussed. As range scanners have a limited field of view, a 3D point cloud of a real object consists of several scans from different directions. In order to align them in a common coordinate system, a feature-based approach for the global registration of partially overlapping point clouds is introduced. The experiments show that the obtained results are almost optimal without any local refinement.

Chapter 5 reviews the physical background for linearly elastic objects and introduces the co-rotational Finite Element Method which is employed in this thesis. Building on that, an approach for the handling of inverted elements is proposed in Chapter 6. Without appropriate inversion handling, inverted elements cause the breakdown of the simulation. Thus, the inversion handling removes a fundamental drawback of the co-rotational Finite Element formulation.

In Chapter 7, an approach for the estimation of appropriate deformation parameters is developed. The approach is based on indentation tests and local registration algorithms and has been realized on a mobile system. Further, an approach for the acquisition of visualization parameters is outlined.

Chapter 8 is concerned with the stability of dynamic simulations. It introduces an optimized damping algorithm that yields a great improve of stability and thus, it allows to choose larger time steps. This results in a more efficient simulation.

Finally, the integration of the simulation system into a mobile path planning algorithm is illustrated in Chapter 9. This allows to plan paths in environments with deformable objects by a trade-off between path and travel cost. Further, a system for collision avoidance during path execution is presented.

2 Related work

In this chapter, the previous work in the context of deformable objects is introduced. In Sec. 2.1, approaches for the reconstruction of 3D models are presented. This comprises e. g. 3D scanning, mesh smoothing and algorithms that compute spatial data structures like tetrahedral meshes from point clouds. However, the focus is on global registration approaches, which is one of the contributions of this thesis. In Sec. 2.2, different approaches for the estimation of physical parameters are introduced. Sec. 2.3 gives an overview about deformation models in Computer Graphics and approaches to improve the stability of dynamic simulations.

2.1 Object acquisition

The construction of 3D representations for virtual environments comprises a variety of methods [BR02, VBS09, KAHD10]. Generally, a three-dimensional point cloud describing the object's surface is obtained first by a range scanner. To obtain a complete object, usually several scans have to be aligned in a common coordinate system. Smoothing techniques can be applied in order to reduce noise, before the point cloud is converted into the structures that are needed. E. g., polygonal meshes are employed in many applications. Thus, this section briefly discusses the related work for the point cloud construction, mesh smoothing and mesh generation techniques, while it mainly focuses on the literature about registration approaches that concern the contribution of this thesis.

For the acquisition of 3D geometry, there are contact based methods that touch an object in order to obtain the coordinates of surface points as well as non-contact based methods [Cur99, RCM*01]. In the last decades, the non-contact based methods have become popular. Among these are approaches like photometric stereo [Woo80, Woo84], and active techniques that send a signal, e. g. light or sound, which is reflected and interpreted appropriately to obtain a depth value. Although their underlying techniques like the triangulation method and the sound velocity are known for centuries, the applicability of these principles was limited and increased with the availability of computers and electronic measurement devices at comparably low costs. [WMW06] presented a low-cost laser

scanner technique and meanwhile extended this to a structured light scanner[1]. A low-cost structured light scanner was also introduced by [RCM*01]. [LPC*00] presented a system for the digitalization of large objects. [CBS00, Bla04] present an overview about the existing techniques and their particular advantages.

To obtain triangular surfaces from point clouds, [HDD*92] employs a signed distance function. Further, [ABK98] use Voronoi diagrams, and [BMR*99] employ a wrapping technique to obtain triangular surfaces. In the context of 3D scans, surface reconstruction techniques have to be especially robust against noise [KBH06]. Also, noise reduction can be applied before the triangulation process. Methods like Moving Least Squares [Lev98, Lev04] are frequently applied to obtain a point cloud representing a smooth surface. Improved variants have been developed that also account for sharp surface features [FCOS05, LCOL07]. While Moving Least Squares computes an interpolating function for surface smoothing, other techniques work directly on the point cloud [LCOLTE07, HLZ*09]. Similar techniques can also be used for mesh simplification [PGK02] or up- and downsampling of surfaces [ABCO*01, AK04].

Further, 3D scans of real objects are likely to be incomplete. To overcome such problems, mesh repair algorithms have been developed. A wide-spread approach is to compute a volumetric object whose surface approximates the given point cloud [CL96], which returns a closed surface via isosurface extraction. In many approaches, implicit functions are used [TO02, SOS04]. For example, a distance field can be employed [HK06, JBS06, PBL09]. Similar approaches based on distance fields [TO02, SWT06] can also be used to construct volumetric meshes. [SLS*07] propose an algorithm with user-interaction to improve the surface reconstruction. [Ju09] provide a survey about current techniques for mesh repair.

Registration

As the contribution of this thesis concerns the global registration problem, the related work of this topic is introduced in greater detail now.

Registration, correspondence and matching algorithms are applied in various research fields such as object reconstruction, shape retrieval, and symmetry detection. The general task is to find matching parts of different objects with unknown relative orientations. These could be several scans of an object that should be aligned in a common coordinate system [RHHL02, GMGP05, AMCO08], or a partial scan of an object which is used to identify a corresponding model in a given library [KFR04, PMG*05, GCO06, SSSCO08]. Symmetry detection approaches look for similar sub-parts of objects [MGP06, MGP07, PMW*08, RBBK10, BBK09]. A recent survey can be found in [vKZHCO11].

All these tasks can be tackled in a similar manner by looking for a transformation between pairs of objects to align them properly. Given at least three correspondent points on each object, such a transformation could be easily computed [FH86, Hor87, LEF95]. Therefore, the task to establish an appropriate

[1] http://www.david-laserscanner.com, accessed November 8, 2011

transformation is equivalent to finding appropriate correspondences between the query objects and computing a transformation based on these correspondences.

Rigid Registration. The Iterative Closest Point algorithm (ICP) [BM92, CM91, Zha94] is a well-known method for aligning overlapping point clouds. As outlined above, it looks for corresponding points on different objects, for example based on shortest distances, and computes a transformation using these correspondences. Much work has been done to improve the convergence rate and the results of the original algorithm, for example using better metrics and sophisticated criteria for corresponding points [Pul99, RL01, MGPG04]. However, the ICP algorithm only optimizes locally and depends on a good initial guess of the relative orientations. This leads to the well-known problem of global registration, where a coarse initial alignment is computed.

In general, objects overlap only partially. Therefore, approaches using global quantities like principal component analysis are not applicable. A straightforward approach to get an initial alignment is to enumerate all three-point-pairs in different objects, compute the corresponding transformation and choose the one that leads to the smallest error with respect to some error metric. For a given triplet in one object, this leads to $O(n^3)$ possibly matching triplets. In case of outliers and noise, the RANSAC algorithm [FB81] can be applied, and [CHC98, CHC99] build up on RANSAC to get an improved global registration algorithm. [AMCO08] also build up on RANSAC, but choose coplanar four-point sets instead of triplets to establish an initial transformation. For a given four-point base, this method only requires to examine all pairs of points instead of all triplets by using affine invariant ratios, which reduces the complexity to $O(n^2)$.

Similarly, voting methods like the Hough transform also employ the fact that three correspondent points are sufficient to compute an initial alignment. The six-dimensional space of possible transformations is subdivided into cells and a transformation for each triplet of points on the different objects is computed. Each computed transformation results in a vote for the corresponding cell, and finally, the transformation represented by the cell with the largest number of votes is chosen as an initial alignment which could be refined using ICP.

Using surface properties like geometry or color can improve the voting and RANSAC-based methods as well as the correspondence search in the ICP algorithm. While color, for example, is used for the matching of fresco fragments [BTFN*08, TFBW*10], geometric features can be used for arbitrary point clouds obtained by a range scanner. Various appropriate geometric descriptors are applied for surface alignment. Among these are differential quantities like curvature, which could be computed based on approximating polynomials [GCO06, CP03], integral invariants [PWY*07] or volumetric descriptors [GMGP05, HFG*06]. Also, spin images [JH97, Joh97] are used to establish a global initial alignment. In addition, spatial relations between points with similar features are used to enhance the registration [GMGP05, AMCO08].

Non-rigid Registration. Most of the above-mentioned approaches are concerned with rigid registration problems. While articulated objects could be matched using rigid registration algorithms by dividing objects into parts

[GMGP05, JZ07], deformable objects have to be handled in a different manner. Therefore, there is a large research area dealing with non-rigid registration and similarity problems [BR07, HAWG08, RBBK10]. The possible transformations are extended allowing deformations instead of rigid transformations, and the goal is to minimize some deformation energy term [HAWG08]. For example, thin-plate splines are used for this purpose [BR07]. Other methods are concerned with finding correspondences between non-rigid objects, which are possibly deformed, or pose-invariant correspondences [OSG08, GSCO07], e. g. between a standing and a sitting dog, without any knowledge about the relative positions. Many approaches are based on geodesic distances [HAWG08, TBW*09] or surface metrics which are invariant under isometric deformations [LZSCO09]. The so called Möbius voting algorithm [LF09] makes use of the fact that isometries of simply-connected surfaces are contained in the Möbius group and performs a voting scheme similar to the Hough transform. Also, multi-dimensional scaling [RBBK10], spectral correspondences [JZ06, JZvK07] and skeleton-based approaches [ZST*10, KCATCO*10] are used to perform non-rigid registration. Further information about non-rigid registration can be found in the thorough overview of [vKZHCO11].

2.2 Parameter estimation

For a realistic simulation, a spatial representation of an object has to be equipped with appropriate deformation parameters. The estimation approaches can be separated into data-driven approaches and analytical approaches [LSH07]. The latter are e. g. applied in the context of mass-spring-models. Starting from a continuum formulation, spring stiffness values are estimated in order to obtain an efficient approximate simulation model. Thus, known parameters for the continuum formulation are mapped onto appropriate values for the spring stiffness. [EGS03] derive a particle system based on the continuum equation by a Finite Differences discretization. [vG98] derive analytical expressions based on geometric considerations, and similarly, [MBT03] derive analytical equations from the elasticity values by a uniaxial elongation test. The approaches allow for spatially varying spring constants, as constant stiffness values do not achieve a sufficient approximation. [LSH07] showed that for a special Poisson ratio, the stiffness matrix of mass-spring-systems can equal a triangular Finite Element formulation, and found analytical expressions for the spring parameters depending on the physical stiffness. For different Poisson ratios, he introduced a minimization scheme to find appropriate values.

As the topology of mass-spring-systems influences the deformation behavior, not only the parameters are challenging, but also the estimation of an appropriate topology. This was tackled by a data-driven approach in [BHS03, BSSH04], who use a Finite Element formulation as reference to obtain appropriate mesh topologies and stiffness parameters. [BC00] use different topologies to approximate isotropic or anisotropic behavior. Also, [SVAC11] try to identify appropriate cubical mass-spring-meshes, which reflect a given material behavior, before

estimating appropriate parameters by fitting a non-linear Finite Element reference simulation. Thus, this approach can be seen as a combination of analytical model construction and data-driven parameter fitting. [BCP*08] use a particle filter to obtain a posterior distribution over the stiffness parameters and evaluate the particles by comparing simulated and observed deformations. [DKT95] used simulated annealing to fit the displacements of vertices. [SVAC11] present a notable overview about the identification of parameters in mass-spring-networks.

In the context of physical deformation models, the deformation parameters can be estimated using a measurement of the force-displacement-relation [Har67]. In contrast to mass-spring-models, they are independent from the discretization of the spatial object representation. Therefore, a standard technique is to apply forces to an object which are observed by a force sensor and to capture the resulting displacement. This can be compared to a simulated model that is deformed by the given forces, which allows to perform an iterative refinement on the simulation parameters to achieve the best match [KL04, ABB*08].

Thus, the estimation of physical parameters is based on an interaction with the deformable object. Such interactions between tools and soft tissue are immanent in medical simulations [MRO08], which are used for training systems and in preoperative planning approaches, for example. Therefore, realistic deformation parameters are needed, and several approaches have been developed in this context. [KVD*01] present an inverse Finite Element estimation of biological soft tissues, which are modeled as non-linear viscoelastic materials. Similarly, [KS05] use an iterative scheme for parameter estimation of soft tissues based on a non-linear viscoelastic Finite Element model. Further, [SGN*05] estimate the deformation parameters of brain tissue, where they use intraoperative MR scans and image registration techniques to compare the deformations.

In more general contexts, [SZ92] employ a non-linear least squares optimization to compute the Young modulus for a given Poisson ratio in a 2D Finite Element formulation with boundary conditions. Also, [ZZ94] use a 2D Finite Element Method for thin plates for the estimation of stiffness values in case of bending. [Lan01, LPW02] present a parameter estimation for the Boundary Element Method [JP99] which is based on Green's function. They formulate a linear estimation problem to obtain Green's function matrix irrespective of its analytical and numerical derivation. [CZ05, ZCLH09] employ an analytical formulation of [HKHM72] to model a two-sided indentation test for the simultaneous estimation of Young modulus and Poisson ratio of soft tissues, which is applied on simulated data. [BT07] exploit the structure of the stiffness matrix in the linear Finite Element Method to formulate a Quadratic Programming approach for the estimation of stiffness parameters, which is also applied to simulated data sets. [SB08] capture the deformation of objects from videos for a comparison with simulated objects to estimate their mechanical properties. [Fon09] extract force-fields for different contact points and displacements on the objects and employ a structured light scanner for deformation observation. For unseen contact points, the forces are interpolated using radial basis functions. Similarly, [BBO*09] present a data-driven representation of heterogeneous and non-linear material by fitting radial basis functions to different measured force-

displacement samples in a Finite Element simulation. [CPP10] estimate the deformation parameters using a grasping robotic hand and capture the deformation by a 2D side view of the deformed contour.

2.3 Deformable objects

The simulation of deformable objects in Computer Graphics started with the pioneering work of Terzopoulos [TPBF87], who introduced the research area of physically plausible, yet efficient and interactive simulations in Computer Graphics. The deformation models are based on differential geometry formulations as well as elastic and visco-elastic mass-spring-systems [TPBF87, TF88a]. Also, plastic behavior and fracture has been integrated [TF88b].

In the following years, a great variety of deformation models had been introduced, which can be separated into two classes. On the one hand, methods based on continuum mechanics are employed to obtain realistic simulations. On the other hand, heuristic approaches like mass-spring-systems are designed to obtain a similar behavior within less computation time.

Models based on continuum theory usually discretize the underlying differential equations that define the relation between displacements and internal forces, which act to withstand the deformation. They generally partition an object into a set of mass points and deduce forces acting on these mass points from the discretized continuum. The method of Finite Differences approximates the derivatives by a differential quotient on a regular grid [TPBF87, EGS03]. The Boundary Element Method [JP99, JP02, JP03] employs Green's functions that allow to proceed a solution of a differential equation given at the boundary into the whole volume. Therefore, only a solution at the boundary has to be found. In the Finite Volume Method [TBHF03, BH11], the object is partitioned into discrete elements surrounding the mass points, and the stress tensor is directly integrated over the surface of the elements to obtain the forces. Similarly, the Finite Element Method [Bat96, GTT89, CZ92, DDCB01], which is probably the most popular among these approaches, partitions the object into polyhedral elements, whose vertices are given by the mass points. Then, the differential equation is converted into an algebraic equation which defines a discrete solution at the vertices.

While a non-linear Finite Element formulation is used in [OH99, OBH02] to simulate brittle and ductile fracture, mostly linear formulations are applied in order to gain fast calculations. However, these are only valid for small deformations and result in the drawback that rotations are not handled correctly, but cause ghost forces. To alleviate this problem, [TW88, CGC*02] introduced a reference frame moving with the object or with manually selected sub-parts of the object, respectively. [MDM*02] proposed to compute a rotation for each vertex of a mesh, which improves the applicability of linear formulations, while still leading to small ghost forces. This problem was overcome by computing rotations per element instead of vertices, which was first introduced by [EKS03] in the context of cloth simulation. [EKS03] compute the rotation by a polar

decomposition of the deformation gradient, which was adopted to general Finite Element formulations by [HS04, MG04]. Hence, due to the so called corotational formulation, the linear Finite Element Method became applicable for a wide range of deformations.

In most cases, the Finite Element Method is formulated on tetrahedral meshes. However, this imposes some restriction on topological changes such as cutting which is necessary in applications like virtual surgery [MGAT09, CDA00, NvdS01, PDA01]. In order to avoid remeshing which may lead to ill-shaped meshes, [WBG07] develop a Finite Element formulation on general convex polytopes. Further applications of the Finite Element Method can be found in the simulation of human data like muscles [ZCK98, HFS*01] and game environments [PO09], for example.

Among the heuristic approaches, the mass-spring-systems [CZKM98, BC00, GW05] are the largest part due to their simplicity and computational efficiency. Mass-spring-models can also be interpreted as a constraint method that computes penalty forces in order to keep the length of an edge. E. g., [BB08, BDB11] employ distance constraints for the simulation of inextensible cloth. However, they use impulse based dynamics [Ben07, BDB09] instead of a force-based formulation in order to avoid the stiff differential equations occurring in the spring force approach. The constraint interpretation of springs was generalized in [THMG04], where area and volume constraints are defined and penalty forces are computed if the constraints are violated. Similarly, [DBB09b, DBB09a] employ volume constraints for the simulation of incompressible deformable objects, where they also use an impulse-based formulation instead of penalty forces.

So far, the approaches were based on meshes that connect the discrete mass points. However, there is another class of mesh-free approaches that do not rely on connectivity structure between points [MKN*04, AOW*08]. A geometrically motivated method was presented in [MHTG05], which computes goal positions based on a geometric matching. This approach was improved by [RJ07], who apply the geometric matching onto appropriate subparts, achieving non-linearly seeming deformations with a linear deformation formulation. [DBB11] apply the geometric matching to the surface of the object, which results in a performance gain compared to [RJ07]. In combination with an efficient volume preservation method, [DBB11] obtain a real-time simulation of incompressible deformable objects. Further, the approach of [MHTG05] was applied in cloth simulation by [SSBT08]. [MC11] combine the approaches of [MKN*04] and [MHTG05] for stable simulations of sparse structures such as chains.

An overview over the development of deformable models in Computer Graphics can be found in the report of Nealen et al. [NMK*06].

Efficiency is an essential task in interactive simulations. Therefore, stability of deformation models is an important topic, which is influenced by the chosen integration scheme [HES03] and the properties of the deformation model. Also, the mesh quality influences the chances to obtain a stable simulation [She02, FSH10, FSAH11].

The deformation model in [MHTG05] takes the integration scheme into account and results in an unconditionally stable simulation. However, this can-

not be generalized to other deformation models. Reduced deformable models [HSO03, CK05, KP11] employ the modal analysis [PW89] to eliminate high-frequency modes to allow for larger time steps. [GW08] improve the stability of the co-rotational Finite Element formulation by an energy minimization. [MHHR07] use a position based formulation of dynamics, which avoids overshooting problems that occur in velocity- or force-based models.

Another possibility is to introduce velocity-dependent damping forces in order to constrain the kinetic energy, which can be referred to as viscous damping [AW01]. With appropriate damping parameters, this results in a great improve of stability and allows for significantly larger time steps. Moreover, it reduces oscillations and results in a more realistic simulation. Therefore, damping is commonly applied in physically-based animation.

Viscous damping was already used by [TPBF87, TF88a]. [PB88] proposed improved damping forces for Finite Elements which depend on the strain rate tensor. These were also used by [CYMTT92] and [OH99], for example. Further, [PB88] introduced damping forces for constrained motions that depend on the time-derivative of a point-to-nail constraint. [BW98] generalized this kind of damping for arbitrary constraints choosing damping forces that depend on the time derivative of the constraints. A damping model which is independent of the deformation model, but relies on the connectivity of the object is described in [NMK*06]. An overview about different kinds of damping can be found in [KYOK09].

3 Simulation framework

In this chapter, the simulation environment *DefCol Studio*[1,2] is described in order to give an overview over the topics that are discussed in this thesis. The approaches in Chapter 5 and Chapter 8 are implemented within this environment. *DefCol Studio* provides methods for the interactive simulation of deformable objects. This comprises deformation models [MG04, THMG04, MHTG05], numerical time integration, and collision handling for deformable objects [THM*03, HTK*04, ST05, SBT07].

This chapter is organized as follows. First, the spatial representation of objects within *DefCol Studio* is introduced (Sec. 3.1). Then, a basic deformation model for the calculation of internal forces is given (Sec. 3.2), before the numerical time integration is illustrated (Sec. 3.3). As the contributions of this thesis do not affect the collision handling, which is also included in *DefCol Studio*, this part is not introduced separately. An overview of this topic can be found in [TKH*05], while the implementation in *DefCol Studio* is given in the above references.

3.1 Object representation

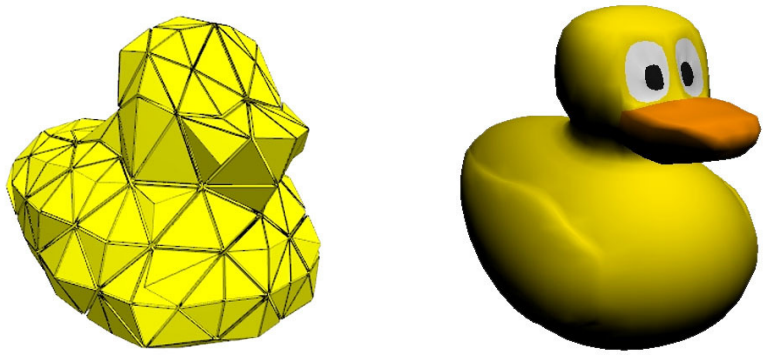
The internal forces acting on an object usually depend on derivatives of the spatial displacement, which results in a partial differential equation. In general, this equation cannot be solved analytically. Thus, the object domain has to be discretized in order to obtain a numerical solution.

In the initial state, an object O occupies a well-defined spatial domain $\Omega \subset \mathbb{R}^3$. Any deformed state of O can be defined as a mapping $\Phi\colon \Omega \to \mathbb{R}^3$, where the deformed object occupies the space $\Phi(\Omega)$. Thus, the differential equation is also defined in the domain Ω, which has to be discretized in order to solve for the internal forces.

In $\mathbb{R}^3$, the simplest polytopes, i. e. the polytopes with the smallest number of vertices, are tetrahedrons. Tetrahedrons are flexible in order to approximate

[1] Heidelberger, B.: DefCol Studio - Interactive deformable modeling framework. http://www.beosil.com/projects.html, accessed November 11, 2011

[2] Teschner, M.: DefCol Studio 1.0.0, http://cg.informatik.uni-freiburg.de/software.htm, accessed November 11, 2011



(a) (b)

Figure 3.1: Illustration of the spatial data structures for the object representation. (a) illustrates a tetrahedral mesh, which is used for the simulation. (b) illustrates the corresponding surface mesh.

spatial structures, and therefore, they are frequently used to discretize the spatial domain Ω. Their vertices are taken as sample points, where the differential equation is evaluated. Internal and external forces as well as other physical properties like velocity and mass are restricted to the sample points, which are also called *mass points*. In a proper discretization, neighboring tetrahedrons share their vertices, edges and triangles. That means, a vertex of one tetrahedron cannot lie inside a triangle of a neighboring tetrahedron, but has to be a vertex of this tetrahedron, too. The discretization of Ω using tetrahedrons is called a *tetrahedral mesh* or *volumetric mesh* of O.

As neighboring tetrahedrons share their vertices, edges and triangles, each inner triangle is contained in exactly two tetrahedrons. Those triangles that belong to one tetrahedron only form the outer surface of the tetrahedral mesh and could be used for the visualization. However, to be efficient in computation time, one aims at a coarse sample point density, which leads to comparably large tetrahedrons and, thus, to a poor visualization with low detail. Therefore, commonly an additional triangular surface mesh is created for an appropriate rendering [CGC*02]. Both meshes are coupled by geometric constraints [CDA00]. The deformation is computed solely on the coarse volumetric mesh, while the triangular mesh is adapted to the deformed state using the geometric constraints, which needs low computation time. Fig. 3.1 illustrates the different representations of objects.

Also, other types of polytopes could be used to discretize the object domain Ω [WBG07, Umm08]. However, tetrahedrons are the most common discretization, and therefore, they are also applied in *DefCol Studio*. Moreover, they are especially suited to the deformation model that is introduced in Chapter 5.

3.2 Deformation model

Depending on the material properties, the internal forces are caused by a displacement of the object O. While in physical applications, the accuracy of the simulation is the main objective, applications in physically-based animation aim at a realistically looking simulation at low computational cost. Thus, a reasonable trade-off between accuracy and computation time has to be found.

For its simplicity and popularity in animation, the mass-spring-model is briefly described in order to illustrate a basic deformation model. Therefore, $\mathbf{x}_i$ denotes the current position of a vertex of the tetrahedral mesh, while $\mathbf{x}_e := \mathbf{x}_j - \mathbf{x}_i$ denotes an edge from $\mathbf{x}_i$ to $\mathbf{x}_j$. $\mathbf{x}_i^0$ and $\mathbf{x}_e^0$ denote the initial positions. $\mathbf{f}_i$ denotes the force acting at vertex $\mathbf{x}_i$, and $\mathbf{f}_e$ denotes the force caused by a deformed edge $\mathbf{x}_e$. All edges of the tetrahedral mesh are considered as springs connecting their incident vertices.

A spring is characterized by the *spring constant* D, which connects the elongation s and the repelling force $f(s)$ acting on a spring according to *Hooke's law*:

$$f(s) = -D \cdot s. \tag{3.1}$$

Note that the deflecting force $f_d(s)$ naturally is the inverse of the repelling force, thus $f_d(s) = D \cdot s$. Using the notation of the tetrahedral mesh, the repelling force becomes

$$\mathbf{f}_e = -D_e(\|\mathbf{x}_e\| - \|\mathbf{x}_e^0\|)\frac{\mathbf{x}_e}{\|\mathbf{x}_e\|} \tag{3.2}$$

This force is symmetrically distributed to the incident vertices $\mathbf{x}_i, \mathbf{x}_j$ as $\mathbf{f}_i -= \mathbf{f}_e, \mathbf{f}_j += \mathbf{f}_e$. Summing up over all incident edges of $\mathbf{x}_i$ results in the total force $\mathbf{f}_i$ which is then used to evolve the movement of $\mathbf{x}_i$ in time (Sec. 3.3).

As similar computations are needed in Sec. 5.1.5, the deformation energy of a deformed spring is introduced. For a constant force $\mathbf{F}$ which acts along a way $\mathbf{ds}$, the work is given as $\mathbf{F} \cdot \mathbf{ds}$. For a spring, $\mathbf{F}$ and $\mathbf{ds}$ have the same direction, which allows to use the scalars $f(s)$, $f_d(s)$ and ds instead of vectors. As the deflecting force depends on the elongation s, the work that has to be done for an elongation of s has to be computed by the integral

$$\int_0^s f_d(s')ds' = \int_0^s D \cdot s' ds' = \frac{1}{2}D \cdot s^2 = \frac{1}{2}f_d(s) \cdot s, \tag{3.3}$$

which introduces a factor of $\frac{1}{2}$. Note that this equation for the deformation energy holds for each situation where the force depends linearly on the elongation.

The mass-spring-method can also be interpreted as a penalty method which penalizes elongations of the edges of a tetrahedral mesh. This can be generalized

to penalty methods for the triangles and tetrahedrons that penalize the change of the triangle area or the volume change of a tetrahedron [THMG04].

DefCol Studio provides the implementation of different deformation models. Among these are penalty methods [THMG04], geometrically motivated methods [MHTG05] and physically-based methods like the linear and non-linear Finite Element Method [MG04].

3.2.1 Momentum conservation

The spring forces have the property that the sum over all forces is zero, as already the forces caused by each spring sum up to zero. This is an essential property for any deformation model, as it states that the linear global movement of the object is not affected.

The center of mass $\mathbf{x}_{cm}$ of an object O is defined as

$$\mathbf{x}_{cm} := \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i} = \frac{1}{M} \sum_i m_i \mathbf{x}_i, \tag{3.4}$$

where m_i denotes the mass of $\mathbf{x}_i$ and M denotes the total mass of O. Thus, its velocity $\mathbf{v}_{cm}$ is given as

$$\mathbf{v}_{cm} = \dot{\mathbf{x}}_{cm} = \frac{1}{M} \sum_i m_i \dot{\mathbf{x}}_i = \frac{1}{M} \sum_i m_i \mathbf{v}_i, \tag{3.5}$$

where $\dot{\mathbf{x}}_{cm}$ denotes the time derivative, and the acceleration can be computed as

$$\dot{\mathbf{v}}_{cm} = \frac{1}{M} \sum_i m_i \dot{\mathbf{v}}_i = \frac{1}{M} \sum_i \mathbf{f}_i = 0. \tag{3.6}$$

Thus, the velocity of the center of mass is not changed and the *linear momentum* $\mathbf{p}_{cm} := M\mathbf{v}_{cm}$ is conserved. However, each individual linear momentum $\mathbf{p}_i := m_i \mathbf{v}_i$ is generally not conserved.

Moreover, the *angular momentum*

$$\mathbf{L} := \sum_i \mathbf{x}_{i,rel} \times \mathbf{p}_i = \sum_i \mathbf{x}_{i,rel} \times (m_i \mathbf{v}_i), \tag{3.7}$$

where $\mathbf{x}_{i,rel} := \mathbf{x}_i - \mathbf{x}_{cm}$ denotes the position of $\mathbf{x}_i$ relative to the center of mass, is also not affected by the spring forces. Its time derivative, the *torque* $\mathbf{T}$, is given as

$$\mathbf{T} := \dot{\mathbf{L}} = \sum_i \mathbf{x}_{i,rel} \times \mathbf{f}_i. \tag{3.8}$$

Thus, if the torque is zero, the angular momentum is conserved. For each single spring $\mathbf{x}_e$, the corresponding torque $\mathbf{T}_e := \mathbf{x}_{i,rel} \times \mathbf{f}_i + \mathbf{x}_{j,rel} \times \mathbf{f}_j = \mathbf{f}_e \times \mathbf{x}_e = 0$ equals zero, as $\mathbf{f}_e$ and $\mathbf{x}_e$ have the same direction. Hence, the total torque also vanishes.

These two conditions are essential for any deformation model. Otherwise, the simulated object would get an additional rotation or an additional linear movement in case of deformation which obviously is not correct. Commonly, forces that cause such an incorrect movement are called *ghost forces*.

3.3 Time integration

After the internal forces have been determined, Newton's equation of motion has to be integrated for the movement of the object O. The position and velocity are only evaluated at the mass points. For simple deformation models and given initial conditions, it is possible to find an analytic solution of the equation of motion. However, this is not possible in general. Especially the presence of external forces caused by collisions or user interaction prevents an analytical solution. Thus, the equation of motion has to be integrated numerically.

With $\mathbf{X} := (\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T)^T$ being a column vector that summarizes the positions of all mass points, $\mathbf{F} := (\mathbf{f}_1^T, \ldots, \mathbf{f}_n^T)^T$ being the force vector, $\mathbf{V} = (\mathbf{v}_1^T, \ldots, \mathbf{v}_n^T)^T$ summarizing the velocities and $\mathbf{M} \in \mathbb{R}^{n \times n}$ being a matrix that represents the mass distribution of the object, Newton's second law can be written as

$$\mathbf{M}\ddot{\mathbf{X}} = \mathbf{F}, \tag{3.9}$$

where $\mathbf{F} = \mathbf{F}(\mathbf{X}, \mathbf{V}) = \mathbf{F}^{ext}(\mathbf{X}, \mathbf{V}) + \mathbf{F}^{int}(\mathbf{X}, \mathbf{V})$ is composed of external and internal forces. Restricting the mass to the mass points, which is referred to as *mass lumping*, $\mathbf{M}$ becomes a diagonal matrix and the equation can be separated for each single mass point:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{f}_i. \tag{3.10}$$

Eq. (3.10) is an ordinary differential equation of second order. Using the velocity $\mathbf{v}_i$ of $\mathbf{x}_i$, this can be separated into a system of two ordinary differential equations of first order.

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= m_i \mathbf{f}_i \end{aligned} \tag{3.11}$$

For h being a small time interval called *time step*, a numerical integration scheme evaluates the functions $\mathbf{x}_i(t), \mathbf{v}_i(t)$ at discrete time values $0, h, 2h$ and so on. As a simple approach, (3.11) can be discretized using a differential quotient from the right:

$$\frac{\mathbf{x}_i(t+h) - \mathbf{x}_i(t)}{h} \approx \mathbf{v}_i(t) \tag{3.12}$$

$$\frac{\mathbf{v}_i(t+h) - \mathbf{v}_i(t)}{h} \approx \frac{1}{m_i}\mathbf{f}_i(t). \tag{3.13}$$

Solving for $\mathbf{x}_i(t+h)$ and $\mathbf{v}_i(t+h)$ results in the *explicit Euler integration scheme*:

$$\begin{aligned} \mathbf{x}_{i,t+h} &= \mathbf{x}_{i,t} + h\,\mathbf{v}_{i,t} \\ \mathbf{v}_{i,t+h} &= \mathbf{v}_{i,t} + h\frac{1}{m_i}\mathbf{f}_{i,t}, \end{aligned} \tag{3.14}$$

where a lower index $\mathbf{x}_{i,t}$ is used to indicate that these values are an approximation of the correct value $\mathbf{x}_i(t)$. Using a differential quotient from the left in (3.12), i. e. using $\mathbf{v}_i(t+h)$ instead of $\mathbf{v}_i(t)$, results in the *Euler-Cromer-scheme*:

$$\begin{aligned} \mathbf{v}_{i,t+h} &= \mathbf{v}_{i,t} + h\frac{1}{m_i}\mathbf{f}_{i,t} \\ \mathbf{x}_{i,t+h} &= \mathbf{x}_{i,t} + h\,\mathbf{v}_{i,t+h}. \end{aligned} \tag{3.15}$$

As a third example, using also a differential quotient from the left in (3.13), i. e. substituting $\mathbf{f}_i(t)$ by $\mathbf{f}_i(t+h)$, leads to the implicit Euler integration scheme:

$$\begin{aligned} \mathbf{x}_{i,t+h} &= \mathbf{x}_{i,t} + h\,\mathbf{v}_{i,t+h} \\ \mathbf{v}_{i,t+h} &= \mathbf{v}_{i,t} + h\frac{1}{m_i}\mathbf{f}_{i,t+h}. \end{aligned} \tag{3.16}$$

While in (3.14) and (3.15), $\mathbf{v}_{i,t+h}$ and $\mathbf{x}_{i,t+h}$ can be computed directly, they occur both on the left hand and on the right hand side of (3.16), as $\mathbf{f}_{i,t+h}$ depends on $\mathbf{x}_{i,t+h}$ and $\mathbf{v}_{i,t+h}$. Thus, to obtain $\mathbf{v}_{i,t+h}$ and $\mathbf{x}_{i,t+h}$ in (3.16), a system of linear equations has to be solved.

Further integration schemes can be derived from (3.11) by e. g. taking the central differential quotient $\frac{\mathbf{v}_i(t+\frac{h}{2}) - \mathbf{v}_i(t-\frac{h}{2})}{h}$ which leads to the leap frog scheme. The Runge-Kutta-scheme of order j computes several sampling values of the form $\mathbf{x}_{i,t+\delta_k}$ and $\mathbf{v}_{i,t+\delta_k}$ for $0 < \delta_k < h$ $(k = 1, \ldots, j)$ and combines these sample values in order to obtain $\mathbf{x}_{i,t+h}$ and $\mathbf{v}_{i,t+h}$. Moreover, predictor-corrector-schemes compute predicted values $\mathbf{x}'_{i,t+h}$ and $\mathbf{v}'_{i,t+h}$ and employ these values itself to obtain a better estimation for $\mathbf{x}_{i,t+h}$ and $\mathbf{v}_{i,t+h}$.

So far, the introduced integration schemes use only the data at time t to compute the future step $t+h$. Therefore, these schemes are called *single-step integration schemes*. So called *multi-step integration schemes* employ several past time steps $t, t-h, \ldots$. An example is the Verlet integration, which can be obtained by directly discretizing (3.10):

$$\begin{aligned}\ddot{\mathbf{x}}_i(t) &= \frac{\dot{\mathbf{x}}_i(t+h) - \dot{\mathbf{x}}_i(t)}{h} \\ &= \frac{1}{h}\left(\frac{\mathbf{x}_i(t+h) - \mathbf{x}_i(t)}{h} - \frac{\mathbf{x}_i(t) - \mathbf{x}_i(t-h)}{h}\right) \\ &= \frac{\mathbf{x}_i(t+h) - 2\mathbf{x}_i(t) + \mathbf{x}_i(t-h)}{h^2}.\end{aligned} \tag{3.17}$$

Inserting (3.17) into (3.10) leads to

$$\mathbf{x}_{i,t+h} = 2\mathbf{x}_{i,t} - \mathbf{x}_{i,t-h} + \frac{h^2}{m_i}\mathbf{f}_{i,t}. \tag{3.18}$$

If the velocity is needed, e. g. for collision handling or damping (Sec. 8.2), it can be interpolated as

$$\mathbf{v}_t = \frac{\mathbf{x}_t - \mathbf{x}_{t-h}}{h} \text{ or } \mathbf{v}_t = \frac{\mathbf{x}_{t+h} - \mathbf{x}_{t-h}}{2h} \tag{3.19}$$

DefCol Studio provides the implementation of several integration methods like Euler, Euler-Cromer, Verlet, Beeman, and different Runge-Kutta-schemes.

The choice of the integration scheme depends on the underlying task. For example, the explicit Euler scheme is not the best choice for approaches that employ penalty forces, as the forces do not influence $\mathbf{x}_{i,t+h}$, but $\mathbf{x}_{i,t+2h}$. E. g., the Euler-Cromer-scheme would be a better choice for such situations. Implicit schemes have proven to be useful in cloth simulation [BW98], while predictor-corrector-schemes are applied in fluid dynamics [IAGT10, IABT11, IBAT11, SP09] and boundary handling [BTT09].

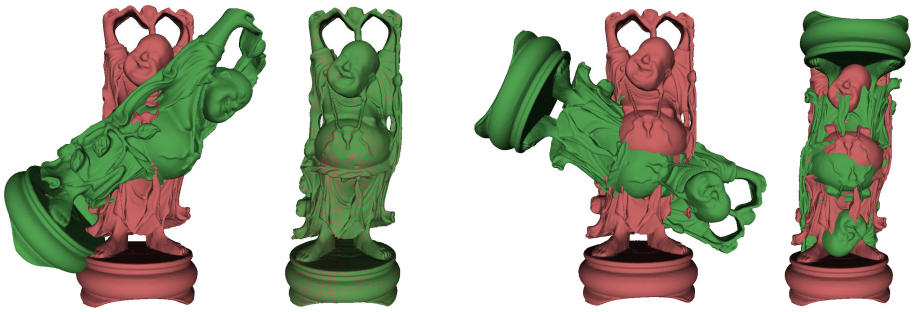
In the context of deformable objects in Computer Graphics, physical realism and computation time have to be balanced. While a fourth-order Runge-Kutta scheme allows for a larger time step than Euler-Cromer or Verlet and for a better approximation of the movement, it needs four force computations per time step. This is the most costly part to compute, as it e. g. requires the evaluation of the deformation model and contact handling forces. Therefore, it would be preferred in applications where accuracy is the main focus. However, it seems that for example the Verlet scheme is the better choice for interactive applications [THMG04].

4

Object acquisition

Realistic representations of real objects significantly enhance the quality of virtual environments. Therefore, the reconstruction of 3D representations is an active area of research. Generally, the first step is to generate a point cloud describing an object [BR02, VBS09]. This can be directly employed in several tasks [GP07], or it can be transformed into appropriate structures that are required [Cur99, BR02, KAHD10].

Modern 3D scanning techniques allow to capture high detail surface scans within short time [Bla04]. However, they have a limited view volume and cannot scan the whole object at once. Thus, in order to obtain a complete point cloud of an object, several scans of different perspectives are needed. Either, several calibrated scanners can be used, or different scans are aligned afterwards by appropriate algorithms. The Iterative Closest Point algorithm (ICP) [BM92, CM91, Zha94] finds an optimal alignment for two scans, if a proper initial configuration is given. However, with badly aligned initial positions, it gets stuck in local minima (Fig. 4.1).



(a) (b) (c) (d)

Figure 4.1: Illustration of the ICP algorithm using two rotated models of the Stanford Buddha. (a,b) For an appropriate initial configuration, the ICP algorithm computes the optimal alignment. (c,d) The ICP is not guaranteed to work and gets stuck in local minima for bad initial configurations.

For two point clouds $\mathbf{X}_r$ and $\mathbf{X}_m$, which are frequently referred to as ref-

erence and model, the registration problem is formulated as the minimization problem

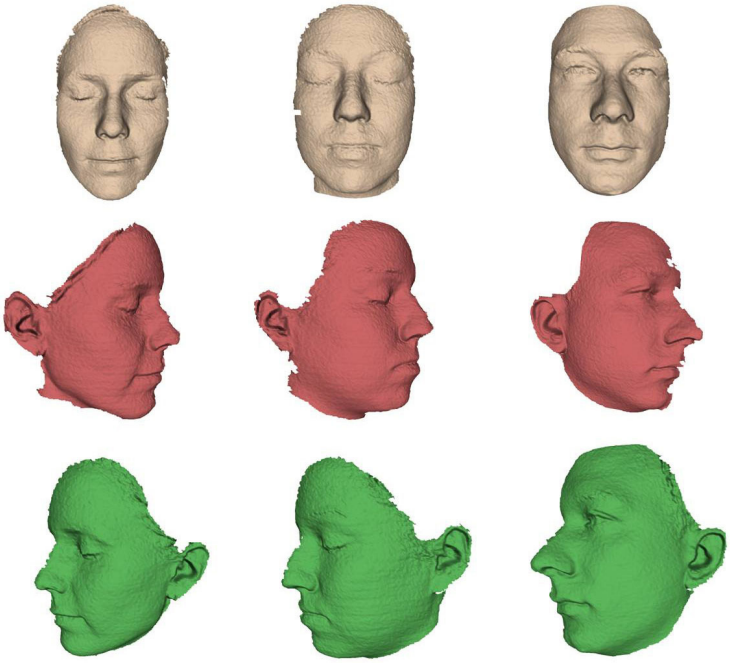
$$(\mathbf{R}, \mathbf{t}) = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\mathbf{x}_{r,i} - \mathbf{R}\mathbf{x}_{m,i} - \mathbf{t}\right)^2, \tag{4.1}$$

where $\mathbf{R}$ is a rotation and $\mathbf{t}$ is a translation. Note that the general registration problem also contains a scaling factor s. However, as the 3D scans have the same scale, this is not needed here. In this formulation, it is assumed that vertices with the same index correspond to each other. Then, an optimal solution for (4.1) can be found analytically [FH86, Hor87, LEF95]. Therefore, the task to establish an appropriate transformation is equivalent to finding appropriate correspondences between the reference and the model, as the transformation can be computed based on these correspondences.

The ICP algorithm locates correspondences $\mathbf{x}_{m,i} \mapsto \mathbf{x}_{r,c(i)}$ locally, e. g. by a nearest neighbor search [BM92] or point-to-plane-correspondences [CM91], and refines the alignment iteratively. Additional criteria like the consistency of normals of corresponding vertices, excluding correspondences with too large absolute distances, or using the best $p\%$ of the found correspondences significantly enhance the convergence of the ICP algorithm [Pul99, RL01, MGPG04]. For partially overlapping point clouds, the registration task becomes more involved. Then, e. g. the border points of objects should not be taken into account, as there generally are lots of wrongly located correspondences to them. Despite all the improvements, the ICP algorithm is still a local method and needs a proper input configuration.

Therefore, global registration algorithms are designed to return a coarsely aligning transformation that can be used as an input for ICP. In order to establish correspondences between the reference and the model without any knowledge about the relative position, they generally employ some kind of surface features like geometry or color, and compare these features over the objects to detect possibly matching candidates.

In this chapter, a novel feature-based algorithm for the global registration of point clouds is introduced. The approach computes surface-approximating polynomials and an extended set of rotationally invariant features for surface points (Sec. 4.2). Further, polynomials with similar features are transformed into a common coordinate system in order to compare their coefficients (Sec. 4.2.2). This results in more discriminating information about different shapes than relying on features only. Further, it allows to optimize the transformation using a local optimization scheme such as Newton iteration. Moreover, a modified distance metric is used to account for the fact that feature values and polynomial coefficients can have different orders of magnitude (Sec. 4.2.5). It is shown that the approach can be used in the simultaneous registration of different objects, each consisting of several partial scans (Sec. 4.3.2). Finally, it is illustrated that the approach yields reasonable global registration results which are almost optimal and require only marginal refinements by the ICP algorithm (Sec. 4.4).



(a)

(b)

Figure 4.2: This example illustrates the application of the transformed polynomials approach in simultaneous multi-scan registration. Three faces are registered simultaneously. Although, e. g., the left parts are very similar to each other and have large overlapping regions, the approach assigns them correctly to the corresponding front scan. (a) Nine arbitrarily oriented scans of different faces. The scans are not presorted in any way. (b) The algorithm computes a proper initial transformation. The result is obtained without local optimization.

4.1 Overview

The proposed registration approach belongs to the class of feature-based voting methods, where the basic idea is to compute local descriptors on surfaces which are invariant under rigid transformations, to compute candidate transformations for points with similar features on various objects, and to perform a voting scheme to locate the transformation with the largest number of votes.

Alg. 1 gives an overview of the global registration algorithm. For two partially overlapping objects A and B, it first computes surface-approximating polynomials (Sec. 4.2.1) and a set of rotationally invariant descriptors based on these polynomials (Sec. 4.2.2, 4.2.3). Then, it locates possible correspondences by comparing the invariant features. For a given point $\mathbf{y} \in B$, a point $\mathbf{x} \in A$ is considered to be a candidate if the descriptor values of the corresponding polynomials p_x and p_y differ less than some threshold ε. Then, a rotation $\mathbf{R}_{xy}$ is extracted that transforms p_x into the local coordinate system of p_y (Sec. 4.2.4). This allows to compare p_y with the transformed polynomial p_x^R. As different polynomials lead to similar descriptor values, the transformed polynomials return more discriminating information about the local neighborhood, which improves the correspondence search.

Let $\mathbf{x}$ be the candidate with the smallest distance between p_x^R and p_y with respect to the distance measure presented in Sec. 4.2.5. Then, $\mathbf{R}_{xy}$ is extended in a straightforward way to a transformation that aligns $\mathbf{x}$ and $\mathbf{y}$ and serves as a candidate for the initial alignment of A and B. Similar to the Hough transform, the corresponding transformation gets a vote, and the transformation with the largest number of votes is chosen as the initial alignment. As shown in Sec. 4.4, using transformed polynomials, it is sufficient to insert only one vote per point, representing the candidate with the best-matching transformed polynomial, to get a global registration result. Indeed, this has some benefits in the simultaneous registration of partial scans of different similarly shaped objects.

Algorithm 1: Transformed Polynomials

1. Compute a surface-approximating polynomial for each surface point (Sec. 4.2.1);
2. Compute rotationally invariant features and find candidate pairs with similar features (Sec. 4.2.3);
3. For each candidate pair, transform the polynomials into a common coordinate system to compare the coefficients (Sec. 4.2.4);
4. For matching polynomials, insert a vote into a subdivision scheme for the transformation space, and choose the transformation with the largest number of votes (Sec. 4.3);

Feature based approaches are widely used in rigid registration, symmetry detection and shape retrieval, e. g. [GMGP05, LG05, MGP06, GCO06, PMW*08, AMCO08]. The approaches employ differential features [MGP06, PMW*08],

integral features [GMGP05, AMCO08] or polynomials to estimate surface descriptors [LG05, GCO06, PMW*08]. To the best of the author's knowledge, none of the approaches uses the transformed polynomials introduced in Sec. 4.2 or the invariants introduced in Sec. 4.2.3.

For an overview of the approach, Fig. 4.2 shows the processing of nine partial scans. The three resulting faces are registered simultaneously, and the partial scans are assigned and registered correctly. Although there is significant overlap between the three left scans and among the three right scans, they are assigned to their corresponding front view of the face.

4.2 Transformed polynomials

In this section, the details of the transformed polynomials are described which are the essential step in the global registration approach. First, the employed surface-approximating polynomials are introduced (Sec. 4.2.1), before the transformation of polynomials is briefly reviewed (Sec. 4.2.2). Then, the invariant descriptors are given (Sec. 4.2.3), and it is shown how the rotation between polynomials in different coordinate systems is obtained (Sec. 4.2.4). Finally, a distance measure is presented that accounts for the fact that the coefficients and invariants of compared polynomials can have different orders of magnitude (Sec. 4.2.5).

4.2.1 Surface-approximating polynomials

To establish correspondences between partial scans of an object, the surface is first described locally using a surface-approximating polynomial at each point. In order to be robust against noise, the approach employs the Moving Least Squares method [Lev98, Lev04]. However, the algorithm does not depend on this choice and also works for other types like osculating jets [CP03], which are used in [PMW*08].

The Moving Least Squares approach first computes a best-fit plane for each point $\mathbf{x}$ such that the squared orthogonal distance of all points in a neighborhood is minimized. Then, the normal $\mathbf{n}_x$ of this best-fit plane is taken as the surface normal for $\mathbf{x}$ and is extended to a local orthonormal coordinate system $(\mathbf{l}_x^1, \mathbf{l}_x^2, \mathbf{n}_x)$. Within this coordinate system, a surface-approximating polynomial $p_x : \mathbb{R}^2 \to \mathbb{R}$ is calculated based on the neighborhood of $\mathbf{x}$. For the plane as well as for the polynomial, the influence of points is weighted depending on their distance to $\mathbf{x}$. The weight is controlled by the feature size h, which hereby implicitly defines the neighborhood of a point $\mathbf{x}$. Note that the Moving Least Squares approach is not used for smoothing the input surface, as this would lead to worse registration results [AMCO08].

Polynomials of degree 3 are well suited for the registration approach, as they return good approximation results, while polynomials with higher degree tend to oscillate and result in a worse approximation [ABCO*01]. In general, a polynomial has the form

$$\begin{aligned} p(u,v) = \; & a_{30}u^3 + a_{20}u^2 + a_{10}u + a_{21}u^2v + a_{11}uv \\ & + a_{12}uv^2 + a_{01}v + a_{02}v^2 + a_{03}v^3 + a_{00}. \end{aligned} \tag{4.2}$$

4.2.2 Polynomial transformation

In this section, the transformation of a polynomial $p : \mathbb{R}^2 \to \mathbb{R}$ into a rotated coordinate system is reviewed, which shows how the coefficients have to be transformed. First, this transformation is needed to determine polynomial invariants under rotation. Second, this is a necessary step for comparing different polynomials, as each surface-approximating polynomial p is defined in a local coordinate system. Thus, they have to be transformed into a common coordinate system in order to be comparable.

Let p be defined in the standard basis $\mathbf{B} = (\mathbf{e}_1, \mathbf{e}_2)$ and let $\mathbf{B}^R = (\mathbf{r}_1, \mathbf{r}_2)$ define a coordinate system which is rotated by $\mathbf{R}$. Then, the basis transformation from $\mathbf{B}^R$ to $\mathbf{B}$ is also given by $\mathbf{R}$. Now, the rotated polynomial p^R is searched that is defined in $\mathbf{B}^R$ and equals p.

For a point given as (u, v) in $\mathbf{B}$ and (u^R, v^R) in $\mathbf{B}^R$, p^R has to fulfill $p^R(u^R, v^R) = p(u, v)$. Inserting the basis transformation leads to $p(u, v) = p(\mathbf{R}(u^R, v^R))$, which can be re-ordered in terms of u^R and v^R to get the coefficients of p^R. In order to compare two given polynomials p in $\mathbf{B}$ and q^R in $\mathbf{B}^R$, p has to be transformed to p^R or q^R to q. Then, the coefficients can be compared.

4.2.3 Invariants

The invariants of the global registration approach are partially based on differential properties of the polynomials. Differential invariants are also used in [PMW*08] and [MGP06], for example. Therefore, the relationship between the derivatives of rotated polynomials is discussed first. Further, the approach combines both differential and integral invariants of a polynomial, which are developed in this section.

For the differential invariants of a polynomial $p(u, v)$, the derivative at the point $(u, v) = (0, 0)$ is considered. The partial derivatives are given as

$$\partial_u p(0,0) = a_{10}, \quad \partial_v p(0,0) = a_{01}, \quad \partial_{uv} p(0,0) = a_{11} \tag{4.3}$$

$$\partial_{uu} p(0,0) = 2a_{20}, \quad \partial_{vv} p(0,0) = 2a_{02}, \quad \partial_{uuv} p(0,0) = 2a_{21} \tag{4.4}$$

$$\partial_{uuu} p(0,0) = 6a_{30}, \quad \partial_{vvv} p(0,0) = 6a_{03}, \quad \partial_{uvv} p(0,0) = 2a_{12} \tag{4.5}$$

The first two invariants are connected to the gradient of p and to the Hessian matrix. For a pair of transformed polynomials p and p^R, the chain rule is used to see that the gradient of p^R is the rotated gradient of p. Hence, its length remains constant.

Further, the second partial derivatives are summarized in the Hessian matrix. Again, the chain rule leads to a relation between the Hessian matrices $\mathbf{H}$ of p and $\mathbf{H}^R$ of p^R, namely $\mathbf{H} = \mathbf{R}\mathbf{H}^R\mathbf{R}^T$. Hence, its determinant is constant.

In the following, two new invariants are introduced that are related to the third derivative and to an integral, respectively. Integral invariants are also used in [GMGP05, HFG*06, PWY*07], for example.

To obtain an integral descriptor that is rotationally invariant, the integration area has to be invariant under rotation. This is fulfilled by a disc with an arbitrary radius d. Then, the integration of the polynomial is easier using polar coordinates $u = r\cos(\varphi), v = r\sin(\varphi)$. The polynomial $p(u, v)$ corresponds to

$$\begin{aligned} p(r,\varphi) =& a_{30}r^3\cos^3(\varphi) + a_{20}r^2\cos^2(\varphi) + a_{10}r\cos(\varphi) \\ &+ a_{21}r^3\cos^2(\varphi)\sin(\varphi) + a_{11}r^2\cos(\varphi)\sin(\varphi) \\ &+ a_{12}r^3\cos(\varphi)\sin^2(\varphi) + a_{01}r\sin(\varphi) \\ &+ a_{02}r^2\sin^2(\varphi) + a_{03}r^3\sin^3(\varphi) + a_{00}. \end{aligned} \tag{4.6}$$

This function is integrated for $\varphi \in [0, 2\pi]$. Using the property

$$\int_0^{2\pi} \cos^k(\varphi)\sin^l(\varphi)d\varphi = 0 \tag{4.7}$$

with k or l being odd, it follows that most of the summands in (4.6) vanish as they contain a sin- or cos-term with an odd exponent. Only the purely quadratic terms $a_{20}r^2\cos^2(\varphi)$ and $a_{02}r^2\sin^2(\varphi)$ remain, and $\int_0^{2\pi} p(r,\varphi)d\varphi$ results in $r^2\pi(a_{20} + a_{02})$. Integrating this term for $r \in [0, d]$ with an arbitrary radius d results in $D \cdot (a_{20} + a_{02})$ with some constant D depending on d. It follows that the sum $a_{20} + a_{02}$ remains constant under rotations.

So far, a_{30}, a_{21}, a_{12} and a_{03} are not part of any invariant. However, as they represent the third partial derivatives, one could conclude that there is an invariant containing these quantities. The following expression can be shown to be invariant under rotation:

$$3a_{30}a_{12} + 3a_{03}a_{21} - a_{21}^2 - a_{12}^2 = const \tag{4.8}$$

The invariance can be shown by transforming the coefficients a_{ij} to the corresponding rotated coefficients a_{ij}^R. For $\mathbf{R} = \left(\begin{smallmatrix} r_{00} & r_{01} \\ r_{10} & r_{11} \end{smallmatrix}\right)$, the transformed coefficients are

$$\begin{aligned} a_{30}^R =& a_{21}r_{00}^2r_{01} + a_{12}r_{00}r_{01}^2 + a_{30}r_{00}^3 + a_{03}r_{01}^3 \\ a_{03}^R =& a_{21}r_{10}^2r_{11} + a_{12}r_{10}r_{11}^2 + a_{30}r_{10}^3 + r_{11}^3a_{03} \\ a_{21}^R =& a_{12}r_{10}r_{01}^2 + 3a_{30}r_{00}^2r_{10} + 2a_{21}r_{00}r_{10}r_{01} \\ &+ 2a_{12}r_{00}r_{01}r_{11} + a_{21}r_{00}^2r_{11} + 3r_{11}r_{01}^2a_{03} \\ a_{12}^R =& 3a_{30}r_{00}r_{01}^2 + 2a_{21}r_{00}r_{01}r_{11} + a_{21}r_{01}^2r_{10} \\ &+ a_{12}r_{00}r_{11}^2 + 2a_{12}r_{01}r_{10}r_{11} + 3a_{03}r_{10}r_{11}^2. \end{aligned} \tag{4.9}$$

Inserting (4.9) into (4.8) and using the fact that $\mathbf{R}$ is a rotation, i. e. $r_{00}^2 + r_{01}^2 = 1$ and so on, leads to

$$\begin{aligned} &3a_{30}^R a_{12}^R + 3a_{03}^R a_{21}^R - {a_{21}^R}^2 - {a_{12}^R}^2 \\ &= 3a_{30}a_{12} + 3a_{03}a_{21} - a_{21}^2 - a_{12}^2, \end{aligned} \tag{4.10}$$

which means that (4.8) is invariant under rotations.

In summary, the global registration approach uses a set of four different invariants:

1. $a_{10}^2 + a_{01}^2 = const$, which is the length of the gradient.
2. $4a_{20}a_{02} - a_{11}^2 = const$, which is the determinant of the Hessian matrix.
3. $a_{20} + a_{02} = const$, which corresponds to an integral of the polynomial over a disc.
4. $3a_{30}a_{12} + 3a_{03}a_{21} - a_{21}^2 - a_{12}^2 = const$, which is related to the third derivative.

4.2.4 Polynomial rotation extraction

Having found a candidate pair $\mathbf{x}, \mathbf{y}$ with polynomials p_x, p_y by comparing the invariants described in the previous section, the rotation $\mathbf{R}_{xy}$ between p_x and p_y has to be computed that aligns the polynomials in one coordinate system in order to compare their coefficients. By first aligning the local coordinate systems of $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{R}_{xy}$ is extended to a transformation $\mathbf{T}_{xy}$ aligning both points. The extraction of $\mathbf{R}_{xy}$ is based on the relationship between the Hessian matrices $\mathbf{H}_x$ and $\mathbf{H}_y$ of p_x and p_y, which is explained in this section. It is similar to [MGP06, PMW*08], who use the principal curvatures to align different points.

As $\mathbf{H}_x$ and $\mathbf{H}_y$ are symmetric, they are orthogonally diagonalizable. Hence, there are rotations $\mathbf{Q}_x$ and $\mathbf{Q}_y$ such that

$$\mathbf{D}_x = \mathbf{Q}_x^T \mathbf{H}_x \mathbf{Q}_x \tag{4.11}$$

$$\mathbf{D}_y = \mathbf{Q}_y^T \mathbf{H}_y \mathbf{Q}_y \tag{4.12}$$

have diagonal form. For a pair of transformed polynomials, the diagonal forms have to be equal. If this is not the case, the two points cannot correspond to each other. Otherwise, it holds

$$\mathbf{Q}_x^T \mathbf{H}_x \mathbf{Q}_x = \mathbf{Q}_y^T \mathbf{H}_y \mathbf{Q}_y \tag{4.13}$$

$$\Rightarrow \mathbf{H}_x = \mathbf{Q}_x \mathbf{Q}_y^T \mathbf{H}_y \mathbf{Q}_y \mathbf{Q}_x^T. \tag{4.14}$$

Thus, $\mathbf{R}_{xy} = \mathbf{Q}_x \mathbf{Q}_y^T$ is the basis transformation matrix that transforms p_x into the coordinate system of p_y. Note that (4.14) only holds if the eigenvalues

of $\mathbf{H}_x$ and $\mathbf{H}_y$ have the same order. If this is not the case, they have to be sorted correspondingly in order to get the correct rotation, and the columns of $\mathbf{Q}_x$ and $\mathbf{Q}_y$, which are built by the eigenvectors of $\mathbf{H}_x$ and $\mathbf{H}_y$, have to be rearranged accordingly.

As p_x is transformed into the coordinate frame of p_y, the aligning transformation $\mathbf{R}_{xy}$ can be improved using some optimization scheme like Newton iteration to minimize the distance between the transformed coefficients. This further improves the quality of the surface matching. The employed distance measure is explained in the following section.

4.2.5 Distance measure

The coefficients and invariant features of a polynomial can have different orders of magnitude. Therefore, a modified distance measure is used to account for this fact. While for small coefficients, the squared distance is an appropriate measure, the squared distance of large coefficients would dominate smaller coefficients. E. g., an absolute distance of 1 between coefficients of 99 and 100 seems not too much, but it would overrule a distance of 0.5 between coefficients of 0.5 and 1. Therefore, a relative distance like the quotient would be more appropriate for large coefficients. Obviously, the quotient should approximate 1, so $(1 - \frac{a}{b})^2$ would be a candidate for a distance measure for large coefficients a and b.

However, the quotient is not symmetric, and it prevents the use of spatial subdivision techniques for the feature search, which worsens the runtime. This can be overcome by comparing the logarithms of large coefficients and invariants, which allows to transform the quotient $\frac{a}{b}$ into the difference $\ln(a) - \ln(b)$.

With the Taylor series of $\ln(x)$ around 1,

$$\begin{aligned} \ln(x) &= \ln(1) + \ln'(1)(x-1) + O((x-1)^2) \\ &= x - 1 + O((x-1)^2), \end{aligned} \tag{4.15}$$

it can be seen that $|\ln(a) - \ln(b)| = \left|\ln(\frac{a}{b})\right|$ is a first-order approximation of $\left|1 - \frac{a}{b}\right|$. For similar coefficients, $\frac{a}{b}$ approximates 1, and therefore, $(\ln(a) - \ln(b))^2$ is a sufficient approximation for $(1 - \frac{a}{b})^2$. Hence, $(\ln(a) - \ln(b))^2$ is defined as the distance of large coefficients a and b.

The question is what "large" coefficients are. However, there is a straightforward way to determine where to switch between the different measures. Assuming that it is reasonable to always take the minimum of both possible measures leads to a criterion where the logarithmic measure should be chosen. This is the case if the condition

$$\left|1 - \frac{a}{b}\right| < |a - b| \tag{4.16}$$

is fulfilled, which can be transformed to

$$\begin{aligned} &\left|\frac{b-a}{b}\right| < |a-b| \\ \Leftrightarrow &|b| > 1, \end{aligned} \tag{4.17}$$

and (4.16) holds iff $|b| > 1$. Symmetrically, the same condition can be deduced for a. Hence, if $|a|$ and $|b|$ are greater than 1, the logarithmic difference should be taken.

4.3 Global registration

In this section, the computation of the global registration of two objects A and B is summarized.

For each point $\mathbf{x} \in A$ and $\mathbf{y} \in B$, the surface-approximating polynomials p_x and p_y together with their invariants are determined. For each pair $\mathbf{x} \in A, \mathbf{y} \in B$ with similar features, the polynomial p_x is transformed into the local coordinate system of p_y and the coefficients are compared. Then, for each $\mathbf{x} \in A$, the point $\mathbf{y} \in B$ with the best-matching polynomial is chosen as a possibly corresponding point and the aligning transformation $\mathbf{T}_{xy}$ is computed. Taking only the best-matching point instead of all feature-matching candidates is possible due to the transformed polynomials, which allows a more thorough comparison of the local neighborhood.

Finally, a voting scheme similar to the Hough transform is performed and the transformation $\mathbf{T}_{AB}$ with the largest number of votes is taken as the initial alignment of A and B. Further, for each point pair $(\mathbf{x}, \mathbf{y})$ whose transformation $\mathbf{T}_{xy}$ voted for $\mathbf{T}_{AB}$, $\mathbf{y}$ is stored as a corresponding point to $\mathbf{x}$. As an alternative to $\mathbf{T}_{AB}$, the stored correspondences can also be used to compute an initial alignment using the method presented in [Hor87].

Of course, it is also possible to take more than only the best-matching polynomial, but as shown in Sec. 4.4, using only the of best-matching polynomial is sufficient. Moreover, this has some benefits especially for similarly shaped objects, as it avoids mismatches that influence the global alignment 4.3.2.

4.3.1 Runtime

The theoretical runtime is dominated by the search for the local neighborhood to establish the surface polynomials and by the search for point pairs with similar features. Using *kd*-trees, the algorithm needs $O(n)$ range queries which take $O(\log n + k)$ time if n is the number of object points and k is the number of points returned by the range query. In the worst case, all points have similar features, which results in $k = n$ and a runtime of $O(n^2)$.

Indeed, the largest part of the runtime is spent in the computation of the polynomials and the search for candidate points, while the remaining parts - including the voting scheme - typically take only a few seconds. However, the computation of the polynomials and candidates for two points $\mathbf{x}_1$ and $\mathbf{x}_2$ is

completely independent of each other, which means that the time-consuming steps can be perfectly parallelized.

4.3.2 Multi-scan registration

In this section, the application of the transformed polynomials approach in multi-scan registration is outlined. It is shown that it returns sufficiently discriminating information when registering different objects simultaneously.

For a given set of partial scans $A_1, \ldots, A_k$, all scans are registered pairwise to obtain a set of transformations $\mathbf{T}_{ij}$ and a set of correspondences between all objects. Like [LG05, HFG*06], a connectivity graph is established with the nodes $A_1, \ldots, A_k$ and weighted edges $e(A_i, A_j)$ between A_i and A_j if an aligning transformation $\mathbf{T}_{ij}$ is found. The weight $c(e(A_i, A_j))$ reflects the registration quality from A_i to A_j. In order to control the weight of an edge, two independent registration steps are performed for each pair of objects (A_i, A_j). First, this allows to check if $\mathbf{T}_{ij}\mathbf{T}_{ji}$ approximates the identity, and if not, the corresponding edges are rejected. This is similar to [HFG*06] who check circles for consistency. Second, if $\mathbf{y} \in A_j$ corresponds to $\mathbf{x} \in A_i$, it can be checked if the reverse relation also holds. Therefore, a bijection between the corresponding points of A_i and A_j is obtained. Then, the number of bijective pairs serves as the weight for the edge $e(A_i, A_j)$.

The quality of these edge weights is illustrated in Fig. 4.2, which shows the simultaneous registration of nine partial scans from three different faces. The left parts of the three faces are similar to each other and have more overlap among themselves than to their corresponding front scan, and the same holds for the right parts. Nevertheless, the approach finds about an order of magnitude more correspondences from the left and right scans to their respective front scan than among each other and to the not corresponding front scans (Sec. 4.4). This is due to the transformed polynomials which allow to take only one candidate pair per point. Thus, lots of matches are excluded which would occur over the cheeks, for example, and would lead to higher weights between the left and between the right scans. Therefore, the edge weights obtained by the transformed polynomials approach return information about the connectivity of the objects, although the structures are quite similar and are likely to be mixed up.

Similar to [LG05, HFG*06], the objects (A_i, A_j) are registered in order of their edge weights. Then, if an object A_k has correspondences to A_i and A_j, the corresponding transformations are checked for consistency as in [HFG*06], and the edge weights are summed up. This is done up to some user-defined threshold. In Fig. 4.2, the approach ends up with separated objects.

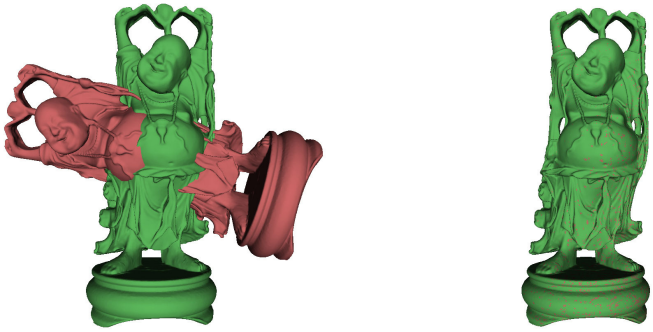
4.4 Results

In this section, the capabilities of the global registration approach are illustrated. All results are obtained using the transformed polynomials approach without any local improvement. The experiments have been performed on an AMD Opteron 8435 with 24 cores at 2.6 GHz and 64 GB memory. The Bunny and the Buddha

models are taken from the Stanford 3D Scanning Repository of the Stanford University Computer Graphics Laboratory.

In the implementation, the feature size for the Moving Least Squares approach (Sec. 4.2.1) is set to about 3% of the bounding box diagonal. Two points $\mathbf{x}, \mathbf{y}$ are considered as candidates if the distance of their polynomial invariants (Sec. 4.2.5) is smaller than 10^{-5}. As the polynomials are transformed afterwards and only the best-matching candidate is taken, this threshold rather affects the runtime than the registration result. The subdivision scheme for the aligning transformation has a cell size of 6° for the rotation angle and of 0.05 for the components of the rotation axis, which is represented as a unit vector. For the translational components, the cell size corresponds to the feature size.

In order to illustrate the quality of the global alignments, the ICP algorithm [BM92, CM91] was applied after the transformed polynomials approach. For an optimal global alignment, ICP should end up with no additional rotation. Therefore, the additional rotation angle obtained by ICP is given to show the quality of the transformed polynomials approach. However, the figures show the global registration results without any local improvement.



(a) (b)

Figure 4.3: The Stanford Buddha consisting of 135634 points. (a) Relative orientation before registration. (b) Transformation found by the global registration approach. 135631 points are assigned correctly by the transformed polynomials approach.

The first experiment in Fig. 4.3 shows an artificial test setup with two models of the Stanford Buddha. The experiment employs a resampled mesh consisting of 135634 points, where one model is rotated by an arbitrary angle. Although there are lots of similar surface parts which lead to similar surface polynomials, the approach locates 135631 correspondences correctly, which is remarkable as exactly one possible correspondence per point is allowed. Using the transformed polynomials, the best-matching polynomial leads to the correct correspondence for most of the points. The runtime for this experiment was 211s. Here, applying ICP results in no additional rotation angle, so the optimal alignment is reached.

All of the following experiments are performed with real data collected by different range scanners.

Fig. 4.4 shows the registration of ten range scans of the Stanford Bunny. A similar experiment has also been performed in [GMGP05]. In contrast to [GMGP05], where overlapping scans are pre-assigned and registered pairwise in single registration steps, no presorting is done in this experiment, and no pair of scans is excluded a priori. Nevertheless, all scans are reliably assigned and registered simultaneously within $583s$ using the transformed polynomials approach. Compared to [GMGP05], where only few correspondences are found between overlapping objects, the approach locates 5376 correspondences on average. Applying ICP, an average additional rotation angle of 0.44° is obtained.

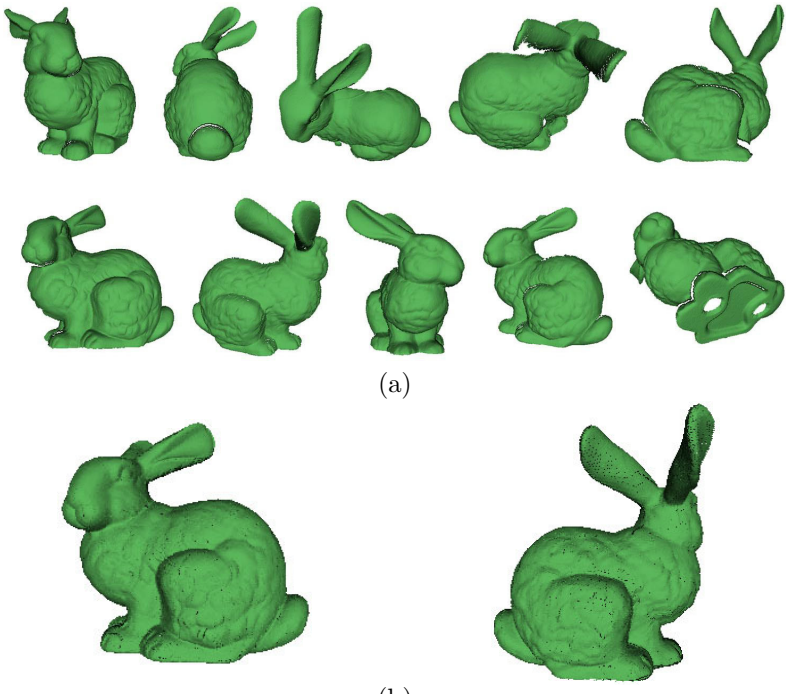


Figure 4.4: (a) 10 range scans of the Stanford Bunny with arbitrary orientations. (b) Point clouds of the registered scans. The result is obtained without any local refinement.

Fig. 4.5 and Fig. 4.6 illustrate that the approach is able to handle noisy data. In Fig. 4.5, random noise is added to the face scans to illustrate the robustness of the approach. Although the noise is in the same order of magnitude as the feature size, the global registration finds a proper initial alignment. The objects in Fig. 4.6, a teddy bear besides a mobile robot, were scanned with a Microsoft® Kinect™-camera and are rather noisy themselves (Fig. 4.6 (b)). Nevertheless, the scans are assigned and registered simultaneously within $1766\,s$. Applying ICP results in an additional rotation angle of 1° on average.

Fig. 4.2 illustrates that the algorithm is also able to assign partial scans of different faces correctly. The face scans are obtained by a face*SCAN* III system

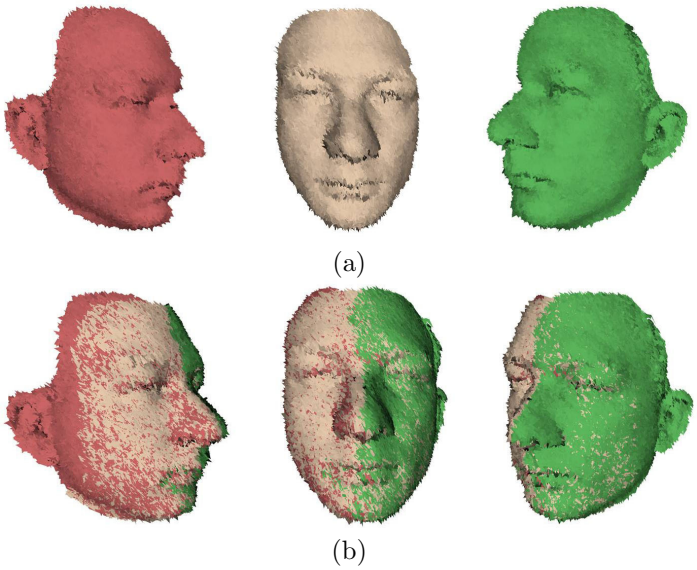


Figure 4.5: (a) Although the scanned data naturally contains noise, random noise is added to a face scan to illustrate the robustness of the transformed polynomials approach. (b) The simultaneous global registration algorithm is able to obtain a proper alignment within 1118 s.

from Breuckmann[1,2] and have an average size of 79800 points. The three resulting faces are registered simultaneously, and the partial scans are assigned and registered correctly. Although there is significant overlap between the three left scans and among the three right scans, they are assigned to their corresponding front view of the face. The approach locates 9500 correspondences on average between correct pairs of scans, while there are typically $200 - 300$ correspondences between false left-left or right-right pairs, which serve as weights for the simultaneous registration approach (Sec. 4.3.2). As in Fig. 4.4, the scans are not presorted in any way and no pairs are excluded a priori. This is a special strength of the transformed polynomials approach. For example, algorithms looking for a largest common point set would be likely to assign a higher weight between two left scans than between the correct left and front scan. Registering nine partial scans simultaneously requires 72 single registration steps, which took 100 s on average. Applying ICP results in an additional rotation angle of $0.3°$ on average.

[1] http://www.breuckmann.com/kunst-kultur/produkte/facescan.html, accessed December 21, 2011

[2] http://www.breuckmann.com/fileadmin/Kundendaten/service/prospekte/pdf/faceScan_de.pdf, accessed December 21, 2011

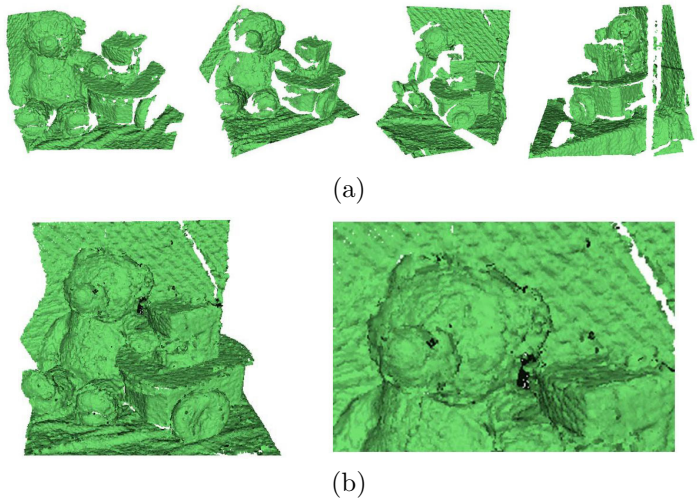


Figure 4.6: (a) Range scans of a teddy bear and a mobile robot. (b) The transformed polynomials approach returns a good global registration, although the scans are quite noisy.

4.5 Conclusion

In this chapter, a novel approach for the global rigid registration of partially overlapping point clouds was introduced. It employs new invariants for the search of possible candidates and the transformed polynomials which lead to an improved correspondence search. The established correspondences can be applied for the simultaneous registration of several similarly shaped objects. The results illustrate the capabilities of the approach for global registration tasks.

The transformed polynomials approach leads to more discriminating information than using surface features only and therefore, it is able to distinguish different objects. However, the approach naturally has some limitations.

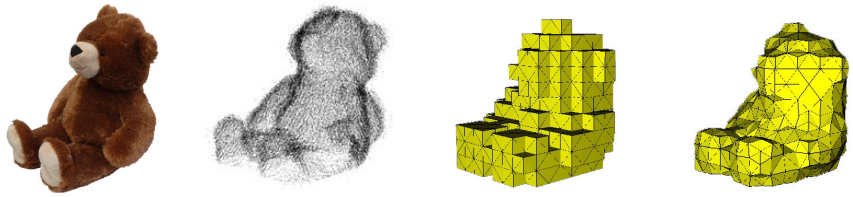
Obviously, the approach runs into problems if it cannot compute distinct features like it is the case for objects with large featureless parts, e. g. planes. For such situations, the approach described in [AMCO08] is more appropriate.

Like all registration approaches, the approach has problems to establish an initial transformation if there is only a small overlap. The multi-scan registration is based on a heuristic, which does not work in every case. However, due to the transformed polynomials, the approach at least returns few false correspondences between different objects, although they have a similar shape. Consequently, a lot of partial scans are registered correctly before the first false match occurs. Therefore, similar to [LG05], the number of non-registered objects decreases, which simplifies the manual separation of the objects, if the heuristic does not completely work.

Further processing of registered scans

Depending on the application, the obtained point cloud is transformed into appropriate structures. In the context of deformable objects, especially a volumetric mesh is needed. However, the usage of real scans and the requirement of a stable simulation impose several requirements onto the mesh generation algorithm. First, scans of real objects might be incomplete. Second, after matching different scans, the overlapping point clouds might not define an unambiguous surface, if the scans contain errors and noise. Third, the deformation model imposes restrictions on the mesh quality, as badly shaped meshes, e. g. containing flat tetrahedrons like slivers, tend to be unstable and are difficult to simulate ([She02], Sec. 5.2.1).

The approach of [SWT06] is suited for all these requirements, as unorientable, non-manifold, and even incomplete input surfaces can be handled. First, the approach computes a signed distance field, where voxels with a negative sign correspond to the pseudo-volume included by the point cloud. In a second step, the spatial domain is divided by an axis-aligned grid, and all cells that do not contain voxels with negative sign are discarded. The remaining cells approximate the object's volume, while the approximation quality directly depends on the grid resolution. The grid cells are divided into five tetrahedrons each, and the mesh is smoothed in a postprocessing step using smoothing filters similar to [DMSB99]. Although the approach is formulated for triangular input surfaces, it equally works for point clouds, as the only condition is that a distance field can be computed. During the smoothing process, the mesh quality can be observed, thus allowing to determine a lower limit of the mesh quality. Fig. 4.7 illustrates the algorithm.



(a) (b) (c) (d)

Figure 4.7: Illustration of the mesh generation algorithm of [SWT06]. (a,b) show the real object with the scanned point cloud. (c,d) illustrate the constructed tetrahedral mesh before and after smoothing.

5 Deformation model

In Chapter 3, the basic steps for the simulation of deformable objects and the simulation environment *DefCol Studio*[1,2] have been introduced. Especially, in Sec. 3.2, the mass-spring-method has been introduced. As this deformation model has several drawbacks, the linear Finite Element Method is used throughout this thesis. As the deformation model is the basis for the simulation as well as for the parameter estimation, it is introduced in this chapter.

While computationally efficient, the mass-spring-method is not physically motivated, and thus, it cannot reflect physical properties correctly. For example, it is hard to represent both isotropic and anisotropic materials [BC00], and the parameter setting for a mass-spring-network is not connected to real material constants of objects that should be represented. Moreover, the spring constants have to be estimated separately for meshes with different topologies [BHS03], which can be illustrated easily by considering two springs of equal length and equal spring constant D. A force F elongates the springs by the amount $s = \frac{F}{D}$. However, connecting both springs in series and applying the same force F leads to an elongation of $2s$, and thus, the spring constant for the composed springs is $D' = \frac{F}{2s} = \frac{D}{2}$. Applying F to a parallel combination of both springs leads to a spring constant $D' = 2D$. Thus, it is almost impossible to predict the necessary new spring constants in case of remeshing.

Therefore, physically-based deformation models like the Finite Element Method are more appropriate for the simulation of real objects. The linear Finite Element Method is based on the constitutive equation of the considered material, and employs the physical parameters that define this equation. For isotropic materials, these are the Young modulus and the Poisson ratio. As the deformation model is derived from the material properties, the parameters do not depend on the tetrahedral mesh. Thus, if the parameters of an object are estimated once, they can be applied to every mesh that approximates the object.

This chapter is organized as follows. First, the physical background concerning linearly elastic objects is introduced in Sec. 5.1. Then, the linear Finite Element Method is introduced in Sec. 5.2. As the linear Finite Element Method

[1] Heidelberger, B.: DefCol Studio - Interactive deformable modeling framework. http://www.beosil.com/projects.html, accessed November 11, 2011

[2] Teschner, M.: DefCol Studio 1.0.0, http://cg.informatik.uni-freiburg.de/software.htm, accessed November 11, 2011

does not handle rotations correctly, it is extended to the co-rotational Finite Element Method [HS04, MG04] in Sec. 5.2.2, which is employed throughout this thesis and in the application in Chapter 9.

Notation

Fig. 5.1 illustrates the notation of this chapter. Let O be an object and $\Omega \subset \mathbb{R}^3$ its spatial domain in the initial state. The current configuration of O can be described by a mapping $\mathbf{\Phi}\colon \Omega \to \mathbb{R}^3, \mathbf{x} = (x, y, z) \in \Omega \mapsto \mathbf{\Phi}(\mathbf{x})$. Usually, $\mathbf{\Phi}$ is decomposed into the embedding $\mathbf{id}$ of Ω into $\mathbb{R}^3$ and the displacement $\mathbf{u}\colon \Omega \to \mathbb{R}^3, \mathbf{x} \mapsto \mathbf{u}(\mathbf{x})$, which is called the *displacement function* or *displacement field*. Then, $\mathbf{\Phi}$ is given as $\mathbf{\Phi}(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x})$ (Fig. 5.1). The components of $\mathbf{u}$ are denoted as (u_x, u_y, u_z). The partial derivatives are denoted as $\partial_x\mathbf{u}, \partial_y\mathbf{u}, \partial_z\mathbf{u}$, while the total derivative is denoted with $\nabla\mathbf{u}$.

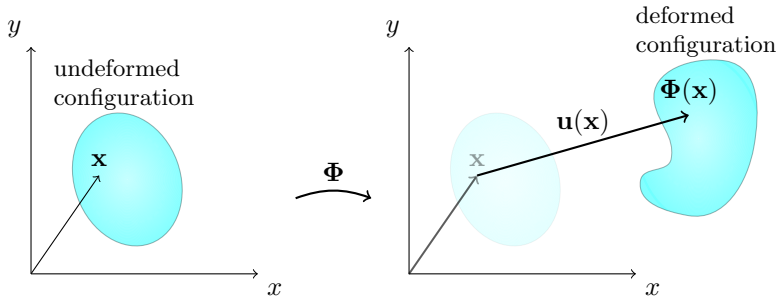


Figure 5.1: Illustration of displacement function $\mathbf{u}(\mathbf{x})$.

5.1 Linear elasticity theory

An elastically deformable object is characterized by the property that it changes its shape in the presence of external forces, while it restores the original shape if the forces are released. Thus, in the case of deformation, there are internal forces acting to restore the resting state. In Sec. 5.2, this will be used to deduce the linear Finite Element Method, which is a physically-based deformation model that is integrated within the framework introduced in Chapter 3. While this section restricts to linear elasticity, further information about elasticity theory can be found in [BB75, GM01, Gre03].

The deformed state of an object is characterized by the central quantities *stress* and *strain*. They describe the forces and the current shape of a deformed object and are introduced in Sec. 5.1.1. The relation between stress and strain is called the *constitutive equation* of the underlying material and defines its elastic behavior. For linearly elastic objects, this relation is linear. If the material is additionally assumed to be isotropic, i. e. it behaves equally in all directions, it depends on two deformation parameters, the Young modulus and the Poisson

ratio. These are illustrated in Sec. 5.1.3, before the stress-strain-relation is deduced in Sec. 5.1.4. In conservative systems, the forces depend only on the current state of an object, and act to a minimum of the potential energy. Thus, for deformable objects, the internal forces can be computed as the derivative of the deformation energy, which is introduced in Sec. 5.1.5.

5.1.1 Stress and strain

In this section, the physical quantities stress and strain are introduced. The stress measures a force per unit area acting on an object, while the strain measures the deformation of the object compared to its original size and shape. It is defined as a quotient $\frac{\text{“}Change\ of\ a\ length\text{”}}{\text{“}Original\ length\text{”}}$. The general description of stress and strain can be reduced to two special cases, which are denoted as *normal strain* and *shear strain* and are shown in Fig. 5.2.

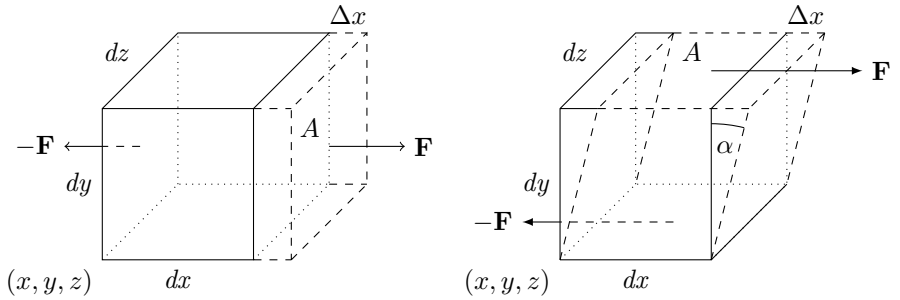


Figure 5.2: Illustration of normal and shear stress and strain. (a) illustrates the normal stress and strain. The force $\mathbf{F}$ acts perpendicular to the surface A and causes an elongation Δx. (b) illustrates the shear stress and strain. The force $\mathbf{F}$ acts parallel to the surface A and causes a deflection Δx.

Normal stress and normal strain. Fig. 5.2 (a) illustrates a force $\mathbf{F}$ acting perpendicular to an area A which is called *normal stress*. As both force and surface normal point in x-direction, the strain is denoted as $\sigma_{xx} = \|\mathbf{F}\|/A$ or σ_x, respectively. This causes an elongation Δx, which defines the normal strain $\varepsilon_{xx} = \varepsilon_x = \frac{\Delta x}{dx}$. Using the displacement function $\mathbf{u}$, Δx can be written as

$$\Delta x = \underbrace{u_x(x+dx,y,z)}_{\approx u_x(x,y,z)+\partial_x u_x(x,y,z)\cdot dx} - u_x(x,y,z) \approx \partial_x u_x(x,y,z)\cdot dx, \tag{5.1}$$

and the normal strain is given as

$$\varepsilon_x = \partial_x u_x. \tag{5.2}$$

Shear stress and shear strain. Fig. 5.2 (b) illustrates a force $\mathbf{F}$ acting parallel to the surface A which is called *shear stress*. The force acts in x-direction, while the surface normal points in y-direction. Therefore, the shear stress is denoted as $\sigma_{xy} = \|\mathbf{F}\|/A$ or τ_{xy}, respectively. This causes a deflection Δx, which defines the shear strain $\gamma_{xy} = \frac{\Delta x}{dy} = \tan(\alpha)$. For small deformations, $\tan(\alpha) \approx \alpha$ and the shear strain becomes $\gamma_{xy} \approx \alpha$.

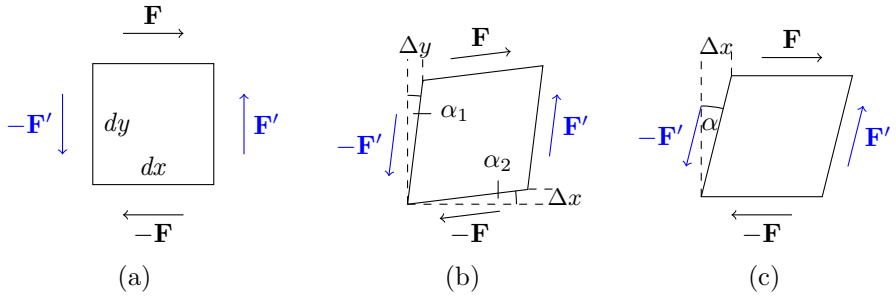


Figure 5.3: Illustration of forces that cause a torque-free shear strain. (a) depicts the forces acting in order to obtain shear strain. (b) illustrates the deformation obtained by the forces in (a). It contains no additional rotation like the state shown in (c), which is obtained from (b) by a clockwise rotation by the angle α_2.

However, the forces $\mathbf{F}$ and $-\mathbf{F}$ in Fig. 5.2 (b) cause a torque. Thus, in order to achieve a torque-free shear deformation, there are additional forces $\mathbf{F}'$ and $-\mathbf{F}'$ as shown in Fig. 5.3 (a). These cause both a deflection Δx and a deflection Δy, leading to the deformation shown in Fig. 5.3 (b). Followed by a clockwise rotation by the angle α_2, the situation in Fig. 5.3 (c), corresponding to Fig. 5.2 (b), is reached. The shear strain $\gamma_{xy} \approx \alpha$ in Fig. 5.3 (c) corresponds to $\alpha_1 + \alpha_2$ in Fig. 5.3 (b). Thus, it is reasonable to redefine the shear strain as $\gamma_{xy} = \frac{\Delta x}{dy} + \frac{\Delta y}{dx} \approx \alpha_1 + \alpha_2$. Similar to the normal strain, the deflection Δx can be expressed as

$$\Delta x = \underbrace{u_x(x, y + dy, z)}_{\approx u_x(x,y,z) + \partial_y u_x(x,y,z) \cdot dy} - u_x(x, y, z) = \partial_y u_x \cdot dy, \tag{5.3}$$

and the shear strain becomes

$$\gamma_{xy} = \partial_y u_x + \partial_x u_y. \tag{5.4}$$

Thus, the stress and strain of an object are defined by nine scalars each. This leads to the following definition.

Definition 1 (Stress)**.** *The stress $\boldsymbol{\sigma} \in \mathbb{R}^{3\times3}$ is defined as a second-order tensor and has the measurement unit force per unit area. Its component $\sigma_{ij}, i, j \in$*

$\{x, y, z\}$ denotes the stress caused by a force in direction i acting on an imaginary area A_j with normal direction j. The total force in direction i then is given by $F_i = \sum_j \sigma_{ij} A_j$.

Fig. 5.4 illustrates the components of the stress tensor.

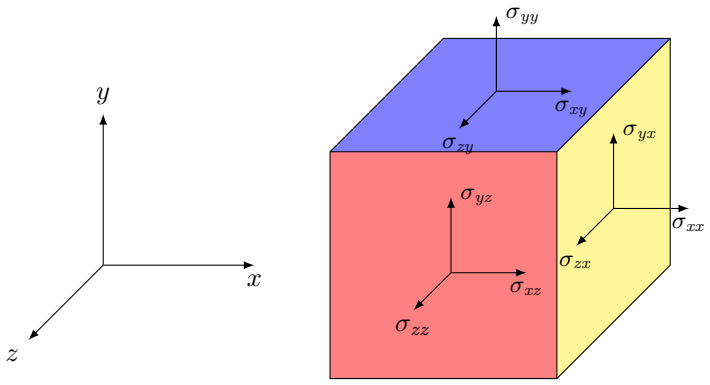


Figure 5.4: Illustration of different stress components.

Similarly, the strain could be defined as a second-order tensor $\boldsymbol{\varepsilon}$ containing the normal strains at the diagonal and the shear strains at the off-diagonal positions. However, this definition would not be entirely consistent, for which reason the strain tensor is defined slightly different based on the strain for an arbitrary line segment. This is shown in the following.

Strain for an arbitrary line segment. The normal and shear strain are aligned to elongations or deflections associated with the coordinate axes. For an arbitrary segment $\mathbf{x}_1 - \mathbf{x}_2$ with length $l = \|\mathbf{x}_1 - \mathbf{x}_2\|$ and deformed length $l + \Delta l$, the strain is defined as $\varepsilon_{x_1 x_2} = \frac{\Delta l}{l}$. The length of the deformed segment is given by

$$\begin{aligned}
&\|\boldsymbol{\Phi}(\mathbf{x}_1) - \boldsymbol{\Phi}(\mathbf{x}_2)\| \approx \|\boldsymbol{\Phi}(\mathbf{x}_1) - (\boldsymbol{\Phi}(\mathbf{x}_1) + \nabla\boldsymbol{\Phi}(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1))\| \\
&= \sqrt{(\mathbf{x}_2 - \mathbf{x}_1)^T (\nabla\boldsymbol{\Phi}(\mathbf{x}_1))^T \nabla\boldsymbol{\Phi}(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1)} \\
&= \sqrt{(\mathbf{x}_2 - \mathbf{x}_1)^T (\mathbf{id} + \nabla\mathbf{u}(\mathbf{x}_1))^T (\mathbf{id} + \nabla\mathbf{u}(\mathbf{x}_1))(\mathbf{x}_2 - \mathbf{x}_1)} \\
&= \|\mathbf{x}_1 - \mathbf{x}_2\| \sqrt{1 + \mathbf{e}_{x_1 x_2}^T (\nabla\mathbf{u}(\mathbf{x}_1)^T + \nabla\mathbf{u}(\mathbf{x}_1) + \nabla\mathbf{u}(\mathbf{x}_1)^T \nabla\mathbf{u}(\mathbf{x}_1)) \mathbf{e}_{x_1 x_2}},
\end{aligned} \tag{5.5}$$

where $\mathbf{e}_{x_1 x_2}$ denotes a unit vector in the direction from $\mathbf{x}_1$ to $\mathbf{x}_2$. Using the Taylor series $\sqrt{1+x} \approx 1 + \frac{1}{2}x$, this results in

$$\frac{\|\boldsymbol{\Phi}(\mathbf{x}_1) - \boldsymbol{\Phi}(\mathbf{x}_2)\|}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \approx 1 + \frac{1}{2} \mathbf{e}_{x_1 x_2}^T (\nabla\mathbf{u}(\mathbf{x}_1)^T + \nabla\mathbf{u}(\mathbf{x}_1) + \nabla\mathbf{u}(\mathbf{x}_1)^T \nabla\mathbf{u}(\mathbf{x}_1)) \mathbf{e}_{x_1 x_2}. \tag{5.6}$$

With $\boldsymbol{\varepsilon}(\mathbf{x}_1) := \nabla\mathbf{u}(\mathbf{x}_1)^T + \nabla\mathbf{u}(\mathbf{x}_1) + \nabla\mathbf{u}(\mathbf{x}_1)^T\nabla\mathbf{u}(\mathbf{x}_1)$, the strain $\varepsilon_{x_1x_2}$ is given by

$$\varepsilon_{x_1x_2} = \frac{\Delta l}{l} = \frac{\|\boldsymbol{\Phi}(\mathbf{x}_1) - \boldsymbol{\Phi}(\mathbf{x}_2)\| - \|\mathbf{x}_1 - \mathbf{x}_2\|}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \approx \mathbf{e}_{x_1x_2}^T \boldsymbol{\varepsilon}(\mathbf{x}_1)\mathbf{e}_{x_1x_2}. \tag{5.7}$$

This leads to the definition of the strain tensor:

Definition 2 (Strain)**.** *The second-order tensor* $\boldsymbol{\varepsilon} \in \mathbb{R}^{3\times3}$

$$\boldsymbol{\varepsilon} := \frac{1}{2}(\nabla\mathbf{u}^T + \nabla\mathbf{u} + \nabla\mathbf{u}^T\nabla\mathbf{u}) \tag{5.8}$$

is called the ***Green-St. Venant strain tensor****. Dropping the quadratic terms* $(\nabla\mathbf{u})^T\nabla\mathbf{u}$ *leads to the* ***infinitesimal*** *or* ***linear strain tensor***

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u}^T + \nabla\mathbf{u}\right). \tag{5.9}$$

Both tensors are popular choices in physically-based animation [NMK*06]. Note that the Green-St.- Venant strain tensor could be written as $\nabla\boldsymbol{\Phi}^T\nabla\boldsymbol{\Phi} - \mathbf{id}$, which implies that it is invariant under rotations. The components of the infinitesimal strain tensor are given as

$$\varepsilon_{ij} = \frac{1}{2}\left(\partial_i u_j + \partial_j u_i\right), \quad i,j \in \{x,y,z\}. \tag{5.10}$$

Thus, the diagonal entries are equal to the normal strain (5.2), while the off-diagonal entries $\varepsilon_{ij}, i \neq j$ are connected to the shear strain γ_{ij} (5.4) via

$$\gamma_{ij} = 2\varepsilon_{ij}. \tag{5.11}$$

The factor of 2 stems from the fact that the strain tensor $\boldsymbol{\varepsilon}$ does not measure the shear strain directly, but only the impact of the shear deformation onto the length of the diagonal, which is explained in Fig. 5.5: The shear strain γ_{xy} causes an elongation of the diagonal, which is measured as a normal strain ε_d along the diagonal. This results in $\varepsilon_d = \frac{1}{2}\gamma_{xy}$.

5.1.2 Symmetry of stress and strain

Obviously, the strain tensor is symmetric, and therefore, it contains only six independent values. In general, this is not the case for the stress tensor, if the stress results from external forces that cause a torque. However, it can be shown that torque-free forces result in a symmetric stress tensor.

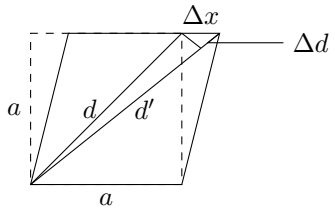


Figure 5.5: A cube with edge length a is subjected to a shear stress σ_{xy}. The shear strain $\gamma_{xy} = \frac{\Delta x}{a}$ causes an elongation of the diagonal of $\Delta d = \frac{\Delta x}{\sqrt{2}}$, which results in a strain $\varepsilon_d = \frac{\Delta x/\sqrt{2}}{\sqrt{2}a} = \frac{\Delta x}{2a} = \frac{1}{2}\gamma_{xy}$.

Therefore, the forces in Fig. 5.3 (a) are considered. The forces $\mathbf{F}$ and $-\mathbf{F}$ cause a torque $\mathbf{T}$ with the norm

$$\|\mathbf{T}_F\| = 2 \cdot \frac{dy}{2} \cdot \|\mathbf{F}\| = dy \cdot \sigma_{xy}\, dx\, dz = \sigma_{xy}\, dV. \tag{5.12}$$

The forces $\mathbf{F}'$ and $-\mathbf{F}'$ cause a torque $\mathbf{T}_{F'}$ with opposite direction and norm

$$\|\mathbf{T}_{F'}\| = 2 \cdot \frac{dx}{2} \|\mathbf{F}'\| = dx \cdot \sigma_{yx}\, dy\, dz = \sigma_{yx}\, dV. \tag{5.13}$$

As $\mathbf{T}_F$ and $\mathbf{T}_{F'}$ have opposite direction, the difference of their absolute values has to be zero in order to eliminate the torque. This results in $\sigma_{xy} = \sigma_{yx}$, which states that the stress tensor $\boldsymbol{\sigma}$ is symmetric for a torque-free set of forces. As internal forces that are caused by a deformation are torque-free, the stress tensor caused by a deformation is symmetric.

Therefore, both stress and strain tensor have only six independent values and can be written as the six-dimensional vectors

$$\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})^T \tag{5.14}$$

$$\boldsymbol{\varepsilon} = (\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{xy}, \gamma_{xz}, \gamma_{yz})^T \tag{5.15}$$

$$= (\varepsilon_x, \varepsilon_y, \varepsilon_z, 2\varepsilon_{xy}, 2\varepsilon_{xz}, 2\varepsilon_{yz})^T \tag{5.16}$$

In the remainder, the symbols $\boldsymbol{\sigma}, \boldsymbol{\varepsilon}$ denote the vector notation of stress and strain. If the tensor notation is employed, this is indicated by a lower index $\boldsymbol{\varepsilon}_T, \boldsymbol{\sigma}_T$.

5.1.3 Deformation parameters

The relation between stress and strain is defined by the material properties, which are given by the deformation parameters. Assuming linearly elastic and

isotropic materials, the deformation behavior is characterized by two independent constants, the Young modulus E and the Poisson ratio ν.

In the situation of Fig. 5.2 (a), the Young modulus linearly relates the stress σ_x and the strain ε_x as

$$\sigma_x = E\varepsilon_x. \tag{5.17}$$

Thus, the elongation Δx can be written as

$$\Delta x = \frac{1}{E}\frac{\|\mathbf{F}\|\,dx}{A}, \tag{5.18}$$

and is proportional to the force $\|\mathbf{F}\|$, the resting length dx and $\frac{1}{A}$. From (5.18), it follows that the elongation Δx gets smaller if E gets larger, which means that the Young modulus characterizes the stiffness of an object.

In addition to the normal strain, it can be observed that applying a force $\mathbf{F}$ in x-direction causes a transversal contraction Δy and Δz in y- and z-direction, which is depicted in Fig. 5.6.

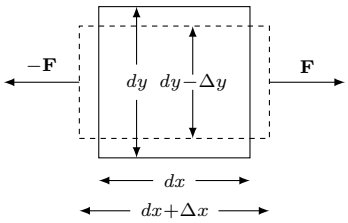


Figure 5.6: Illustration of the Poisson ratio. The picture shows a side view of Fig. 5.2 (a) and illustrates the transversal contraction Δy in y-direction caused by the strain ε_x.

Hence, the stress ε_x in x-direction causes a stress $\varepsilon_y = \frac{-\Delta y}{dy}$ in y-direction, and similarly in z-direction. The Poisson ratio ν is defined as the quotient

$$\nu := \frac{-\varepsilon_y}{\varepsilon_x} \overset{(\star)}{=} \frac{-\varepsilon_z}{\varepsilon_x}, \tag{5.19}$$

where $(\star)$ holds as the material is assumed to be isotropic. Hence, the Poisson ratio is dimensionless and measures the relation between the change of length in the direction of the force and the change of thickness orthogonal to the force direction. Solving (5.19) for ε_y leads to

$$\varepsilon_y = \varepsilon_z = -\nu\varepsilon_x, \tag{5.20}$$

which states that the strain ε_x caused by a force in x-direction also causes a strain ε_y and ε_z in y- and z-direction.

Similar to the Young modulus E, the *shear modulus* G connects the shear stress and shear strain. In the situation of Fig. 5.2 (b), this can be written as

$$\sigma_{xy} = G\gamma_{xy}, \tag{5.21}$$

and the deflection Δx becomes

$$\Delta x = \frac{1}{G}\frac{\|\mathbf{F}\|\,dy}{A}. \tag{5.22}$$

However, G is not an independent constant, but is associated to the Young modulus and the Poisson ratio as

$$G = \frac{E}{2(1+\nu)}. \tag{5.23}$$

With respect to the parameter estimation in Sec. 7.1, the possible values of E and ν are deduced in the following.

Obviously, the Young modulus is always greater than zero, but not upper bounded. The greater E is, the stiffer the material behaves. Typical values are in the order of $200\frac{kN}{mm^2}$ for steel, within the range between 1000 and $5000\frac{N}{mm^2}$ for Polyester and in the order of $100\frac{N}{dm^2} = 0.01\frac{N}{mm^2}$ for soft foam-like materials (cf. [CZ05]).

In contrast, the range for the Poisson ratio is bounded. First, if an object is expanded in one direction, it is contracted perpendicular to this direction, and vice versa. In (5.19), this means that $\varepsilon_y \leq 0$ if $\varepsilon_x > 0$, and hence, $\nu \geq 0$. Further, it is reasonable to assume that the volume does not shrink if an object is expanded in x-direction, i. e. if $\varepsilon_x > 0$. The volume change can be approximated by the first-order Taylor series

$$\begin{aligned} V(dx+\Delta x, dy-\Delta y, dz-\Delta z) =& V(dx,dy,dz) \\ &+ \Delta x\cdot dy\,dz - \Delta y\cdot dx\,dz - \Delta z\cdot dx\,dy. \end{aligned} \tag{5.24}$$

Thus, the relative volume change is given by

$$\begin{aligned} \frac{\Delta V}{V} &= \frac{\Delta x\cdot dy\,dz - \Delta y\cdot dx\,dz - \Delta z\cdot dx\,dy}{dx\,dy\,dz} \\ &= \frac{\Delta x}{dx} - \frac{\Delta y}{dy} - \frac{\Delta z}{dz} \\ &= \varepsilon_x + \varepsilon_y + \varepsilon_z \\ &\overset{(5.19)}{=} \varepsilon_x(1-2\nu). \end{aligned} \tag{5.25}$$

Assuming that ΔV is greater or equal to zero leads to $\varepsilon_x(1-2\nu) \geq 0 \Leftrightarrow \nu \leq 0.5$, where $\nu = 0.5$ would imply perfect volume conservation. Hence, the

Poisson ratio is bounded by $0 \leq \nu \leq 0.5$. Typical values for materials like foam lie within the range between 0.1 and 0.4. Note that the last equation in (5.25) is valid only for isotropic materials.

5.1.4 Stress-strain-relation

For linearly elastic materials, stress and strain depend linearly on each other. This was already used in the definition of the deformation parameters in the previous section. In the most general version, each entry of $\boldsymbol{\varepsilon}$ could depend on each entry of $\boldsymbol{\sigma}$. Thus, the relation is given by a fourth-order tensor $\mathbf{C}$:

$$\sigma_{T,ij} = \sum_{k,l} C_{ijkl} \varepsilon_{T,kl}, \tag{5.26}$$

which is the generalization of Hooke's law (Sec. 3.2). However, due to the symmetries of stress and strain, $\mathbf{C}$ can be reduced to a second-order tensor and (5.26) becomes

$$\sigma_i = \sum_j C_{ij} \varepsilon_j \tag{5.27}$$

In the following, $\mathbf{C}$ is expressed by the deformation parameters E and ν. Due to (5.20), the strain in x-direction is given by $\varepsilon_x - \nu\varepsilon_y - \nu\varepsilon_z$, and the following relations hold:

$$\begin{aligned}
\varepsilon_x &= \frac{\sigma_x}{E} - \nu\frac{\sigma_y}{E} - \nu\frac{\sigma_z}{E} & \qquad \gamma_{xy} &= \frac{\tau_{xy}}{G} = \frac{\tau_{xy}}{E} \cdot 2(1+\nu) \\
\varepsilon_y &= \frac{\sigma_y}{E} - \nu\frac{\sigma_x}{E} - \nu\frac{\sigma_z}{E} & \qquad \gamma_{xz} &= \frac{\tau_{xz}}{G} = \frac{\tau_{xz}}{E} \cdot 2(1+\nu) \\
\varepsilon_z &= \frac{\sigma_z}{E} - \nu\frac{\sigma_x}{E} - \nu\frac{\sigma_y}{E} & \qquad \gamma_{yz} &= \frac{\tau_{yz}}{G} = \frac{\tau_{yz}}{E} \cdot 2(1+\nu).
\end{aligned} \tag{5.28}$$

These relations can be written in matrix-vector-notation

$$\boldsymbol{\varepsilon} = \mathbf{M}\boldsymbol{\sigma} \qquad \Leftrightarrow \qquad \boldsymbol{\sigma} = \mathbf{M}^{-1}\boldsymbol{\varepsilon}, \tag{5.29}$$

which defines the matrix $\mathbf{C}$ for the stress-strain-relation:

$$\mathbf{C} := \mathbf{M}^{-1} = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\nu \end{pmatrix}. \tag{5.30}$$

5.1.5 Potential energy

As the deformation energy in linearly elastic objects depends only on the current state, but not on the way how this state was obtained, such an object defines a conservative system. Thus, the internal forces correspond to the gradient of the potential energy.

To compute the deformation energy for normal and shear strain, the volume elements in Fig. 5.2 are considered. The force $\mathbf{F}$ acting in x-direction enlarges the element by an amount Δx. Similar to the spring energy (Sec. 3.2), the force is linearly increasing with the magnitude of Δx. Thus, the deformation energy is given by

$$E_{def,x} = \frac{1}{2} F_x \Delta x = \frac{1}{2}(\sigma_x \, dy \, dz)(\varepsilon_x \, dx) = \frac{1}{2}\sigma_x \varepsilon_x \, dV. \tag{5.31}$$

Similarly, the shear force acting on the xy-plane of the volume element is linearly increasing with Δx and leads to a deformation energy

$$E_{def,xy} = \frac{1}{2}\gamma_{xy}\tau_{xy} \, dV. \tag{5.32}$$

Thus, the deformation energy of the volume element is

$$\begin{aligned} E_{def} &= \frac{1}{2}(\sigma_x \varepsilon_x + \sigma_y \varepsilon_y + \sigma_z \varepsilon_z + \gamma_{xy}\tau_{xy} + \gamma_{xz}\tau_{xz} + \gamma_{yz}\tau_{yz}) dV \\ &= \frac{1}{2}\boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \, dV, \end{aligned} \tag{5.33}$$

and the deformation energy density is given by $\frac{1}{2}\boldsymbol{\sigma}^T \boldsymbol{\varepsilon}$. The deformation energy over the whole object then can be computed by the volume integral

$$E_{def} = \int_\Omega \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} \, dV. \tag{5.34}$$

In the tensor notation, the energy is given as $\int_\Omega \boldsymbol{\sigma}_T : \boldsymbol{\varepsilon}_T \, dV$, where $\boldsymbol{\sigma}_T : \boldsymbol{\varepsilon}_T$ denotes the sum of the component-wise products. For this formulas, it is important that the off-diagonal elements in the tensor notation are $\varepsilon_{T,ij}$ as defined in Def. 2, while the vector notation uses the shear strains γ_{ij}.

5.2 Linear Finite Element Method

The deduction of the linear Finite Element Method basically follows the derivation given in [CB02]. In order to compute the internal forces based on continuum mechanics, the gradient of the deformation energy (5.34) has to be calculated. The main step thereby is to find a solution for the partial differential equation

$$\boldsymbol{\varepsilon}_T = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T) \tag{5.35}$$

defining the infinitesimal strain tensor of a deformed object for a given displacement field $\mathbf{u}$. The solution $\boldsymbol{\varepsilon}_T$ of this equation can be approximated by a Finite Element formulation. As introduced in Sec. 3.1, an object is represented by a tetrahedral mesh, and therefore, the tetrahedrons are used as the finite subsets for the Finite Element Method. Thus, let e be a tetrahedron with vertices $\mathbf{x}_1, \ldots, \mathbf{x}_4$, and let the displacements of the vertices be given as $\mathbf{q}_1, \ldots, \mathbf{q}_4$. To interpolate the displacement field $\mathbf{u}$, appropriate shape functions have to be chosen. In the linear Finite Element Method, usually the so called *barycentric coordinates* are used, which are defined as follows. Each point $\mathbf{x}$ within e can be written as a linear combination

$$\mathbf{x} = \sum_{i=1}^{4} \lambda_i \mathbf{x}_i \quad \text{with} \sum_{i=1}^{4} \lambda_i = 1. \tag{5.36}$$

Then, the coefficients $\lambda_i = \lambda_i(\mathbf{x})$, which naturally depend on the point $\mathbf{x}$, are called the barycentric coordinates of $\mathbf{x}$. Note that λ_4 could also be written as $1 - \lambda_1 - \lambda_2 - \lambda_3$, and therefore, only three of the coefficients are independent. Using these coefficients, the displacement function $\mathbf{u}(\mathbf{x})$ is interpolated as

$$\begin{aligned} \mathbf{u}(\mathbf{x}) &= \sum_{i=1}^{4} \lambda_i(\mathbf{x}) \mathbf{q}_i \\ &= \mathbf{q}_4 + \sum_{i=1}^{3} \lambda_i(\mathbf{x})(\mathbf{q}_i - \mathbf{q}_4) \end{aligned} \tag{5.37}$$

As the shape functions have the Kronecker-delta property that $\lambda_i(\mathbf{x}_j) = 1$ if $i = j$ and $\lambda_i(\mathbf{x}_j) = 0$ if $i \neq j$, it follows that $\mathbf{u}(\mathbf{x}_i) = \mathbf{q}_i$. Therefore, the shape functions are also called *associated* to the vertices. Note that this is not generally mandatory in the Finite Element Method.

Using linear shape functions obviously results in a linear approximation of the displacement field, which restricts the strain to be constant over a tetrahedron. However, for small elements, a piecewise constant strain function sufficiently approximates the true strain, which justifies the choice of linear shape functions.

In order to evaluate (5.35), the gradient $\nabla \mathbf{u}$ of the displacement function has to be calculated. Here, ∇ denotes the derivative with respect to the spatial coordinates $\mathbf{x} = (x, y, z)$. However, $\mathbf{u}$ is not given explicitly, but only as an interpolation of the displacements $\mathbf{q}_i$. Thus, it depends on the three independent barycentric coordinates $\boldsymbol{\lambda} := (\lambda_1, \lambda_2, \lambda_3)$, which themselves depend on $\mathbf{x}$. Thus, the chain rule $\nabla_x \mathbf{u} = \nabla_\lambda \mathbf{u} \cdot \nabla_x \boldsymbol{\lambda}$ has to be used to obtain the gradient of $\mathbf{u}$,

where the derivatives are specified as ∇_x for the spatial derivative and ∇_λ for the derivative with respect to the barycentric coordinates.

First, the derivative $\nabla_\lambda \mathbf{u}$ is given as

$$\begin{aligned} \nabla_\lambda \mathbf{u} &\overset{(5.37)}{=} (\mathbf{q}_1 - \mathbf{q}_4, \mathbf{q}_2 - \mathbf{q}_4, \mathbf{q}_3 - \mathbf{q}_4) \\ &=: (\mathbf{q}_{14}, \mathbf{q}_{24}, \mathbf{q}_{34}) \end{aligned} \tag{5.38}$$

For the derivative $\nabla_x \boldsymbol{\lambda}$, it is helpful to use the derivative of inverse functions, which states that fore a function $\mathbf{g}$ and its inverse function $\mathbf{g}^{-1}$, the derivatives are connected as $(\nabla \mathbf{g}^{-1})(\mathbf{g}(\mathbf{x})) = (\nabla \mathbf{g})(x)^{-1}$.

As $\boldsymbol{\lambda}(\mathbf{x})$ and its inverse $\mathbf{x}(\boldsymbol{\lambda})$ are linear functions, their gradients are constant and do not depend on the argument of the function. This results in $\nabla_x \boldsymbol{\lambda} = (\nabla_\lambda \mathbf{x})^{-1}$, where the latter is more easy to compute. Starting with the barycentric coordinates of a tetrahedron,

$$\begin{aligned} \mathbf{x}(\lambda) &= \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \lambda_3 \mathbf{x}_3 + \lambda_4 \mathbf{x}_4 \\ &= \mathbf{x}_4 + \lambda_1(\mathbf{x}_1 - \mathbf{x}_4) + \lambda_2(\mathbf{x}_2 - \mathbf{x}_4) + \lambda_3(\mathbf{x}_3 - \mathbf{x}_4) \\ &=: \mathbf{x}_4 + \lambda_1 \mathbf{x}_{14} + \lambda_2 \mathbf{x}_{24} + \lambda_3 \mathbf{x}_{34}, \end{aligned} \tag{5.39}$$

it can be seen that $\nabla_\lambda \mathbf{x}$ is similar to $\nabla_\lambda \mathbf{u}$:

$$\begin{aligned} \nabla_\lambda \mathbf{x} &= (\mathbf{x}_{14}, \mathbf{x}_{24}, \mathbf{x}_{34}) \\ &= \begin{pmatrix} x_{14} & x_{24} & x_{34} \\ y_{14} & y_{24} & y_{34} \\ z_{14} & z_{24} & z_{34} \end{pmatrix}. \end{aligned} \tag{5.40}$$

Inverting (5.40) leads to

$$\nabla_x \boldsymbol{\lambda} = \frac{1}{\det(\nabla_\lambda \mathbf{x})} \begin{pmatrix} y_{24}z_{34} - z_{24}y_{34} & z_{24}x_{34} - x_{24}z_{34} & x_{24}y_{34} - y_{24}x_{34} \\ z_{14}y_{34} - y_{14}z_{34} & x_{14}z_{34} - z_{14}x_{34} & y_{14}x_{34} - x_{14}y_{34} \\ y_{14}z_{24} - z_{14}y_{24} & z_{14}x_{24} - x_{14}z_{24} & x_{14}y_{24} - y_{14}x_{24} \end{pmatrix} =: \mathbf{A} \tag{5.41}$$

In total, the gradient $\nabla_x \mathbf{u}$ of the displacement function can be summarized as

$$\nabla_x \mathbf{u} = (\mathbf{q}_{14}, \mathbf{q}_{24}, \mathbf{q}_{34}) \cdot \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \tag{5.42}$$

Thus, the normal strain ε_x which is given as the component $\varepsilon_{T,11}$ of the strain tensor $\boldsymbol{\varepsilon}_T = \frac{1}{2}(\nabla_x \mathbf{u} + \nabla_x \mathbf{u}^T)$ has the form

$$\begin{aligned}\varepsilon_x &= A_{11}q_{14,x} + A_{12}q_{23,x} + A_{13}q_{34,x} \\ &= A_{11}q_{1,x} + A_{12}q_{2,x} + A_{13}q_{3,x} + (-A_{11} - A_{12} - A_{13})q_{4,x}\end{aligned} \tag{5.43}$$

Similarly, the other components can be derived. With $\mathbf{Q}_e := (\mathbf{q}_1^T, \dots, \mathbf{q}_4^T)^T$ summarizing the individual displacements of the tetrahedron e, this allows to write the strain vector $\boldsymbol{\varepsilon}$ (5.16) as a matrix-vector-multiplication

$$\boldsymbol{\varepsilon} = \widetilde{\mathbf{A}}\mathbf{Q}_e, \tag{5.44}$$

where $\widetilde{\mathbf{A}}$ arises from $\mathbf{A}$ as

$$\widetilde{\mathbf{A}} = \begin{pmatrix} A_{11} & 0 & 0 & A_{12} & 0 & 0 & A_{13} & 0 & 0 & -\overline{A}_1 & 0 & 0 \\ 0 & A_{21} & 0 & 0 & A_{22} & 0 & 0 & A_{23} & 0 & 0 & -\overline{A}_2 & 0 \\ 0 & 0 & A_{31} & 0 & 0 & A_{32} & 0 & 0 & A_{33} & 0 & 0 & -\overline{A}_3 \\ 0 & A_{31} & A_{21} & 0 & A_{32} & A_{22} & 0 & A_{33} & A_{23} & 0 & -\overline{A}_3 & -\overline{A}_2 \\ A_{31} & 0 & A_{11} & A_{32} & 0 & A_{12} & A_{33} & 0 & A_{13} & -\overline{A}_3 & 0 & -\overline{A}_1 \\ A_{21} & A_{11} & 0 & A_{22} & A_{12} & 0 & A_{23} & A_{13} & 0 & -\overline{A}_2 & -\overline{A}_1 & 0 \end{pmatrix} \tag{5.45}$$

with $\overline{A}_i = A_{i1} + A_{i2} + A_{i3}$.
Thus, the strain energy of the deformed tetrahedron e can be written as

$$\begin{aligned} E_{def,e} &= \frac{1}{2}\int_e \boldsymbol{\sigma}^T\boldsymbol{\varepsilon}\, dV \\ \text{(Sec. 5.1.4)} &= \frac{1}{2}\int_e \boldsymbol{\varepsilon}^T\mathbf{C}\boldsymbol{\varepsilon}\, dV \\ (\boldsymbol{\varepsilon}=const) &= \frac{1}{2}\boldsymbol{\varepsilon}^T\mathbf{C}\boldsymbol{\varepsilon}\int_e dV \\ \text{(5.44)} &= \frac{1}{2}\mathbf{Q}_e^T\widetilde{\mathbf{A}}^T\mathbf{C}\widetilde{\mathbf{A}}\mathbf{Q}_e\int_e dV \\ &= \frac{1}{2}\mathbf{Q}_e^T\widetilde{\mathbf{A}}^T\mathbf{C}\widetilde{\mathbf{A}}\mathbf{Q}_e V_e \\ &= \frac{1}{2}\mathbf{Q}_e^T\mathbf{K}_e\mathbf{Q}_e, \end{aligned} \tag{5.46}$$

where $\mathbf{K}_e \in \mathbb{R}^{12\times 12}$ is called the *stiffness matrix* of element e. The volume $V_e = \int_e dV$ can be calculated using the scalar triple product $\frac{1}{6}\mathbf{x}_{14} \cdot (\mathbf{x}_{24} \times \mathbf{x}_{34})$.

In contrast to non-linear Finite Elements, the stiffness matrix $\mathbf{K}_e$ can be precomputed for linear shape functions. This results in a large performance gain, as otherwise the integral (5.46) has to be evaluated in each time step, which can be avoided in the linear Finite Element Method.

With this formula for the deformation energy, the internal forces can be computed as

$$\mathbf{F}_e^{int}(\mathbf{Q}_e) = -\nabla_{Q_e} E_{def,e}(\mathbf{Q}_e) = -\mathbf{K}_e\mathbf{Q}_e, \tag{5.47}$$

where $\mathbf{F}_e^{int} = (\mathbf{f}_1^T, \ldots, \mathbf{f}_4^T)^T$ summarizes the individual forces acting at the vertices $\mathbf{x}_1, \ldots, \mathbf{x}_4$. Thus, in an equilibrium condition, it holds that

$$\mathbf{F}^{ext} = -\mathbf{F}_e^{int} = \mathbf{K}_e\mathbf{Q}_e. \tag{5.48}$$

Inserting (5.47) into the equation of motion (3.9) results in

$$\mathbf{M}\ddot{\mathbf{X}} = \mathbf{F}^{ext} - \mathbf{K}_e\mathbf{Q}_e, \tag{5.49}$$

which can be integrated numerically (Sec. 3.3). Depending on the connectivity of the object, all individual stiffness matrices $\mathbf{K}_e$ can be assembled into a stiffness matrix $\mathbf{K}$ for the whole object (see e. g. [CB02]). This can be useful for solving static equilibrium conditions of the form $\mathbf{KQ} = \mathbf{F}^{ext}$ for a given external force $\mathbf{F}^{ext}$. However, $\mathbf{K}$ is a sparse matrix, as each vertex lies in a few tetrahedrons only. Thus, for the dynamic case, it is more useful to compute the forces for each tetrahedron individually and to assign them to the corresponding vertices. Consequently, *DefCol Studio* uses this implementation.

For later use in Chapter 9, the deformation energy E_{def} of the whole object can be computed as the sum of the individual deformation energies $E_{def,e}$:

$$\begin{aligned} E_{def} = \sum_e E_{def,e} &= \sum_e \frac{1}{2}\int_e \boldsymbol{\sigma}^T\boldsymbol{\varepsilon}\, dV \\ &= \sum_e \frac{1}{2}\mathbf{Q}_e^T\mathbf{K}_e\mathbf{Q}_e \end{aligned} \tag{5.50}$$

5.2.1 Properties of the stiffness matrix

For later use in Sec. 7.1, some properties of the stiffness matrix are considered in this section. From the decomposition $\mathbf{K}_e = V_e\widetilde{\mathbf{A}}^T\mathbf{C}\widetilde{\mathbf{A}}$, it can be seen that $\mathbf{K}_e$ is symmetric and hence, it is orthogonally diagonalizable. Further, $\mathbf{K}_e$ is positive semidefinite, as $\mathbf{Q}_e^T\mathbf{K}_e\mathbf{Q}_e = \|\sqrt{C}\widetilde{\mathbf{A}}\mathbf{Q}_e\|_2^2 \geq 0$. As $\widetilde{\mathbf{A}} \in \mathbb{R}^{6\times 12}$, the rank of $\mathbf{K}_e$ is less or equal to 6, and at least six eigenvalues are zero.

Using (5.35), it can be shown that exactly six eigenvalues are zero and that they correspond to translations and linearized rotations. Due to the linear interpolation of $\mathbf{u}(\mathbf{x})$, it can be written as an affine mapping $\mathbf{u}(\mathbf{x}) = \mathbf{Bx} + \mathbf{d}$ with $\mathbf{B} \in \mathbb{R}^{3\times 3}$ and $\mathbf{d} \in \mathbb{R}^3$. First, for a pure translation, it follows that $\nabla\mathbf{u} = 0$. As the translation has three possible directions, three of the zero-eigenvalues belong to translations, which is a desired property. Otherwise, it follows $\nabla\mathbf{u} = \mathbf{B}$, and $\nabla\mathbf{u} + \nabla\mathbf{u}^T = 0$ (5.35) holds if and only if $\mathbf{B}$ is skew-symmetric. With the exponential function for matrices, $\exp(\mathbf{B})$ is a rotation

if and only if $\mathbf{B}$ is skew-symmetric, and $\mathbf{B}$ is a first-order approximation of $\exp(\mathbf{B})$. Thus, $\mathbf{B}$ represents a linearized rotation. As skew-symmetric matrices have three degrees of freedom, exactly three zero-eigenvalues of $\mathbf{K}_e$ correspond to linearized rotations. Hence, exactly the six remaining eigenvalues of $\mathbf{K}_e$ are non-zero and correspond to the class of *symmetric* matrices.

Further, the non-zero eigenvalues not only depend on the deformation parameters, but also on the shape of tetrahedrons [She02]. Thus, they get larger if elements contain sharp angles or are flat such as slivers, which affects the stability of the simulation. This should be considered by the mesh generation algorithm, but it can also be alleviated e. g. by using appropriate damping schemes (Sec. 8.3).

5.2.2 Co-rotational Finite Element Method

The linear Finite Element Method as described above is not rotationally invariant, as rotations cannot be represented by a linear interpolation of the displacement of the tetrahedral vertices [BIT09]. This can be seen in the fact that pure rotations of objects cause forces, and that there are deformed states that do not cause any forces, which results in volume artifacts [MG04]. These states correspond to the skew-symmetric deformations. [BIT09] also note that using the Green-St. Venant strain tensor does not alleviate this problem, although it is rotationally invariant, as already $\mathbf{u}$ is not interpolated correctly. However, the linear interpolation is essential for the performance of the linear Finite Element Method, and therefore, rotations have to be handled separately.

To overcome this drawback, [HS04, MG04] proposed the co-rotational Finite Element Method. They extract the rotational part of the deformation and compute the internal forces for the back-rotated deformation. [HS04] proposed to employ a polar decomposition to the deformation gradient $\nabla\mathbf{\Phi}$, which is a linear mapping $\mathbf{B}_e$ in the case of linear Finite Elements (Fig. 5.7). Similarly, [MG04] introduced a geometric way to compute the deformation gradient $\mathbf{B}_e$ in the linear case, and also proposed a polar decomposition to find the rotational part. This idea is summarized in this section.

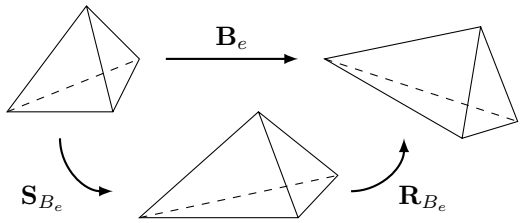


Figure 5.7: The deformation $\mathbf{B}_e$ of a tetrahedron e can be decomposed into a rotation $\mathbf{R}_{B_e}$ and a symmetric part $\mathbf{S}_{B_e}$ using a polar decomposition $\mathbf{B}_e = \mathbf{R}_{B_e}\mathbf{S}_{B_e}$.

For a tetrahedron e, let $\mathbf{x}_1^0, \ldots, \mathbf{x}_4^0$ denote the positions of the vertices in its resting state, while $\mathbf{x}_1, \ldots, \mathbf{x}_4$ denote the positions of the deformed tetrahedron.

Then, the deformed positions $\mathbf{x}_i$ can be written as $\mathbf{x}_i = \mathbf{B}_e\mathbf{x}_i^0 + \mathbf{d}$ with a linear mapping $\mathbf{B}_e \in \mathbb{R}^{3\times3}$ and a translation $\mathbf{d} \in \mathbb{R}^3$. This defines a system of linear equations, which can be written in homogeneous notation as

$$\begin{pmatrix} \mathbf{B}_e & \mathbf{d} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^0 & \mathbf{x}_2^0 & \mathbf{x}_3^0 & \mathbf{x}_4^0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{5.51}$$

Thus, it follows

$$\begin{pmatrix} \mathbf{B}_e & \mathbf{d} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^0 & \mathbf{x}_2^0 & \mathbf{x}_3^0 & \mathbf{x}_4^0 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \tag{5.52}$$

where the inverse $\begin{pmatrix} \mathbf{x}_1^0 & \mathbf{x}_2^0 & \mathbf{x}_3^0 & \mathbf{x}_4^0 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1}$ can be precomputed. Then, a polar decomposition decomposes $\mathbf{B}_e = \mathbf{R}_{B_e}\mathbf{S}_{B_e}$ into a rotation and a symmetric part, and the co-rotational forces are computed as

$$\begin{aligned} \mathbf{F}_e^{rot} &= -\mathbf{R}_e\mathbf{K}_e(\mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0) \\ &=: -\mathbf{R}_e\mathbf{K}_e\mathbf{Q}_e^{rot}, \end{aligned} \tag{5.53}$$

where $\mathbf{X}_e := (\mathbf{x}_1^T, \mathbf{x}_2^T, \mathbf{x}_3^T, \mathbf{x}_4^T)^T$, similarly $\mathbf{X}_e^0$, and $\mathbf{Q}_e^{rot} := (\mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0)$ denotes the co-rotated displacement. $\mathbf{R}_e$ arises from $\mathbf{R}_{B_e}$ as

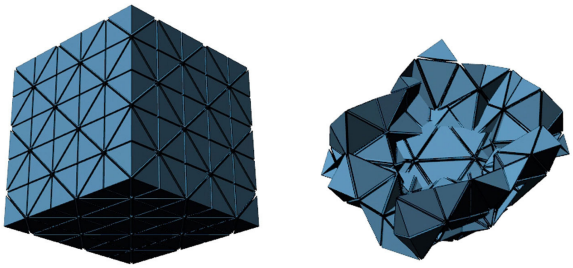
$$\mathbf{R}_e = \begin{pmatrix} \mathbf{R}_{B_e} & 0 & 0 & 0 \\ 0 & \mathbf{R}_{B_e} & 0 & 0 \\ 0 & 0 & \mathbf{R}_{B_e} & 0 \\ 0 & 0 & 0 & \mathbf{R}_{B_e} \end{pmatrix}. \tag{5.54}$$

This means that the deformed tetrahedron is first rotated back using $\mathbf{R}^T$, the co-rotated displacement $\mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0$ is used to compute the deformation forces, and the forces are rotated into the rotated coordinate frame of the deformed tetrahedron.

6

Inversion handling for dynamic simulations

In the previous chapter, the linear Finite Element Method was introduced which is based on linear elasticity theory. It was extended to the co-rotational Finite Element Method [MG04, HS04], which eliminates the problem that rotations are not handled correctly. Thus, it allows to use the linear Finite Element Method in order to simulate arbitrary deformations maintaining physical realism. However, it introduces a new drawback which affects the stability of dynamic simulations. If a tetrahedron gets inverted, i. e. one vertex crosses its opposite face, the co-rotational Finite Element Method keeps this element inverted, which causes erroneous equilibrium states and the breakdown of the whole simulation (Fig. 6.1).



(a) (b)

Figure 6.1: (a) shows the tetrahedral mesh of a cube falling onto the ground. (b) shows an erroneous equilibrium state after the impact if inverted tetrahedrons are not adequately handled.

At this point, it should be emphasized that inversion is not a problem of the linear Finite Element Method itself, but only of the co-rotational formulation. In Sec. 6.1, it is shown that problems with inverted elements are solely caused by the possibly improper rotation returned by the polar decomposition in the co-rotational Finite Element Method.

One way to address this problem would be to prevent the inversion of ele-

ments. However, this requires additional forces that are difficult to motivate. Further, it is difficult to guarantee that inversion is really avoided, and strategies like untangling meshes [ERM*03, VGS04] are also not guaranteed to work properly. Besides, there are cases where inversion is the correct behavior of elements (see e. g. [ITF04]). As the linear model itself works well for inverted elements, it seems to be more appropriate to allow the inversion and to adequately handle inverted elements.

In order to process inverted elements, a particular inversion direction has to be determined. This direction cannot be extracted by simply considering the current deformation state. If an inappropriate direction is chosen, the computation results in force discontinuities. [ITF04] introduced an inversion handling approach based on the heuristic assumption that an inverted tetrahedron is "as uninverted as possible". This implies that an inverted tetrahedron should get uninverted along the direction that causes the minimum movement of a vertex. While the underlying heuristic is very useful and appropriate, there exist cases where the inversion handling of [ITF04] is not conform to this assumption.

In this chapter, a new method is proposed to determine the inversion direction of inverted elements. The approach is similar to [ITF04]. It is based on the same heuristic that elements are as uninverted as possible. However, it is shown that the proposed method exceeds the approach of [ITF04] in the fact that it always chooses the direction that is implied by the heuristic assumption. Further, positive effects of the proposed inversion handling approach in dynamic simulations are illustrated. Compared to other strategies, inversions are efficiently resolved within a small number of simulation steps. The approach requires that the deformation gradient can be transformed to a diagonal form. Thus, it can be implemented within any material constitutive model. Moreover, the approach can easily be implemented in combination with an efficient handling of degenerated elements. Thus, it improves the robustness and stability of Finite Element based deformable modeling approaches.

Notation

The notation of this chapter is similar to Chapter 5. For a tetrahedron e, let $\mathbf{x}_1, \ldots, \mathbf{x}_4$ denote the current positions of its vertices. $\mathbf{x}_{cm}$ denotes its center of mass, and $\mathbf{x}_{i,rel} := \mathbf{x}_i - \mathbf{x}_{cm}$ denotes the position of $\mathbf{x}_i$ relative to the center of mass. An upper index $\mathbf{x}_i^0$ denotes the initial positions, and capital letters like $\mathbf{X}_e \in \mathbb{R}^{12}$ denote a vector that summarize the individual values of all vertices of e, i. e. $\mathbf{X}_e = (\mathbf{x}_1^T, \ldots, \mathbf{x}_4^T)^T$. $\mathbf{K}_e \in \mathbb{R}^{12\times 12}$ denotes the stiffness matrix of e, and $\mathbf{F}_e \in \mathbb{R}^{12}$ summarizes the forces $\mathbf{f}_1, \ldots, \mathbf{f}_4$ that act on the vertices.

The matrix $\mathbf{B}_e \in \mathbb{R}^{3\times 3}$ denotes the deformation gradient with the polar decomposition $\mathbf{B}_e = \mathbf{R}_{B_e}\mathbf{S}_{B_e}$. $\mathbf{R}_e, \mathbf{S}_e \in \mathbb{R}^{12\times 12}$ denote the blockdiagonal matrices

$$\mathbf{R}_e = \begin{pmatrix} \mathbf{R}_{B_e} & 0 & 0 & 0 \\ 0 & \mathbf{R}_{B_e} & 0 & 0 \\ 0 & 0 & \mathbf{R}_{B_e} & 0 \\ 0 & 0 & 0 & \mathbf{R}_{B_e} \end{pmatrix} \text{ and } \mathbf{S}_e = \begin{pmatrix} \mathbf{S}_{B_e} & 0 & 0 & 0 \\ 0 & \mathbf{S}_{B_e} & 0 & 0 \\ 0 & 0 & \mathbf{S}_{B_e} & 0 \\ 0 & 0 & 0 & \mathbf{S}_{B_e} \end{pmatrix}. \tag{6.1}$$

Further, let $\mathbf{Q}_e^{lin} := \mathbf{X}_e - \mathbf{X}_e^0$ denote the displacement of tetrahedron e as applied in the linear Finite Element Method, while $\mathbf{Q}_e^{rot} := \mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0$ denotes the co-rotational displacement (5.53). For later use, note that the co-rotational displacement could be equally defined as

$$\mathbf{Q}_e^{rot,S} = \mathbf{S}_e\mathbf{X}_e^0 - \mathbf{X}_e^0, \tag{6.2}$$

as $\mathbf{R}_e^T\mathbf{X}_e$ and $\mathbf{S}_e\mathbf{X}_e^0$ differ only by a translation. Thus, using $\mathbf{Q}_e^{rot,S}$ in the co-rotational force computation would lead to the same forces.

6.1 Problem description

In this section, it is shown that the problems concerning inverted elements are only due to the co-rotational formulation of the linear Finite Element Method. Therefore, first the polar decomposition of the deformation gradient is introduced, before the actual reason for inversion problems is described.

Note that the current state of a tetrahedron e can be deduced from the determinant of $\mathbf{B}_e$. The tetrahedron is inverted if and only if $\det(\mathbf{B}_e) < 0$, it is degenerated if and only if the deformation gradient is singular, i. e. $\det(\mathbf{B}_e) = 0$, and in a "normal" deformation state otherwise.

6.1.1 Polar decomposition

For the polar decomposition $\mathbf{B}_e = \mathbf{R}_{B_e}\mathbf{S}_{B_e}$, it is known that $\mathbf{R}_{B_e}$ is an orthogonal matrix and $\mathbf{S}_{B_e}$ is symmetric positive definite if $\mathbf{B}_e$ is nonsingular. It follows that $\det(\mathbf{R}_{B_e}) = \text{sign}(\det(\mathbf{B}_e))$. This implies that if e is inverted and hence, $\det(\mathbf{R}_e) = -1$, the internal forces are not only rotated, but also reflected as $\mathbf{R}_{B_e}$ is an improper rotation. Hence, they act to keep the tetrahedron inverted (Fig. 6.2). Furthermore, the reflection of forces implies force discontinuities during the inversion of a tetrahedron which cause visual artifacts.

For convenience, one variant of the polar decomposition is summarized here. First, the square root $\sqrt{\mathbf{B}_e^T\mathbf{B}_e}$ of $\mathbf{B}_e^T\mathbf{B}_e$ is computed. This is done by computing the diagonalization $\mathbf{D}_e = \mathbf{P}^T\mathbf{B}_e^T\mathbf{B}_e\mathbf{P}$ of $\mathbf{B}_e$, where $\mathbf{P}$ is an orthogonal matrix with the eigenvectors of $\mathbf{B}_e^T\mathbf{B}_e$ being the columns of $\mathbf{P}$. The diagonalization exists because $\mathbf{B}_e^T\mathbf{B}_e$ is symmetric. The square root of $\mathbf{B}_e^T\mathbf{B}_e$ then is simply given by $\sqrt{\mathbf{B}_e^T\mathbf{B}_e} = \mathbf{P}\sqrt{\mathbf{D}_e}\mathbf{P}^T$, where $\sqrt{\mathbf{D}_e}$ is obtained by taking the square root of all diagonal entries. Then, $\mathbf{S}_{B_e}$ is set to $\mathbf{S}_{B_e} := \sqrt{\mathbf{B}_e^T\mathbf{B}_e}$, and the rotation $\mathbf{R}_{B_e}$ is computed as $\mathbf{R}_{B_e} = \mathbf{B}_e\mathbf{S}_{B_e}^{-1}$. The inverse $\mathbf{S}_{B_e}^{-1}$ is simply computed as $\mathbf{P}\sqrt{\mathbf{D}_e}^{-1}\mathbf{P}^T$ and does not produce significant computational overhead.

An overview of various techniques of polar decomposition can be found in [ZZ95].

6.1.2 Inversion problems due to rotation

For a tetrahedron e, w. l. o. g. it can be assumed that $\mathbf{x}_{cm} = \mathbf{x}^0_{cm}$, as a translation does not cause any forces. For the analysis of the linear Finite Element Method, it is additionally assumed that the deformation does not contain a rotational part. However, the tetrahedron might be inverted. The stiffness matrix $\mathbf{K}_e$ is diagonalizable and positive semidefinite (Sec. 5.2.1). Let $\mathbf{k}_1, \ldots, \mathbf{k}_{12}$ be the eigenvectors of $\mathbf{K}_e$ with eigenvalues $\kappa_1, \ldots, \kappa_{12}$ and consider the displacement $\mathbf{Q}_e^{lin}$ represented using the eigenvectors, i. e. $\mathbf{Q}_e^{lin} = \sum_{i=1}^{12} q_{e,i}^{lin} \mathbf{k}_i$. The internal forces can simply be written as $\mathbf{F}_e^{lin} = \sum_{i=1}^{12} (-\kappa_i q_{e,i}^{lin} \mathbf{k}_i)$. For the dynamic equation, it follows $\mathbf{M}\ddot{\mathbf{X}}_e = \sum_{i=1}^{12} (-\kappa_i q_{e,i}^{lin}) \mathbf{k}_i$. The minus sign indicates that each component of the accelerating forces along an eigenvector acts opposite to the causal displacement. Hence, the displacement is reduced and the tetrahedron tends to its uninverted resting state. Note that in the argument, no knowledge was used whether the tetrahedron is inverted or not. Thus, it holds true also for inverted tetrahedrons.

In the same manner, the co-rotational displacement $\mathbf{Q}_e^{rot}$ can be decomposed into its components along the eigenvectors, and analogously, it can be concluded that each component causes a force that accelerates the vertices in the direction opposite to the causal displacement component. Thus, if the displacement is computed correctly, the forces are correct, too. Therefore, the only possible reason for any errors that occur when inverted elements are involved must be located in the rotation $\mathbf{R}_e$. As seen in Sec. 6.1.1, element inversion yields an improper rotation in the polar decomposition. Due to this fact, the computed forces cause an acceleration of the vertices to the positions of the reflected resting state. This is described in the two-dimensional example in Fig. 6.2.

6.2 Approach

In this section, the inversion handling approach is explained. First, the basic idea of the inversion handling method is outlined in Sec. 6.2.1. The idea of [ITF04] is briefly described in Sec. 6.2.2, and the properties of [ITF04] are discussed in Sec. 6.2.3. In particular, a case of a non-intuitive inversion direction is shown. The improved inversion handling approach, which is described in Sec. 6.2.4, addresses this issue. In Sec. 6.2.5, the incorporation of degenerated elements is explained.

6.2.1 Basic idea

Having recognized that problems with inverted elements are caused solely by the co-rotational formulation and come up due to polar decomposition, it is clear that eliminating the arising lacks should only be done by a modification of the polar decomposition. The idea of the co-rotational formulation was to separate the rotational part from the deformation, which is violated if $\mathbf{R}_{B_e}$ contains a reflection. Hence, the polar decomposition has to be modified in such a way that the reflection is contained in $\mathbf{S}_{B_e}$ and $\mathbf{R}_{B_e}$ is a proper rotation.

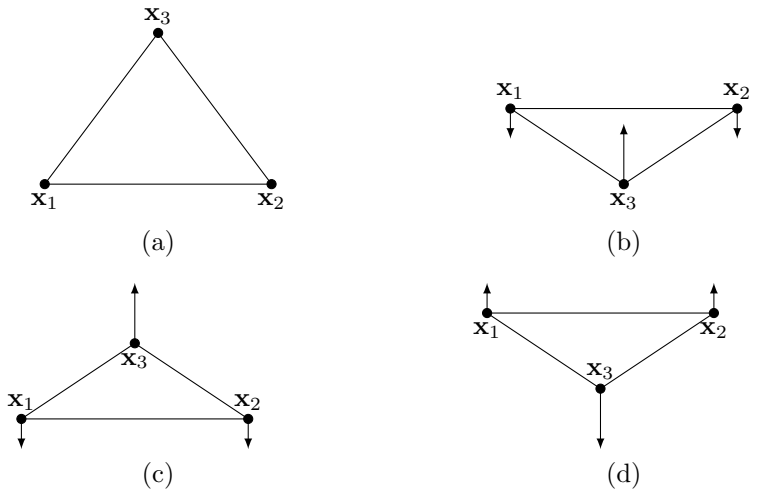


Figure 6.2: This figure shows the error in the co-rotational force computation due to an improper rotation. (a) shows the original triangle. (b) shows the current deformation state $\mathbf{X}_e$, where the triangle is inverted. The arrows indicate the correct forces that should be computed in this state. (c) visualizes an intermediate step in the co-rotational force computation (Eq. (5.53): $\mathbf{F}_e^{rot} = -\mathbf{R}_e\mathbf{K}_e(\mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0)$), where the computation is decomposed in its single steps. It shows the transformed deformation state $\mathbf{R}_e^T\mathbf{X}_e$. The internal forces are computed w. r. t. this virtual state. As $\det(\mathbf{R}_{B_e}) = -1$, this virtual state is the reflection of the correct state seen in (b). Therefore, the triangle seems not to be inverted during force computation. The arrows depict the intermediate forces $\mathbf{F}_e^{temp} = -\mathbf{K}_e(\mathbf{R}_e^T\mathbf{X}_e - \mathbf{X}_e^0)$ that are caused by this virtual state. (d) illustrates the current deformation state of the triangle and the co-rotational forces $\mathbf{F}_e = \mathbf{R}_e\mathbf{F}_e^{temp}$, which are just the reflected intermediate forces shown in (c) and therefore act in the wrong direction.

Let $\hat{\mathbf{S}}_{B_e} = \mathbf{P}^T\mathbf{S}_{B_e}\mathbf{P}$ be the diagonalization of $\mathbf{S}_{B_e}$. To include a reflection in $\mathbf{S}_{B_e}$, either one or three of the diagonal entries of $\hat{\mathbf{S}}_{B_e}$ can be inverted which results in some $\hat{\mathbf{S}'}_{B_e}$ with $\det(\hat{\mathbf{S}'}_{B_e}) < 0$, and $\mathbf{S}_{B_e}$ can be redefined as $\mathbf{S}_{B_e} := \mathbf{P}\hat{\mathbf{S}}'_{B_e}\mathbf{P}^T$, computing $\mathbf{R}_{B_e} = \mathbf{B}_e\mathbf{S}_{B_e}^{-1}$ afterwards. Ideally, the direction in which the tetrahedron got inverted can be identified and the corresponding entry of $\hat{\mathbf{S}}_{B_e}$ is chosen to be negative, because inverting another one would lead to force discontinuities. As this is not possible by just looking at the current state of the tetrahedron, some heuristic assumption has to be made about the current deformation.

6.2.2 Existing approach

[ITF04] assume that a tetrahedron is as uninverted as possible, which implies two aspects: Only one component of $\hat{\mathbf{S}}_{B_e}$ should be chosen to be negative, and

this component should correspond to the direction that causes minimum movement to uninvert the tetrahedron in this direction. They use the singular value decomposition (SVD) $\mathbf{B}_e = \mathbf{U}_{B_e}\hat{\mathbf{S}}_{B_e}\mathbf{V}_{B_e}^T$, where $\mathbf{U}_{B_e}$ and $\mathbf{V}_{B_e}$ are orthogonal matrices, to diagonalize the deformation gradient. This leads to the same diagonal matrix as the polar decomposition (see Sec. 6.2.5), where they choose the smallest diagonal entry of $\hat{\mathbf{S}}_{B_e}$ which should correspond to the desired direction.

6.2.3 Discussion of the existing approach

The choice of [ITF04] seems to be founded in [Kan94], where the minimization of the quadratic form

$$\sum_{i=0}^{3} ||\mathbf{x}_{i,rel} - \mathbf{R}\mathbf{x}_{i,rel}^0||^2 \tag{6.3}$$

among all rotations $\mathbf{R}$ is considered. Remember that $\mathbf{x}_{i,rel}$ was set to $\mathbf{x}_{i,rel} = \mathbf{x}_i - \mathbf{x}_{cm}$ to be the coordinates relative to the center of mass. They show that the optimal rotation can be extracted from the matrix $\mathbf{A}_{xx^0} := \sum \mathbf{x}_{i,rel}(\mathbf{x}_{i,rel}^0)^T$ by the polar decomposition $\mathbf{A}_{xx^0} = \mathbf{R}_{xx^0}\mathbf{S}_{xx^0}$ in the case $\det(\mathbf{A}_{xx^0}) > 0$. If $\det(\mathbf{A}_{xx^0}) < 0$, they show that one has to compute the diagonalization $\hat{\mathbf{S}}_{xx^0}$ of the symmetric part $\mathbf{S}_{xx^0}$ of the polar decomposition and indeed invert the smallest diagonal entry to obtain the optimal rotation $\mathbf{R}_{xx^0}$, like it is proposed by [ITF04] for the deformation gradient.

However, $\mathbf{A}_{xx^0}$ generally does not equal the deformation gradient $\mathbf{B}_e$. Although they differ only by a symmetric matrix (see [MHTG05]), e. g. $\mathbf{B}_e = \mathbf{A}_{xx^0} \cdot \mathbf{A}_{sym}$, their polar decompositions return different rotations. This is founded by the fact that the product of two symmetric matrices is symmetric if and only if both matrices are simultaneously diagonalizable. Therefore, $\mathbf{S}_{xx^0} \cdot \mathbf{A}_{sym}$ is not symmetric in general and $\mathbf{B}_e = \mathbf{R}_{xx^0} \cdot (\mathbf{S}_{xx^0} \cdot \mathbf{A}_{sym})$ is not equal to the polar decomposition $\mathbf{R}_{B_e}\mathbf{S}_{B_e}$ of $\mathbf{B}_e$. It follows that the inversion of the smallest diagonal entry of $\hat{\mathbf{S}}_{B_e}$ does not necessarily minimize (6.3), which seemed to be the motivation for the choice of [ITF04], and that the minimization of (6.3) is not the goal of the co-rotational model, as $\mathbf{B}_e \neq \mathbf{A}_{xx^0}$ in general.

Further, choosing the smallest diagonal entry does not necessarily fulfill the heuristic assumption as it does not always correspond to the direction that causes minimum movement to uninvert the tetrahedron. This is easily seen by the two-dimensional example in Fig. 6.3 which could analogously be performed in $\mathbb{R}^3$. The upper triangle shows its resting state that has small extension h in y-direction. The lower one shows its current deformation state. It is inverted in y-direction and compressed in x-direction, so the deformation gradient $\mathbf{B}_e$ is already diagonal. As it has the same y-height h as in the resting state, the entry corresponding to this direction is -1, while the entry in x-direction is a little smaller than one, say 0.9. Then, the deformation gradient $\mathbf{B}_e$ would be $\mathbf{B}_e = \left(\begin{smallmatrix} 0.9 & 0 \\ 0 & -1 \end{smallmatrix}\right)$ and the symmetric part $\mathbf{S}_{B_e} = \left(\begin{smallmatrix} 0.9 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. Hence, as the entry corresponding to the x-direction is smaller, the triangle is chosen to reinvert along

x-direction. This clearly contradicts the heuristic assumption. The problem is that the smallest diagonal entry does not reflect the distance of any vertex to its opposite face, because it obviously depends on the resting state. The flatter the triangle is, the more probable it is that this rule chooses a non-intuitive component.

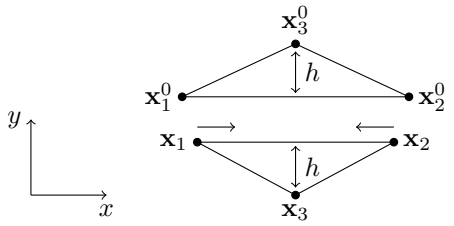


Figure 6.3: This figure illustrates a non-intuitive choice of the inversion direction. The upper triangle shows the resting state, and the lower one shows the current deformation. The triangle is deformed such that it is inverted in y-direction and slightly compressed in x-direction. The entry of $\mathbf{B}_e$ corresponding to the y-direction is -1, because it is inverted and has the same height h as in the initial state. Hence, the entry of $\mathbf{S}_{B_e}$ corresponding to the y-direction is 1. The entry in x-direction is smaller than 1 and therefore, it is chosen as the inversion direction. Consequently, the triangle gets uninverted in x-direction as indicated by the arrows. However, according to the heuristic assumption, obviously the y-direction should be chosen.

6.2.4 Improved inversion handling approach

The novel approach is also based on the heuristic assumption of [ITF04] which states that the tetrahedron is as uninverted as possible. This assumption demands that the direction that causes minimum movement to uninvert the tetrahedron is located. As this direction does not necessarily correspond to the smallest entry in $\mathbf{S}_{B_e}$, which is assumed in [ITF04], there is some more work to do. First, the direction to reinvert the tetrahedron should be the direction in which one of the vertices has the shortest distance to its opposite face. However, note that "distance" must not be interpreted as the standard orthogonal distance, but it rather means "distance along some given direction". The reason is that $\hat{\mathbf{S}}_{B_e}$ restricts the choice to three directions, which are given by the eigenvectors of $\mathbf{S}_{B_e}$. Taking the orthogonal direction would not indicate which diagonal entry to invert. Hence, only three predetermined directions can be chosen as the inversion direction, and the shortest distance along one of these directions has to be computed.

To determine the direction causing minimum movement, a pair $(\mathbf{c}, \mathbf{s})$ consisting of an eigenvector $\mathbf{s}$ of $\mathbf{S}_{B_e}$ with $||\mathbf{s}|| = 1$ and a vertex $\mathbf{c}$ has to be found such that the distance from $\mathbf{c}$ along $\mathbf{s}$ to its opposite face $\mathbf{F}_c$ is minimized among all possible pairs (Fig. 6.4). This is done by computing a parameter $\lambda_{c,s}$ for each

pair $(\mathbf{c}, \mathbf{s})$ with $||\mathbf{s}|| = 1$ such that $\mathbf{x}_c + \lambda_{c,s} \cdot \mathbf{s}$ lies on $\mathbf{F}_c$, where $\mathbf{x}_c$ denotes the position of $\mathbf{c}$ (Fig. 6.5). Clearly, $|\lambda_{c,s}|$ then denotes the distance of $\mathbf{c}$ along $\mathbf{s}$ to $\mathbf{F}_c$. Hence, the desired pair $(\mathbf{c}_0, \mathbf{s}_0)$ is the one that corresponds to the minimum $|\lambda_{c_0,s_0}|$.

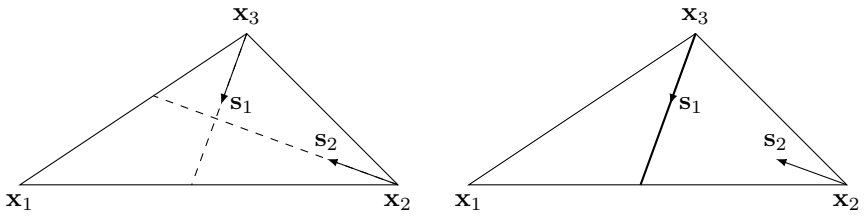


Figure 6.4: (a) The approach looks for the shortest distance of a vertex $\mathbf{c}$ to its opposite face along the given eigenvectors of $\mathbf{S}_{B_e}$. (b) In this case, $\mathbf{x}_3$ has the shortest distance along the eigenvector $\mathbf{s}_1$.

For the computation of $\lambda_{c,s}$, it has to be considered that the eigenvectors of $\mathbf{S}_{B_e}$ are related to the unrotated coordinate frame, while the current deformation state contains a rotation. Remembering (6.2), it can be seen that the co-rotational forces can be computed equivalently with respect to the state $\mathbf{S}_e\mathbf{X}_e^0$. This state is related to the unrotated coordinate frame and obviously fits to the eigenvectors of $\mathbf{S}_{B_e}$. However, as shown in Sec. 6.1.2 it also does not contain the inversion and therefore, it causes erroneous forces. Inverting one of the diagonal entries of $\hat{\mathbf{S}}_{B_e}$ now is equivalent to inverting the reference state $\mathbf{S}_e\mathbf{X}_e^0$ in order to get the correct forces. $\mathbf{S}_e\mathbf{X}_e^0$ and the current deformation state $\mathbf{X}_e$ differ by translation, rotation and inversion. Since none of these changes any distance, looking for the direction that causes minimum movement to uninvert the deformed, rotated tetrahedron is equivalent to looking for the direction that causes minimum movement to invert the unrotated reference state $\mathbf{S}_e\mathbf{X}_e^0$. Therefore, the parameters $\lambda_{c,s}$ can be computed with respect to $\mathbf{S}_e\mathbf{X}_e^0$.

Now let $\mathbf{c}'$ be any vertex of the face $\mathbf{F}_c$ and $\mathbf{x}_{c'}^S$ its position in the reference state $\mathbf{S}_e\mathbf{X}_e^0$. E. g., $\mathbf{c}'$ can be one of the tetrahedron's vertices. Let $\mathbf{x}_c^S$ be the position of $\mathbf{c}$ in this state. As $\lambda_{c,s}\mathbf{s}$ and $(\mathbf{x}_{c'}^S - \mathbf{x}_c^S)$ are vectors that point from $\mathbf{c}$ to some point on the plane that contains $\mathbf{F}_c$, they have the same component along the face normal of $\mathbf{F}_c$ (Fig. 6.5). With $\mathbf{n}_c$ being the unit face normal of $\mathbf{F}_c$, this fact could be expressed by the dot product in the following equation.

$$\lambda_{c,s}\mathbf{s} \cdot \mathbf{n}_c = (\mathbf{x}_{c'}^S - \mathbf{x}_c^S) \cdot \mathbf{n}_c \tag{6.4}$$

Now it follows that $\lambda_{c,s}$ can be computed using

$$\lambda_{c,s} = \frac{(\mathbf{x}_{c'}^S - \mathbf{x}_c^S) \cdot \mathbf{n}_c}{\mathbf{s} \cdot \mathbf{n}_c}. \tag{6.5}$$

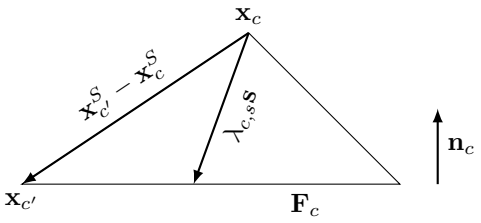


Figure 6.5: $\mathbf{x}^S_{c'} - \mathbf{x}^S_c$ and $\lambda_{c,s}\mathbf{s}$ have the same component along the face normal $\mathbf{n}_c$ of $\mathbf{F}_c$.

Choosing $\mathbf{s_0}$ according to the minimum $|\lambda_{c_0,s_0}|$ provides the direction with minimum movement to invert the reference state $\mathbf{S}_e\mathbf{X}^0_e$. As argued above, $\mathbf{s}_0$ also provides the direction causing minimum movement to uninvert the tetrahedron, and the sign of the diagonal entry that corresponds to $\mathbf{s}_0$ should be changed. Since the length of $\mathbf{n}_c$ cancels out in (6.5), the condition that it has to be a unit vector can be dropped.

Further, $\mathbf{c}_0$ is stored as the vertex that is seen as the one that caused the inversion by crossing $\mathbf{F}_{c_0}$. Until the tetrahedron returns to an uninverted state, the optimal direction $\mathbf{s}_0$ is computed with respect to $\mathbf{c}_0$, so that the tetrahedron gets uninverted by $\mathbf{c}_0$ crossing $\mathbf{F}_{c_0}$ again. Note that there is no possibility to store the computed direction $\mathbf{s}_0$, because the eigenvectors of $\mathbf{S}_{B_e}$ and therewith the possible inversion directions change in each time step. Hence, one has to refer to a vertex to achieve a consistent choice of inversion directions in subsequent iterations.

Although the approach has been introduced especially for the co-rotational formulation of the linear Finite Element Method, it can be implemented for arbitrary deformation models. The only condition is that the deformation gradient can be transformed into diagonal form. This is always possible using the SVD $\mathbf{B}_e = \mathbf{U}_{B_e}\hat{\mathbf{S}}_{B_e}\mathbf{V}^T_{B_e}$, where $\mathbf{U}_{B_e}$ can be interpreted as a rigid body rotation, and $\mathbf{V}_{B_e}$ as a material rotation (see [ITF04]). The columns of $\mathbf{V}_{B_e}$ can be interpreted as the possible inversion directions that are needed in the proposed approach. [ITF04] show how this can be generalized to anisotropic materials.

6.2.5 Degenerated tetrahedrons

As shown in Sec. 6.1, a tetrahedron is inverted if and only if $\det(\mathbf{B}_e) < 0$, whereas it is degenerated if and only if $\det(\mathbf{B}_e) = 0$. Therefore, inverted elements and degenerated elements are handled completely independent from each other. Together with an efficient handling of degenerated elements, this yields a stable simulation of arbitrary deformations.

Like inversion, problems with degenerated elements occur only in the co-rotational formulation. The problem is again located in the polar decomposition. In this section, it is described how a simple solution shown in [ITF04] that is based on SVD can be adopted to obtain a solution for the polar decomposition. The polar decomposition and the SVD of a deformation gradient $\mathbf{B}_e$ are con-

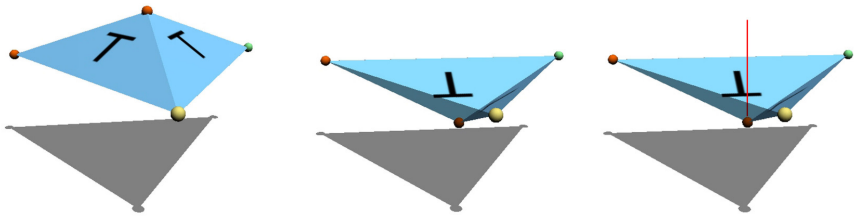
nected by $\mathbf{B}_e = \mathbf{R}_{B_e}\mathbf{S}_{B_e} = \mathbf{R}_{B_e}\mathbf{P}\hat{\mathbf{S}}_{B_e}\mathbf{P}^T =: \mathbf{U}\hat{\mathbf{S}}_{B_e}\mathbf{Q}^T$, which is the SVD of $\mathbf{B}_e$. If $\mathbf{B}_e$ is singular, then at least one of the diagonal entries of $\hat{\mathbf{S}}_{B_e}$ is zero. If it is exactly one entry, the two columns of $\mathbf{U}$ corresponding to the nonzero entries are uniquely determined. The third column then is computed as the cross product of the other two and therefore, it is unique, too. Hence $\mathbf{R}_{B_e} = \mathbf{U}\mathbf{Q}^T$ results in a unique solution for the polar decomposition. If more than one diagonal entry equals zero, the solution is no longer unique and a system of orthonormal columns for $\mathbf{U}$ can be chosen. In every case, $\mathbf{R}_{B_e}$ is guaranteed to be a proper rotation.

6.3 Results

In this section, several experiments are shown to illustrate how the improved inversion handling approach works. Remember the heuristic assumption that a tetrahedron always is as uninverted as possible. In Sec. 6.3.1, it is shown that the method chooses the direction corresponding to this assumption and that it guarantees fast and reliable recovery from inversion. In Sec. 6.3.2, it is shown that it can easily be implemented together with a stable handling of degenerated elements.

6.3.1 Recovery from inversion

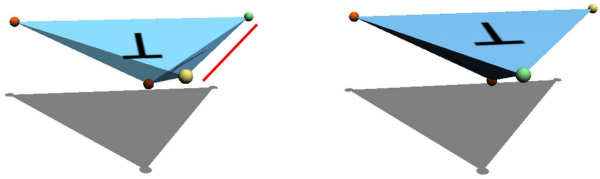
In the first experiment, a single tetrahedron is inverted manually. The resting state and the inverted state are shown in Fig. 6.6. After the manual inversion, the behavior of the tetrahedron is simulated starting with the inverted state. The inversion direction that is chosen by the existing approach is indicated by the red line in Fig. 6.7 (a). Compared to Fig. 6.6 (c), it can be seen that this choice is non-intuitive. Fig. 6.7 (b) illustrates the resting state that is reached using the existing approach.



(a) (b) (c)

Figure 6.6: This figure illustrates the setting of the first experiment. (a) shows the resting state of the tetrahedron. (b) shows an inverted state where the top vertex was moved below its opposite face. (c) depicts the expected inversion direction which is preferred by the heuristic assumption.

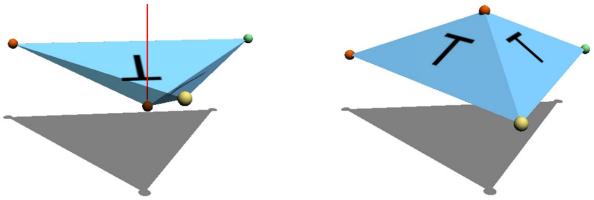
Fig. 6.8 illustrates that the tetrahedron is uninverted correctly by the improved inversion handling method. As the picture illustrates, the approach



(a) (b)

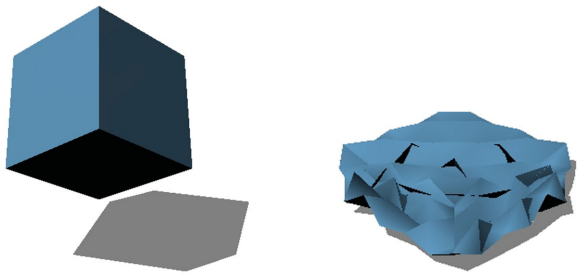
Figure 6.7: (a) illustrates the reinversion direction that is chosen by the existing approach. (b) shows the resting state after the existing approach is applied.

locates the correct direction and uninverts the tetrahedron as expected.



(a) (b)

Figure 6.8: (a) repeats the expected reinversion direction to illustrate that the improved approach restores the intuitive resting state which is depicted in (b).

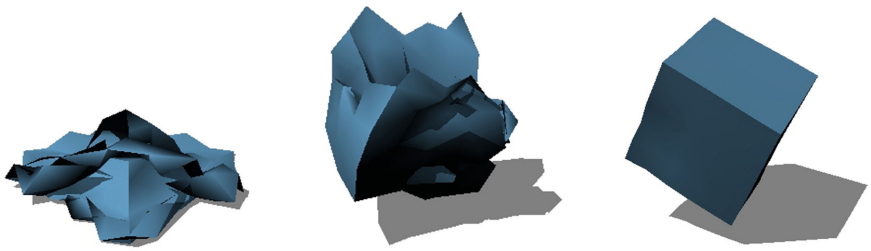


(a) (b)

Figure 6.9: (a) shows a cube that is falling down onto the ground. (b) If inversion is not handled correctly, the cube stays in an erroneous equilibrium state.

In the experiment shown in Fig. 6.9, 6.10 and 6.11, a cube falls down onto the ground (Fig. 6.9 (a)) which leads to the inversion of many tetrahedrons. This setting demonstrates that inversion handling is required: Fig. 6.9 (b) depicts what happens if the co-rotational Finite Element Method is applied without any inversion handling. Many tetrahedrons stay inverted due to the improper rotations, and as a consequence, the cube loses volume. Also, its energy gets lost in this example, and it does not bounce up any more.

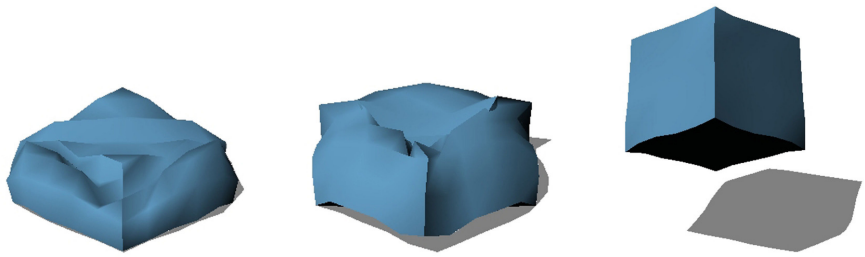
Moreover, the experiment shows that the chosen inversion direction significantly influences the behavior of an object: Fig. 6.10 illustrates a naive inversion handling, where just any diagonal entry is chosen as the inversion direction. The cube recovers to its original shape, but it takes a long time and it suffers from self-intersections during the recovery. Due to this fact, it loses energy and does not bounce as high as it should.



(a) (b) (c)

Figure 6.10: Naive inversion handling. (a) shows the maximum compression of the cube. (b) illustrates that there are self-intersections during inversion recovery. (c) shows that the cube restores its original shape.

In Fig. 6.11, the behavior of the falling cube is illustrated if the proposed approach is employed. First of all, it is not deformed as much as in the other two cases, because the inversion handling locates the correct inversion direction and therefore, the internal forces react faster. The original shape is restored after a few simulation steps, which shows the efficiency of the approach. Further, it can be seen that the inversion handling does not result in an artificial rotation of the object.



(a) (b) (c)

Figure 6.11: The improved inversion handling approach. Compared to Fig. 6.10, the original shape is restored in less simulation steps.

6.3.2 Handling degenerated elements

This section shows the combination of inversion handling with degenerated elements. A rubber cow is shrunk in a virtual sphere until the radius of the sphere is zero and all tetrahedrons are degenerated to a single point. After that, the spherical boundary is removed and the internal forces restore the shape of the cow. During the deformation and the recovery process, many inverted tetrahedrons have to be processed.

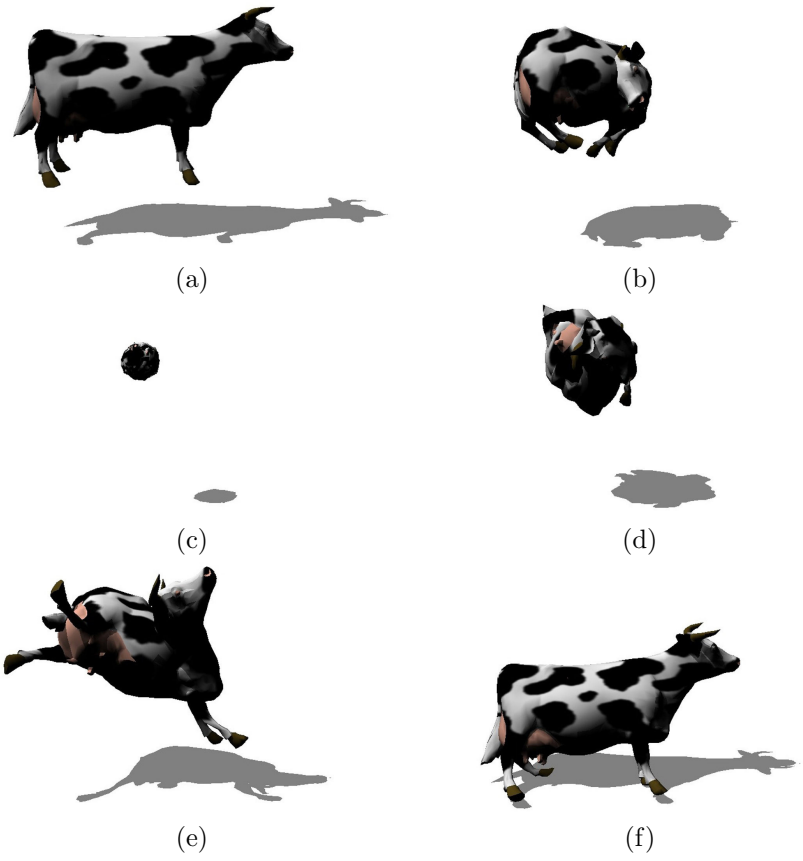


Figure 6.12: (a)-(c) A cow is shrunk by a virtual sphere. (d)-(f) After the sphere is removed, the cow restores its original shape.

As stated in Sec. 6.2.5, the rotation in the degenerated case is not unique when a tetrahedron is degenerated in more than one direction. Therefore, when all tetrahedrons are compressed to a single point, one has to choose any kind of rotation. Hence, it is possible that although the cow restores its correct shape, it does not find the correct rotation.

6.4 Conclusion

In this chapter, it has been shown that there is a strong need for efficient and stable inversion handling in algorithms that are based on a co-rotational Finite Element formulation. An existing approach has been reviewed, where a non-intuitive behavior has been observed in some configurations. Therefore, an improved method for inversion handling has been introduced that always chooses the most appropriate direction to uninvert an inverted tetrahedron. In contrast to existing approaches, the method can store the inverted component to guarantee a consistent processing of inverted elements in subsequent simulation steps. Thus, the approach improves the stability and robustness of Finite Element based deformable modeling approaches.

7 Parameter estimation

In Chapter 4, the acquisition of object models has been illustrated. After the point clouds have been transformed to appropriate structures like tetrahedral or triangular meshes, the models can be used within a simulation environment.

For an appropriate simulation of the objects, their geometric representation has to be equipped with proper physical parameters. For some applications like movies and computer games, it is sufficient to set parameters manually such that the visual appearance is satisfying. In other contexts such as virtual surgery, realistic deformation parameters are essential for the usability of simulation systems. A possible estimation approach exerts forces on an object and measures the deformation. Then, the force is transferred to the object's virtual counterpart, and the parameters are refined until the simulated deformation matches the measured one [ABB*08].

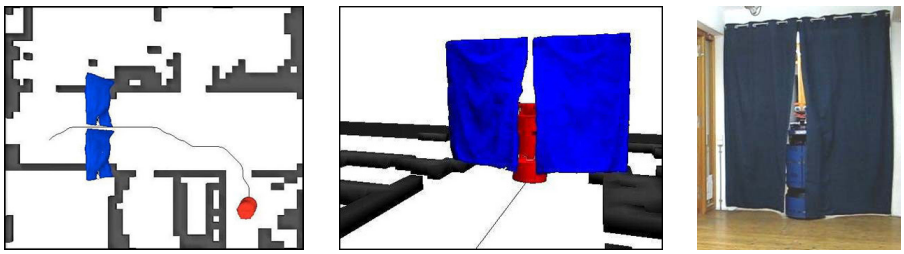
In Sec. 7.1, this principle is applied to the co-rotational Finite Element Method (Chapter 5), where the Young modulus and the Poisson ratio have to be determined (Sec. 5.1.3). The approach performs a series of indentation tests to obtain several samples of the force-displacement-relation. It employs a force-feedback-sensor to acquire the exerted forces, which are then transferred to the simulation environment. The deformation is measured by a range scanner, and the deformed surface is compared to the simulated object using the Iterative Closest Point algorithm [BM92]. The returned registration error is taken as an error function, which allows to apply a gradient descent to refine the deformation parameters. As the approach is based on a comparison of forces and displacements and does not use the deformation model explicitly, it is not limited to the co-rotational Finite Element Method and can be also applied to other deformation models.

Besides the deformation behavior, the visual appearance contributes to the realism of a simulation. Thus, also appropriate reflection parameters have to be found, either by manual adaption or by appropriate estimation algorithms. As for the deformation parameters, the illumination parameters depend on the underlying illumination model. A wide spread approach in interactive applications is the Phong illumination model [Pho75, AMHH08], as it approximates the illumination by local terms and therefore, it can be evaluated fast. In Sec. 7.2, an approach for the estimation of the Phong illumination parameters is outlined. The basic idea is to obtain several color samples for each surface point

from different directions, which allows to split the measured illumination into the separate terms of the Phong illumination model. The approach has been implemented in [Osw09] and is based on [WMP*06, Wey06], who built a rather complex environment to obtain the color samples. To account for simpler and more realistic hardware settings, the implementation of [Osw09] employs the ICP algorithm to correlate different color samples.

7.1 Deformation parameters

The approach for the estimation of deformation parameters has been developed in [FSS*10a, FSS*10b] with regard to the application described in Chapter 9, where a system for the autonomous navigation of mobile robots in environments with deformable objects is illustrated. Fig. 7.1 gives a short overview of the application. A mobile robot navigates from a starting position to a goal position crossing a set of curtains. While standard planning approaches concentrate on rigid environments, the approach in Chapter 9 takes the deformation energy into account and allows the robot to plan paths with a trade-off between deformation and travel cost. The travel cost is estimated by a simulation within the framework introduced in Chapter 3. In order to compute realistic deformation energies, the robot has to be able to acquire appropriate deformation parameters. Certainly, apart from the sketched application, the estimated parameters can be used within other tasks in the simulation environment.



(a) (b) (c)

Figure 7.1: Illustration of a planning algorithm in environments containing deformable objects. (a) A path through a pair of curtains with optimal trade-off between travel and deformation cost. The robot avoids a more expensive detour through the upper part. (b) The robot moving in the simulation environment to estimate the deformation energy. (c) The robot moving on the planned path.

In Sec. 7.1.1, the system setup is illustrated. Sec. 7.1.2 motivates that the comparison of displacements implicitly compares the estimated deformation parameters with the real ones, and thereby justifies the chosen error function. In Sec. 7.1.3, the gradient descent is outlined, before estimation results are shown in Sec. 7.1.4.

7.1.1 System setup

The system setup for the parameter estimation approach is shown in Fig. 7.2 (a). It illustrates the manipulator of a mobile robot which is equipped with a force-feedback-sensor and a wooden stick. The stick is necessary in order to minimize occlusions by the manipulator. The deformation is observed by a depth scanner that is also attached to the manipulator. Fig. 7.2 (b) illustrates the depth image obtained by a stereo camera. As the range scan contains parts of the robot's manipulator, a model of the robot's body is used to detect these points and to eliminate them from the depth scan.

(a) (b)

Figure 7.2: (a) The manipulator of a mobile robot is equipped with a wooden stick to perform indentation tests and a stereo camera to obtain a depth image. (b) The 3D image obtained by the stereo camera. Points that belong to the robot are eliminated.

With this setting, the robot approaches the object and increases the applied force step by step to a maximum of $50N$ or until the end-effector is moved farther than $10cm$. In each step t, the exerted force $\mathbf{F}_t$ is measured together with the corresponding depth image $\mathbf{X}_t \in \mathbb{R}^{3n}$ of the deformed surface. As in Sec. 3.3, $\mathbf{X}_t = (\mathbf{x}_{t,1}^T, \ldots, \mathbf{x}_{t,n}^T)^T$ denotes a vector that summarizes all individual positions $\mathbf{x}_{t,i}$ of the surface mesh. To ensure that the objects are deformed and not moved instead, they are assumed to be fixed, e. g. lying on a table or leaning at a wall.

7.1.2 Error function

Applying the measured force $\mathbf{F}_t$ in the simulation environment, the simulated deformed surface $\mathbf{X}_{s,t}(E,\nu) = (\mathbf{x}_{s,t,1}^T(E,\nu), \ldots, \mathbf{x}_{s,t,m}^T(E,\nu))^T \in \mathbb{R}^{3m}$ is obtained, which depends on the deformation parameters E and ν. Note that $m \neq n$ in general, as the number of vertices of $\mathbf{X}_{s,t}(E,\nu)$ corresponds to the number of vertices of the scan of the undeformed surface, while n is the number of vertices of the scan of the deformed object. Then, $\mathbf{X}_t$ is registered to $\mathbf{X}_{s,t}(E,\nu)$ using the

ICP algorithm [BM92] with point-to-point correspondences $\mathbf{x}_{t,i}^r \mapsto \mathbf{x}_{s,t,c(i)}(E,\nu)$, where $\mathbf{x}_{t,i}^r$ denotes the position of $\mathbf{x}_{t,i}$ after applying the ICP algorithm. A good initial configuration for the local registration can be found from the configuration of the manipulator of the robot. As the point density is high, the point-to-point correspondences lead to reasonable results. Further, this error metric does not require a triangular surface, but can directly work on the scanned point clouds. Then, the distance between the registered surfaces is defined similar to the error function in the ICP algorithm:

$$err(\mathbf{X}_{s,t}(E,\nu), \mathbf{X}_t^r) := \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_{t,i}^r - \mathbf{x}_{s,t,c(i)}(E,\nu)\|^2. \tag{7.1}$$

In the following, it is shown that this is a reasonable choice for an error function in order to obtain the deformation parameters. In other words, parameters that are close to the optimal parameters result in a smaller distance between the aligned surfaces than parameters that are farther away. To show this, the simulated displacement $\mathbf{Q}_{s,t}(E,\nu)$ is related to the displacement $\mathbf{Q}_t$ measured in reality. This is done by the force-displacement-relation in an equilibrium state (5.48).

$$\begin{aligned} \mathbf{K}_{s,t}(E,\nu)\mathbf{Q}_{s,t}(E,\nu) &= \mathbf{F}_{s,t} \\ \mathbf{K}_t\mathbf{Q}_t &= \mathbf{F}_t \end{aligned} \tag{7.2}$$

As $\mathbf{F}_{s,t} = \mathbf{F}_t$, inverting the equations and subtracting them from each other results in

$$\|\mathbf{Q}_{s,t}(E,\nu) - \mathbf{Q}_t\| = \|(\mathbf{K}_{s,t}(E,\nu)^{-1} - \mathbf{K}_t^{-1})\mathbf{F}_t\|. \tag{7.3}$$

For a fixed force $\mathbf{F}_t$, it can be seen that smaller differences between the simulated and measured displacements directly correspond to smaller deviations between the corresponding stiffness matrices. Thus, it is reasonable to compare the displacements in order to estimate the stiffness parameters. As the initial positions $\mathbf{X}_{s,t}^0$ and $\mathbf{X}_t^0$ are equal, the difference of the displacements $\mathbf{Q}_{s,t}(E,\nu) - \mathbf{Q}_t$ equals the difference of the absolute positions $\mathbf{X}_{s,t}(E,\nu) - \mathbf{X}_t$, and the latter can be compared as it is done in the error function (7.1).

In (7.3), the inversion of the stiffness matrices has to be justified. As illustrated in Sec. 5.2.1, six eigenvalues of the stiffness matrix $\mathbf{K}$ are zero, which belong to translations and (linearized) rotations. Thus, if $\mathbf{K} \in \mathbb{R}^{n\times n}$, the kernel $kern(\mathbf{K})$ has dimension 6, and the image space $im(\mathbf{K})$ has dimension $n-6$. As the kernel corresponds to translations and rotations, forces that cause translations or rotations do not lie in $im(\mathbf{K})$. However, as the object is fixed, the deformed surface is observed in an equilibrium state, and therefore, the exerted force $\mathbf{F}$ does neither contain a torque nor causes a translation. Thus, it lies in the image space of $\mathbf{K}$ and there is an inverse image $\mathbf{K}^{-1}\mathbf{F}$. Additionally, translations and rotations are eliminated by the registration algorithm, and hence, the

inverse image is forced to be orthogonal to the kernel of $\mathbf{K}$. Thus, the inverse image is unique, which justifies the relation $\mathbf{Q} = \mathbf{K}^{-1}\mathbf{F}$. The pseudo-inverse $\mathbf{K}^{-1}$ can be calculated by diagonalizing $\mathbf{K}$ and taking the reciprocal value of all eigenvalues that are greater than zero, while not changing the eigenvalues that are zero. This establishes a bijective mapping between $im(\mathbf{K})$ and $\mathbb{R}^n \setminus kern(\mathbf{K})$.

7.1.3 Gradient descent for parameter estimation

In this section, the gradient descent algorithm for the parameter estimation is introduced. First, it is experimentally shown that the error function is convex within a neighborhood of the correct parameters (Fig. 7.3 (b)), which states that the application of a gradient descent scheme leads to a minimization of the error function. Moreover, the experiment in Fig. 7.5 (b) shows that the error has a unique global minimum and no other local minima. Thus, it is possible to apply a simple gradient descent scheme, which is illustrated in this section.

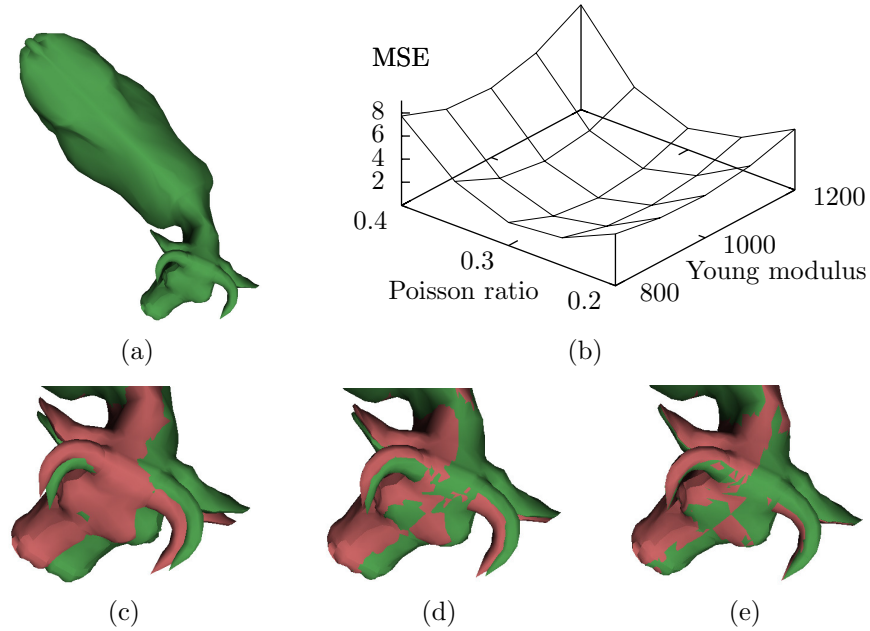


Figure 7.3: Illustration of the error function for a synthetic setting. (a) illustrates the setting of the experiment, where a cow with Young modulus $E = 1000\frac{N}{dm^2}$ and Poisson ratio $\nu = 0.3$ is deformed. (b) illustrates the error function which is convex in a neighborhood of the optimal parameters. (c,d,e) illustrate a detailed view of the registered models with $E = 3000, \nu = 0.3$ (c), $E = 2000, \nu = 0.3$ (d), and $E = 1300, \nu = 0.3$ (e). For better parameters, the registration error gets smaller.

The estimation algorithm is summarized in Alg. 2. In step 7, the gradient of the error function is needed. As the function is not given explicitly, this has

to be done numerically. Hence, the parameters are varied locally, which approximates the gradient. In step 11, the relative errors in subsequent iterations are compared. As the absolute error generally does not approximate zero, it is more appropriate to examine the difference $err_{i-1} - err_i$, which converges to zero if the error is minimized.

As the error function has a unique global minimum and no other local minima, the initialization of the values influences the runtime rather than the result. However, a reasonable initialization of E and ν can be deduced from the possible range of the parameters (Sec. 5.1.3) and the experimentally observed error functions (Sec. 7.1.4). The range of ν is bounded, i. e. $0 \leq \nu \leq 0.5$, and ν can be initialized with 0.25, for example. As $E \geq 0$, it can be initialized with any positive value. The experiments illustrate that the error function is steeper for small values, while it converges for larger values of E and thus, the slope converges to zero. Hence, it is probably better to underestimate the Young modulus than to overestimate it, and to initialize E with a small value. Note that the formulation of Hayes [HKHM72] cannot be applied for the initialization in this case as it requires two simultaneous indentations [CZ05].

Algorithm 2: Parameter estimation

Input: A model $\mathbf{X}^0$ of an object, exerted force $\mathbf{F}_t$, scanned surface $\mathbf{X}_t$
Output: Deformation parameters E, ν

1 Initialize E_0, ν_0;
2 $err_0 = \infty$;
3 $i = 0$;
4 Compute $\mathbf{X}_{s,t}(E_0, \nu_0)$;
5 **repeat**
6 $\quad i$++;
7 $\quad (E_i, \nu_i)^T = (E_{i-1}, \nu_{i-1})^T - \lambda \nabla err(\mathbf{X}_{s,t}(E_{i-1}, \nu_{i-1}), \mathbf{X}_t^r)$;
8 $\quad$ Compute $\mathbf{X}_{s,t}(E_i, \nu_i)$;
9 $\quad ICP(\mathbf{X}_t, \mathbf{X}_{s,t}(E_i, \nu_i))$;
10 $\quad err_i = err(\mathbf{X}_{s,t}(E_i, \nu_i), \mathbf{X}_t^r)$;
11 **until** $err_{i-1} - err_i < \delta$;

7.1.4 Results

Several experiments have been performed to illustrate the capabilities of the approach. Concerning the spatial representation in the simulation environment, there are different possibilities. If a complete object model is available (Fig. 4.7), this is used in the simulation. However, as the objects are assumed to be fixed by a wall or a table, it is not always possible to obtain scans from all directions for a complete model. As the wall or a table obviously delimit the object, they can be used to extract a planar surface that is heuristically assumed to be the back side of the object. Thus, a model of the object can be obtained by one scan only (Fig. 7.4).

(a) (b)

Figure 7.4: A tetrahedral mesh generated from a single incomplete scan of an inflated balloon by assuming the wall to be the back side of the ball. Although the scan suffers from large holes, the approach of [SWT06] generates an applicable tetrahedral mesh.

In the first experiment, the elasticity parameters of a foam cube with an edge length of $15cm$ were determined. The robot deformed the object in the center. The force-distance curve of this experiment is shown in Fig. 7.5 (a). In this experiment, the depth scan was obtained by a PMD time-of-flight camera. The robot moved for $9cm$ and collected a force measurement and a surface scan every $1cm$. As it can be seen in the curve, the deformation behavior of the cube is approximately linear, except in the beginning, where slippage of the probe tip occurred. This shows that the material assumptions are reasonable. Fig. 7.5 (b) illustrates the error function for one force-displacement sample with a uniform sampling of elasticity parameters. The error function is very high if the simulated object is too deformable, i. e. the Young modulus is too small. With increasing Young modulus, the simulated deformation gets smaller and the error converges to the registration error that is obtained when registering the scanned, deformed surface to the undeformed mesh. Furthermore, the Young modulus has a substantially larger influence on the error function than the Poisson ratio. Fig. 7.5 (c) shows the estimated Young modulus for different force-displacement samples. It shows that the variance of the estimated values is rather low among the different estimations. Fig. 7.6 shows a comparison of the deformed real surface and the deformed simulated surface, together with the registration result for a specific force-displacement sample.

In a second experiment, the approach was applied to the inflatable balloon that is shown in Fig. 7.2 (b). In this experiment, a Bumblebee stereo camera was used. First, a model of the undeformed object was constructed on the fly for the simulation system, as shown in Fig. 7.4. Second, the ICP algorithm was used to obtain the alignment of the deformed model with the observation of the deformed object. The resulting estimate for the Young modulus with seven different force-displacement samples is shown in Fig. 7.7. The estimation results in similar values for the Young modulus, indicating a homogeneous deformation behavior of the object. The average runtime for computing the optimal parameters per force-displacement sample was $192.5s$ and needed 7 iterations

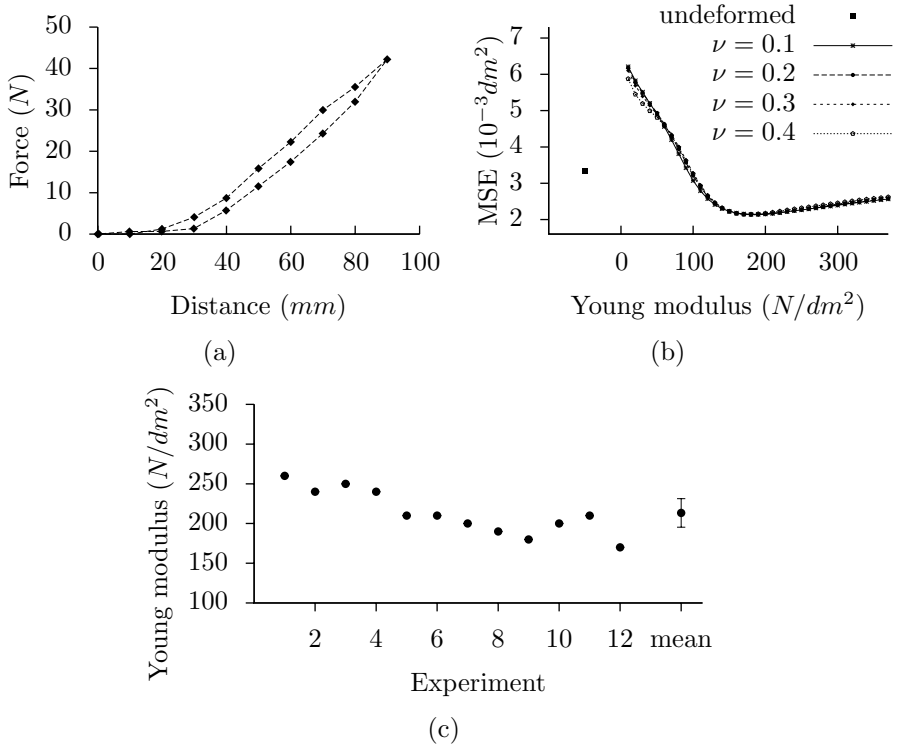
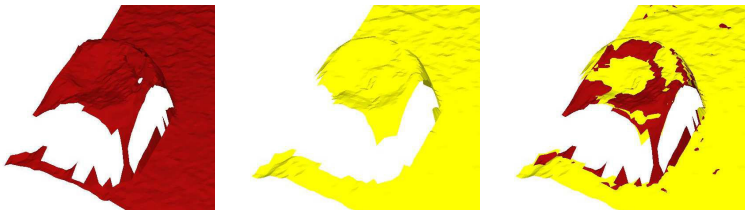


Figure 7.5: Parameter estimation for a foam cube. (a) illustrates the force depending on the distance that the manipulator moved. (b) shows the error function for one force-displacement sample. The influence of the Young modulus is much larger than the influence of the Poisson ratio. (c) illustrates the estimated Young modulus for different force-displacement samples.

of deformation simulations with different parameters on average.

In order to evaluate the robustness of the parameter estimation, the approach was applied to a plush teddy, which was deformed at six different body parts, e. g. the head, the belly, and the back. The results are summarized in Fig. 7.8. In each experiment, the parameters were determined as the average over seven different force-displacement samples. Additionally, the mean over the six different experiments is shown. The variance among the different experiments is higher than the variance among different force-deformation samples for the same location, which suggests that the assumption of homogeneous material is not valid in this case.



(a) (b) (c)

Figure 7.6: The registration result in one experiment of the parameter estimation. (a) shows the scanned, real surface, (b) shows the simulated deformed surface, and (c) shows the registration result.

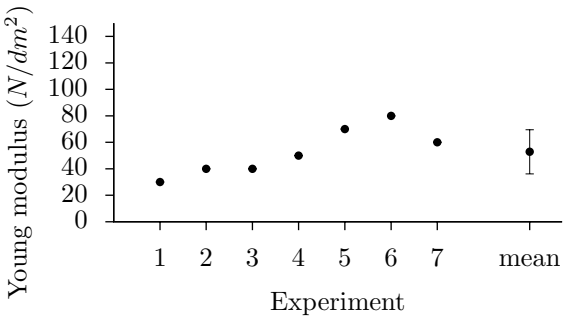
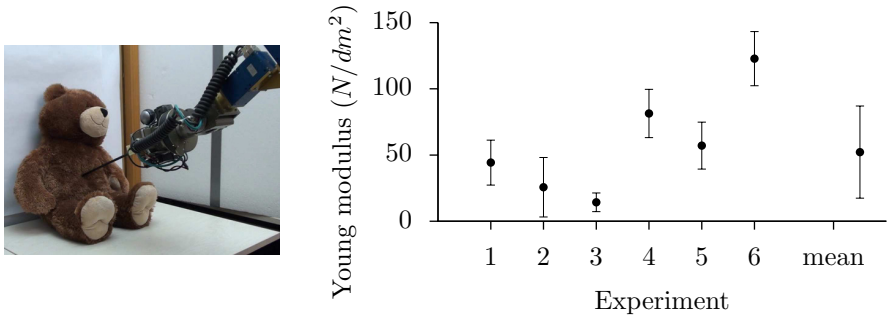


Figure 7.7: Parameter estimation for the inflated balloon (Fig. 7.2 (b)) for different force-displacement samples.



(a) (b)

Figure 7.8: Estimation of the deformation parameters for a plush teddy at different body parts. Experiment $1, 2, 3$ correspond to the head, $4, 5$ correspond to the belly, and 6 corresponds to the back of the teddy. In each experiment, the average over several force-displacement samples was computed.

7.2 Illumination parameters

In Chapter 4, the registration of several 3D scans has been illustrated. For the visualization of the scanned objects, the employed face*SCAN* III system also takes a color snapshot of the objects. However, as a complete object is composed of different scans, the color values of overlapping parts are taken several times from different directions. Thus, they do not match and cannot be used for the rendering of the object (Fig. 7.9). To overcome this limit, the reflection values of objects are estimated instead of color values, and are applied with an appropriate illumination model in order to obtain a proper visualization.

Figure 7.9: The color values taken by the scanning devices do not fit at overlapping parts, as they are taken from different directions [Osw09].

In this section, an approach for the estimation of the illumination parameters is illustrated. The approach has been implemented in [Osw09] and is based on previous work by [WMP*06, JMLH01, DJ05]. While [WMP*06, Wey06] use a rather complex scanning environment, [Osw09] use a simplified setting which employs the ICP algorithm to find correspondences between different scans. Thus, this version better meets a realistic hardware setting. In this section, the idea of the estimation implemented by [Osw09] is outlined.

7.2.1 Illumination model

The parameters that should be estimated depend on the chosen illumination model. In Computer Graphics, the Phong illumination model [Pho75, AMHH08] is widely used due to its simplicity and low computation time. Therefore, it is also employed in *DefCol Studio*. To illustrate the corresponding parameters, it is briefly introduced here. It evaluates the color value only locally at the surface point and disregards reflections between the objects. These are only approximated by the ambient value.

Fig. 7.10 illustrates the components of the Phong illumination model. The color value is composed of three individual values. First, the *ambient* term is considered, which approximates the global ambient illumination that is basically

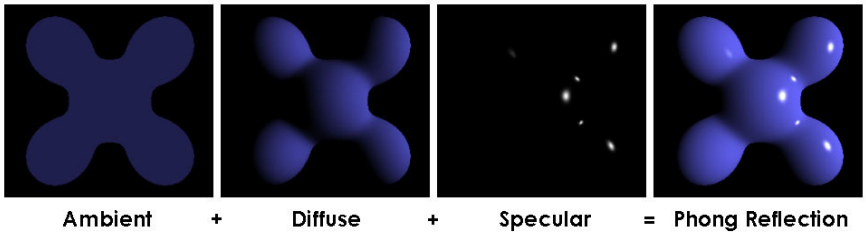


Figure 7.10: Illustration of the different components of the Phong illumination model.[1]

caused by the reflected light of all the objects in a scene. Second, a *diffuse* reflection term is considered which represents the color of the object, i. e. the colors that are reflected. This component does not depend on the viewing direction, but on the angle between the surface normal and the incoming light. Thus, it produces a spatial impression. Third, light is partially mirrored into the ideal reflection direction, which is represented by the *specular* term. Thus, the color value is given as

$$I = I_{ambient} + I_{diffuse} + I_{specular} \tag{7.4}$$

The ambient value is computed as

$$I_{ambient} = k_a \cdot I_a, \tag{7.5}$$

where I_a denotes the ambient intensity of the incoming light and k_a denotes a material constant determining the amount of reflected ambient light. In order to represent the reflected light of the objects in the scene, I_a is frequently chosen to be similar to the dominating color of the objects within the scene.

The diffuse value depends on the cosine of the angle between the direction of the light source $\mathbf{L}$ and the surface normal $\mathbf{N}$ (Fig. 7.11). If $\mathbf{L}$ and $\mathbf{N}$ are unit vectors, the diffuse component is computed as

$$I_{diffuse} = k_d \cdot I_d \cdot \cos(\angle(\mathbf{L}, \mathbf{N})) = k_d \cdot I_d \cdot (\mathbf{L} \cdot \mathbf{N}), \tag{7.6}$$

where I_d denotes the diffuse intensity of the light source and k_d denotes the diffuse reflection parameter of the surface.

The specular component depends on the ideal reflection direction and the direction of the viewer. It is given as

$$I_{specular} = k_s \cdot I_s \cdot \cos(\angle(\mathbf{R}, \mathbf{V}))^n = k_s \cdot I_s \cdot (\mathbf{R} \cdot \mathbf{V})^n, \tag{7.7}$$

[1] http://de.wikipedia.org/wiki/Phong-Beleuchtungsmodell, accessed on November 7, 2011.

where I_s denotes the specular intensity of the light source, k_s denotes the specular reflection parameter of the object, and n denotes a constant that determines the width of the specular highlight.

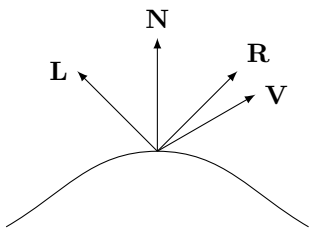


Figure 7.11: Directions that are employed in the Phong illumination model. **L** denotes the direction to the light source, **N** denotes the surface normal, **R** denotes the ideal reflection direction, and **V** denotes the direction of the viewer. All vectors point off the surface and are assumed to have unit length.

The given equations (7.5)-(7.7) are only valid for a single color. Thus, for the standard rgb-model, the equations have to be evaluated separately for the three color values.

Typically, the diffuse and ambient reflection coefficients k_d^i, k_a^i can be chosen as $k_d^i = k_a^i$ for all $i \in \{r, g, b\}$, as they represent the color of an object. Moreover, the specular and diffuse intensity of the light source I_s^i, I_d^i are frequently assumed to be equal, as they originate from the same light source, while the components I_a^i are chosen to reflect the dominating color of the objects within the scene. The specular reflection components k_s^i are frequently chosen to reflect the incoming light value, thus $k_s^r = k_s^g = k_s^b$, which is intuitive as it represents an ideally mirroring component.

Thus, mainly the constants $k_d^r, k_d^g, k_d^b, k_s^r, k_s^g, k_s^b$ and n have to be estimated for an appropriate illumination of objects.

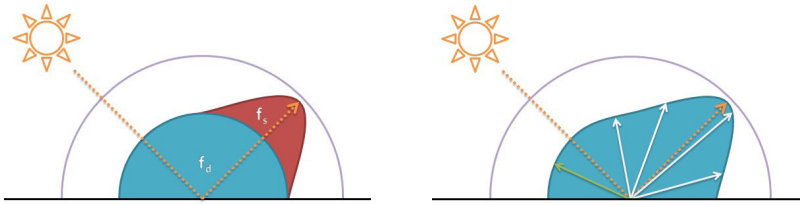
7.2.2 Estimation idea for illumination parameters

In order to estimate the diffuse and specular reflection constants, the ambient light should be eliminated. By eliminating all surrounding light sources, only the scanner emits light. This can be assumed to be directional. Moreover, in a darkly colored environment, there are almost no reflections in the neighborhood that concern the scanned object. In any case, the illumination is by far dominated by the diffuse and specular components. Thus, the color values obtained at a surface point are given as

$$I = I_{diffuse} + I_{specular}. \tag{7.8}$$

A simple reflection distribution given by diffuse and specular reflectance only is illustrated in Fig. 7.12 (a). As the specular reflection occurs only under

certain directions, there are directions which reflect exactly the diffuse component. Thus, depending on the direction of the camera, a snapshot contains always the diffuse part and possibly any fraction of the specular reflection. However, there are directions which contain only the diffuse lighting (green arrow in Fig. 7.12 (b)).



(a) (b)

Figure 7.12: (a) Illustration of the diffuse and specular reflection components. (b) As the specular component occurs only near the perfect reflection direction, there are directions that contain only the diffuse term (green arrow). The number of sample values (white / green arrows) determines the approximation quality of the reflection function [Osw09].

Thus, the basic idea for the parameter estimation is to separate the diffuse and specular reflection by taking several snapshots from different directions (Fig. 7.13). Using the ICP algorithm, the points of each scan are assigned to their corresponding points in the other scans. Thus, for each surface point, several sample values $I_1, \ldots, I_j$ for the light intensity I are obtained.

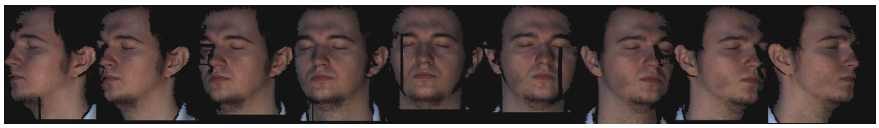


Figure 7.13: Several snapshots are taken in order to obtain different sample values for each surface point [Osw09].

As the diffuse component does not depend on the viewing direction, the quotient

$$\frac{I_{diffuse}}{\mathbf{L} \cdot \mathbf{N}} = k_d I_d \tag{7.9}$$

is constant for each surface point, while the quotient

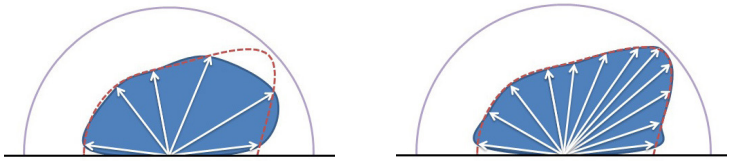
$$\frac{I_{specular}}{\mathbf{L} \cdot \mathbf{N}} \geq 0 \tag{7.10}$$

is always greater than zero. Thus, the minimum quotient of all sample values

$$I_{min} := \min \frac{I_j}{\mathbf{L}_j \cdot \mathbf{N}_j} \tag{7.11}$$

reflects a direction where the specular reflectance is zero. Therefore, it equals the diffuse quotient in (7.9), and the diffuse reflection coefficient can be calculated by dividing (7.11) through the light intensity of the scanner. The intensity and the light direction $\mathbf{L}_j$ are given by the scanner setup, while the surface normal $\mathbf{N}_j$ is returned within the 3D model that is obtained by the scanner.

Having separated the diffuse reflection, the specular part of a sample value can be obtained by the difference $I_j - I_{min}(\mathbf{L}_j \cdot \mathbf{N}_j)$. As this part is more sensitive to directional changes, it is much more sensitive to the sampling rate than the diffuse value as can be seen in Fig. 7.14.



(a) (b)

Figure 7.14: (a) With few sample values, the specular component cannot be approximated well, while the diffuse component can still be estimated. (b) More samples allow a better approximation of the specular component [Osw09].

Further, one needs to make assumptions on the specular exponent n which is based on experimental values in [WMP*06, Osw09]. However, as the diffuse value is more important for the visualization and errors in the specular component are more tolerable, this is an acceptable limitation of the implementation.

Fig. 7.15 compares the taken snapshots with the parameters obtained in [Osw09].

Figure 7.15: Comparison between the original color values obtained by the scanner and the illumination obtained by the estimated parameters [Osw09].

7.3 Conclusion

In this chapter, the estimation of deformation and illumination parameters has been introduced. The acquisition of physical parameters relies on the linear elasticity theory (Sec. 5.1) and estimates the Young modulus and the Poisson ratio, which define the behavior of linearly elastic, isotropic materials. Therefore, the approach employed a force-feedback-sensor and a depth camera to establish several force-displacement samples which lead to physical parameters using the force-displacement-relation. The used materials were assumed to be homogeneous, which showed to be correct for some of the experiments, but it did not apply to the experiment shown in Fig. 7.8. However, the Finite Element Method allows to assign distinct values of Young modulus and Poisson ratio to individual tetrahedrons. Thus, the approach could also be extended to inhomogeneous materials. For the estimation of illumination parameters, an approach implemented in [Osw09] has been outlined. The approach employs registration techniques to obtain various illumination samples from different directions, which allow to estimate the parameters of the Phong illumination model. Due to the application of registration techniques, it can be realized within simple hardware settings.

8

Stability in dynamic simulations

In Sec. 3.3, the computation of the dynamic movement of deformable objects has been introduced, which is based on a numerical time integration scheme. Obviously, the efficiency of a dynamic simulation depends directly on the choice of an appropriate time step. While the simulation gets faster for larger time steps, the approximation gets worse and the simulation becomes unstable. In a demonstrative explanation, this means that the total energy is not conserved, but grows uncontrolled, which causes the breakdown of the simulation. In contrast, small time steps yield a better approximation, however at the price of a slow simulation result.

Thus, methods that allow for a large time step while keeping the simulation stable greatly enhance the efficiency of simulations. An approach which is commonly applied in simulation environments is to introduce velocity-dependent damping forces. They are inspired by the idea to prevent an unbounded growth of the kinetic energy, as they reduce local oscillations of objects. Therefore, damping forces enhance both stability and realism, while requiring low computation time. Additionally, they can cover defects e. g. in mesh generation, as also badly shaped meshes can be simulated if an appropriate damping is employed (cf. Sec. 5.2.1, [She02]).

In Sec. 8.1, the basic concepts which describe the quality of an integration scheme are depicted. It is shown that velocity-dependent damping forces improve the numerical stability. In Sec. 8.2, a wide-spread damping approach for dynamic simulations is shown, which is frequently referred to as spring damping (cf. [NMK*06]). In Sec. 8.3, an optimized damping approach is introduced that is based on spring damping. It is motivated by the fact that the iterative application of spring damping further reduces local oscillations and in this way, it improves the stability. It is shown that the iterative forces can be computed directly, and that they allow for larger time steps in dynamic simulations. Further, the approach simplifies the parameter setting, as the damping parameter can always be chosen within the range between 0 and 1, and the damping forces cannot overshoot. In Sec. 8.3.7, the damping approach is applied for a fast propagation of external forces, which makes the simulation less sensitive to external forces like collision or user interaction.

8.1 Motivation

The applicability of a numerical integration scheme to a given differential equation is described by different terms which are consistency, stability, and convergence. As stability is the main interest in this chapter, the other topics are only depicted here, while a thorough introduction to these terms can be found in [DR08].

To illustrate the terms, the simple initial value problem

$$\dot{u}(t) = u(t), u(0) = u_0 = 1, t \in [0, T], T \in \mathbb{R} \tag{8.1}$$

is considered, which has the solution $u(t) = e^t$. The explicit Euler integration scheme with time step h leads to the discrete solution $u_{t+h} = u_t + h\dot{u}_t = (1 + h)u_t$. Thus, for $t = jh,\ j \in \{1, ..., \lfloor \frac{T}{h} \rfloor\}$ the value u_{jh} can be written as $u_{jh} = (1 + h)^j u_0 = (1 + h)^j$.

Consistency. The *local discretization error* is defined as

$$\left| \frac{u(t + h) - u'_{t+h}}{h} \right|, \tag{8.2}$$

where $u(t + h)$ is the correct solution at time $t + h$ and $u'_{t+h} := u(t) + h\dot{u}(t)$ is the approximated value obtained after one integration step, starting at the correct solution $u(t)$. This difference can be estimated by means of the Taylor series $u(t + h) = u(t) + h\dot{u}(t) + O(h^2) = u(t) + hu(t) + O(h^2)$, which results in

$$\left| \frac{u(t + h) - u'_{t+h}}{h} \right| = \left| \frac{(u(t) + hu(t) + O(h^2)) - (u(t) + hu(t))}{h} \right| = O(h). \tag{8.3}$$

With $h \to 0$, the local discretization error vanishes. Thus, the Euler integration scheme is called *consistent* for the given initial value problem. In other words, consistency means that the integration scheme sufficiently approximates the given problem.

Convergence. The *global discretization error* is defined as

$$\max_j |u(jh) - u_{jh}|, j \in \{1, ..., J = \lfloor \frac{T}{h} \rfloor\}. \tag{8.4}$$

An integration scheme is called *convergent* for a given initial value problem if the global approximation error vanishes for $h \to 0$. In the example, this is the case. To illustrate this, the convergence is shown for a fixed $t \in [0, T]$. With $t = jh$, it follows that $j \to \infty$ if $h \to 0$. j can be written as $j = t\frac{1}{h} = t\frac{J}{T}$. Then, it holds

$$(1+h)^j = \left(1+\underbrace{\frac{T}{J}}_{=:\frac{1}{n}}\right)^{\overset{=:n}{\frac{J}{T}}\cdot t} \widehat{=} \underbrace{\left(\left(1+\frac{1}{n}\right)^n\right)^t}_{\to e} \to e^t \text{ for } (h \to 0 \Rightarrow J \to \infty), \tag{8.5}$$

which shows that the Euler scheme converges for $h \to 0$. Thus, in contrast to consistency, which states the approximation of the differential equation, the convergence states the approximation of the solution function. For single-step integration schemes, consistency usually implies convergence.

Stability. As illustrated above, each integration step introduces a discretization error ε_t. Further, the initial value might contain an error ε_0 due to rounding. After one integration step, the error is evolved as $u_h = u_0 + \varepsilon_0 + h(u_0 + \varepsilon_0) = (1+h)u_0 + (1+h)\varepsilon_0$. For u_{jh}, it holds $u_{jh} = (1+h)^j u_0 + (1+h)^j \varepsilon_0$. Thus, after j simulation steps, the initial error ε_0 became $(1+h)^j\varepsilon_0$, and similarly, the discretization errors ε_t are evolved. As $(1+h)^j \to \infty$ for $j \to \infty$, it states that initial errors are always enlarged for $t \to \infty$. Note that this does not contradict the convergence, as convergence demands the pointwise convergence for a fixed t with $h \to 0$, whereas the stability is considered for $t \to \infty$ with a fixed $h > 0$.

Thus, the Euler scheme is consistent and convergent, but a small time step h has to be chosen. At the contrary, an interactive simulation has to trade off accuracy and efficiency. It requires a large time step which might result in an augmentation of initial errors. Therefore, the stability of an integration system is an important condition in interactive dynamic simulations. Note that for multi-step integration schemes, consistency on its own does not imply convergence, but in conjunction with stability, the convergence is implied, which highlights the significance of stability.

8.1.1 Benefit of damping schemes

The benefit of an appropriate damping scheme for the stability of dynamic simulations can be illustrated by a simple harmonic oscillator with mass m and spring stiffness D (Fig. 8.1).

The displacement s of the harmonic oscillator obeys the ordinary differential equation

$$m\ddot{s} = -Ds, \tag{8.6}$$

where m is the mass of the oscillator and D is the spring constant. With $v = \dot{s}$, this is transformed to

$$\begin{pmatrix}\dot{s}\\ \dot{v}\end{pmatrix} = \begin{pmatrix} v \\ -\frac{D}{m}s\end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{D}{m} & 0\end{pmatrix}\begin{pmatrix}s\\ v\end{pmatrix} \tag{8.7}$$

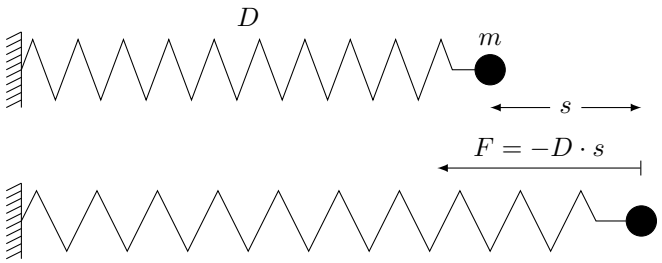


Figure 8.1: Illustration of a harmonic oscillator with spring constant D and mass m in its resting state (above) and deflected position (below).

Similar to (8.1), this can be solved with the exponential function for matrices:

$$\begin{pmatrix} s(t) \\ v(t) \end{pmatrix} = \exp\left(t \cdot \begin{pmatrix} 0 & 1 \\ -\frac{D}{m} & 0 \end{pmatrix} \right) \begin{pmatrix} s_0 \\ v_0 \end{pmatrix}. \tag{8.8}$$

Applying the explicit Euler scheme (Sec. 3.3) with time step h results in

$$\begin{pmatrix} s_{t+h} \\ v_{t+h} \end{pmatrix} = \begin{pmatrix} 1 & h \\ -h\frac{D}{m} & 1 \end{pmatrix} \begin{pmatrix} s_t \\ v_t \end{pmatrix}. \tag{8.9}$$

The structure of the solution is exactly the same as for (8.1), and the consistency and convergence can be shown similarly. Likewise, for an initial error $\begin{pmatrix} \varepsilon_s \\ \varepsilon_v \end{pmatrix}$, after j iterations it holds

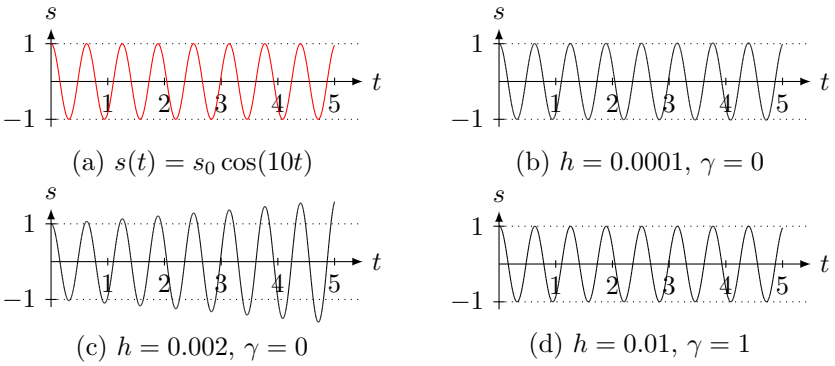
$$\begin{pmatrix} \varepsilon_s \\ \varepsilon_v \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & h \\ -h\frac{D}{m} & 1 \end{pmatrix}^j \begin{pmatrix} \varepsilon_s \\ \varepsilon_v \end{pmatrix}. \tag{8.10}$$

As the eigenvalues of the system matrix are $1 \pm ih\sqrt{\frac{D}{m}}$ with absolute value greater than 1, the initial error grows unbounded, and the Euler scheme is unstable. However, inserting a velocity-dependent damping force $-\gamma v$ leads to

$$\begin{pmatrix} s_{t+h} \\ v_{t+h} \end{pmatrix} = \begin{pmatrix} s_t \\ v_t \end{pmatrix} + \begin{pmatrix} hv_t \\ -h\frac{D}{m}s_t - h\gamma v_t \end{pmatrix} = \begin{pmatrix} 1 & h \\ -h\frac{D}{m} & 1 - h\gamma \end{pmatrix} \begin{pmatrix} s_t \\ v_t \end{pmatrix}. \tag{8.11}$$

For an appropriately chosen γ, which can be calculated exactly using the eigenvalues, the spectral radius of the system matrix gets smaller than one.

For $D = 100$, $m = 1$ and initial deflection $s_0 = 1$, the solution for (8.6) is given by $s(t) = s_0 \cos(10t)$. Fig. 8.2 illustrates the exact solution and discrete solutions obtained by the explicit Euler integration with different time steps and damping. While the approximation is sufficient for very small time steps (Fig. 8.2 (b)), it gets worse if the time step is larger (Fig. 8.2 (c)). With an appropriate damping constant, the time step can be chosen much larger (Fig. 8.2 (d)).



(a) $s(t) = s_0 \cos(10t)$ (b) $h = 0.0001$, $\gamma = 0$

(c) $h = 0.002$, $\gamma = 0$ (d) $h = 0.01$, $\gamma = 1$

Figure 8.2: Illustration of the explicit Euler scheme for the harmonic oscillator (8.6) with different time steps and damping. The constants are chosen as $D = 100, m = 1$. (a) illustrates the correct solution of (8.6). (b) shows that the Euler integration results in a sufficient approximation for very small time steps. (c) Using a larger time step, the integration error grows rapidly. (d) The velocity-dependent damping allows for a much larger time step.

8.2 Damping in dynamic simulations

In the previous section, the benefit of velocity-dependent damping forces was illustrated. They allow for significantly larger time steps in numerical integration schemes, which leads to fast and efficient simulations. Therefore, damping is a relevant topic in the area of computer animation. On the one hand, it enhances the stability, which improves the perceived performance of dynamic simulations. On the other hand, they influence the movement of objects. This can be desired, e. g. the reduction of oscillations, which improves the perceived realism of a simulation. However, it can also lead to undesired effects, e. g. if the global movement is affected and linear or angular momentum are concerned (Sec. 3.2.1).

The notation of this chapter is similar to Sec. 3.2. Thus, $\mathbf{x}_i, i = 1, \ldots, n$ denote the current positions of the mass points of a deformable object, $\mathbf{v}_i$ denotes the current velocity of mass point $\mathbf{x}_i$, and $\mathbf{q}_i$ denotes the displacement of $\mathbf{x}_i$. The scalar m_i denotes the mass of $\mathbf{x}_i$, and $\mathbf{f}_i$ denotes the force acting on $\mathbf{x}_i$, which is composed of internal and external forces $\mathbf{f}_i^{int}$ and $\mathbf{f}_i^{ext}$. Further, a capital letter $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T)^T$ denotes a column vector summarizing all positions, and similarly, $\mathbf{Q}$, $\mathbf{F}$ and $\mathbf{V}$ are defined. As $\dot{\mathbf{X}} = \dot{\mathbf{Q}}$, the equation of motion (3.9) can be written as

$$\mathbf{M}\ddot{\mathbf{Q}} = \mathbf{F}^{ext} + \mathbf{F}^{int} = \mathbf{F}^{ext} - \mathbf{K}\mathbf{Q}, \tag{8.12}$$

where $\mathbf{M}$ denotes the mass matrix, $\mathbf{F}^{ext}$ the external forces, and $\mathbf{K}$ denotes the stiffness matrix. Here, it does not matter if $\mathbf{K}$ stems from the Finite Element Method, the mass-spring-model, or any other deformation model. Disregarding

external forces and inverting $\mathbf{M}$, the equation has the same structure as the harmonic oscillator (Fig. 8.1). Accordingly, a velocity-dependent damping term is commonly inserted into (8.12).

$$\mathbf{M}\ddot{\mathbf{Q}} = \mathbf{F}^{ext} - \mathbf{K}\mathbf{Q} - \mathbf{C}\dot{\mathbf{X}}, \tag{8.13}$$

In this equation, $\mathbf{C}$ is a user-defined matrix that defines the velocity-dependent damping force. To decouple the system of equations, mass lumping is commonly assumed (Sec. 3.3) to get a diagonal mass matrix. Similarly, either the damping matrix has to be diagonal (see, for example, [MDM*02]), or the damping forces have to be computed in an additional step. The first alternative leads to the following equation of motion for a single mass point:

$$m_i\ddot{\mathbf{x}}_i = \mathbf{f}_i - \gamma\dot{\mathbf{x}}_i, \tag{8.14}$$

where γ is a user-defined parameter [TPBF87, TF88a]. The second alternative leads to the equation

$$m\ddot{\mathbf{x}} = \mathbf{f} + \mathbf{f}^d, \tag{8.15}$$

where $\mathbf{f}^d$ denotes a damping force that does not only depend on one, but on various mass points.

Eq. (8.14) corresponds to the damping term in (8.11). It is commonly known as point damping and only damps the absolute velocities of the points. In contrast, (8.15) allows a variety of possible computations of $\mathbf{f}^d$, including, for example, the damping of relative velocities. Of course, $\mathbf{f}^d$ generally depends on the damping parameter γ.

The objective of the damping configuration in dynamic simulations is to find a setting that maximizes the stability, while undesired effects are avoided or within an acceptable range. For example, the point damping force $\gamma\dot{\mathbf{x}}$ in (8.14) results in an improved numerical stability, but at the price of preventing the acceleration of a mass point. This behavior is desired in the specific context of friction and might be appropriate in some configurations like (8.11), but in general, the effect is undesired in an animation as it affects the linear momentum (Sec. 3.2.1).

Therefore, damping terms commonly employ relative velocities for the computation of $\mathbf{f}^d$ in (8.15) to avoid the undesired influence on the global movement. For example, the relative velocity of adjacent mass points can be used. In this case, damping forces are computed for pairs of points and are applied symmetrically to both points. This method preserves the linear momentum of the point pair and, thus, of the entire structure. For the conservation of angular momentum, a projection technique has to be applied. Then, this type of damping only influences relative movements of points, i. e. internal oscillations, while the global movement of the structure is not affected. However, estimating suitable damping parameters is a challenging task [KYOK09]. Compared to (8.11),

where the parameter could be determined by the eigenvalues of the system matrix, this is generally not possible in (8.13) and (8.15) as the matrices are large, which prevents an analytical analysis.

8.2.1 Spring damping approach

The optimized damping approach (Sec. 8.3) is based on the well-known spring damping which is briefly reviewed in this section (see e. g. [NMK*06]). In this approach, a force is symmetrically applied to two adjacent points in order to damp their relative velocity.

The spring damping approach needs some kind of neighborhood information for the mass points $\mathbf{x}_i$, and damping forces are computed for each pair of points that are considered to be adjacent. Neighborhood can be defined, for example, by an edge in a tetrahedral mesh, or by an influence radius of a particle. Each pair of adjacent points in the neighborhood information is referenced by the distance or relative position $\mathbf{x}_e = \mathbf{x}_i - \mathbf{x}_j$ of the incident points. The relative velocity of two adjacent points is denoted as $\mathbf{v}_e = \mathbf{v}_i - \mathbf{v}_j$. Similar to the damping forces at the points, the damping forces which are related to relative velocities are given by $\mathbf{f}_e^d$.

The spring damping approach works as follows. For two points $\mathbf{x}_i$ and $\mathbf{x}_j$, the relative velocity $\mathbf{v}_e$ is computed. A damping force $\mathbf{f}_e^d = -\gamma \mathbf{v}_e$, i. e. proportional to the relative velocity $\mathbf{v}_e$, is computed and symmetrically applied to $\mathbf{x}_i$ and $\mathbf{x}_j$. Since both damping forces add to zero, the linear momentum is preserved (Fig. 8.3).

However, the angular momentum is generally not preserved, as the relative velocity $\mathbf{v}_e$ is not necessarily aligned to the relative position $\mathbf{x}_e$, and nor are the damping forces $\mathbf{f}_e^d$. This issue can be addressed by projecting the forces onto the direction $\mathbf{x}_e$ using $\overline{\mathbf{f}_e^d} := (\mathbf{x}_e \cdot \mathbf{f}_e^d) \frac{\mathbf{x}_e}{\|\mathbf{x}_e\|_2^2}$ (Fig. 8.3 (b)).

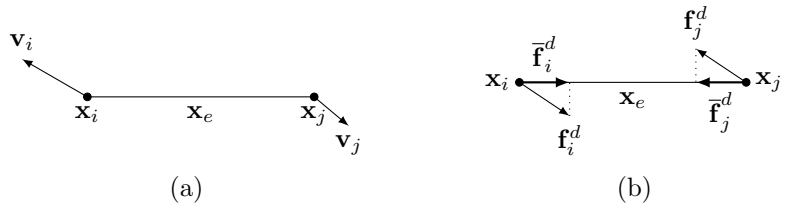


Figure 8.3: Illustration of the spring damping approach.

The result of the computed damping forces can be further improved if the future velocities $\mathbf{v}_i'$ and $\mathbf{v}_j'$ are predicted using an Euler integration step. In this case, the damping forces are computed with respect to the predicted future relative velocity $\mathbf{v}_e'$.

Fig. 8.4 illustrates the impact of damping in dynamic simulations. External forces are applied to the object using a spring dragger. While the undamped object suffers from local oscillations and the simulation finally fails (Fig. 8.4 (a)), the point-damped one comes to a resting state quickly, but it does not fall down

as the global movement is disturbed (Fig. 8.4 (b)). In contrast, the spring-damped sneaker (Fig. 8.4 (c)) behaves perfectly and keeps stable even with a significantly enlarged time step.

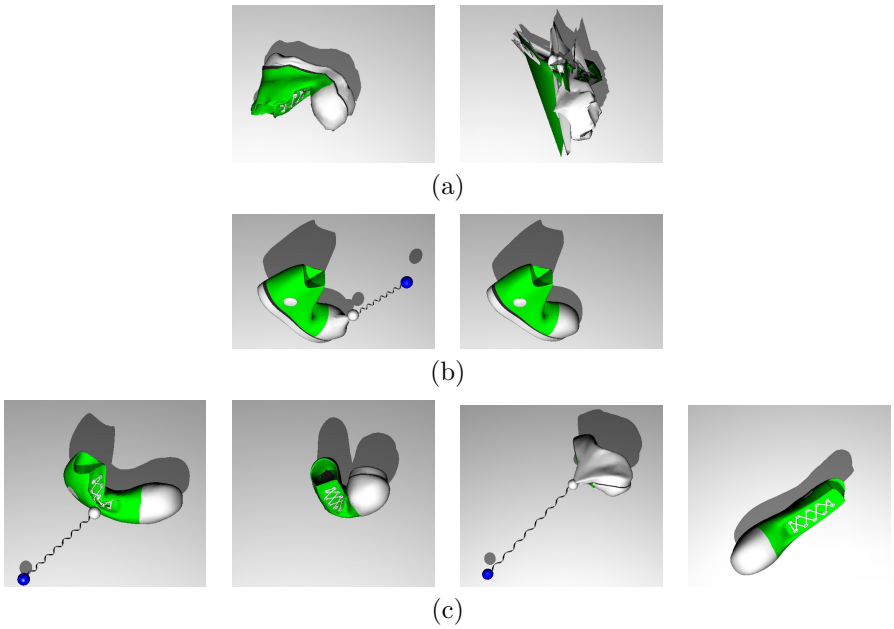


Figure 8.4: (a) The undamped sneaker does not recover to a stable resting state. (b) The point-damped sneaker remains stable and recovers to a resting state quickly. However, the global movement is heavily influenced. (c) The spring-damped sneaker recovers to a stable resting state. The global movement is not affected.

8.3 Optimized damping

In this section, the optimized damping approach is introduced. It is motivated by the fact that an iterative computation of damping forces within one simulation step can improve the reduction of oscillations (Sec. 8.3.1). It is shown that the iterative damping procedure converges and that the limit can be computed directly (Sec. 8.3.2). While the linear momentum is inherently preserved, the angular momentum has to be handled in a postprocessing step as in the spring damping. To do this, three possible ways are given and compared with respect to their applicability (Sec. 8.3.4). Further, the parameter setting and the possibilities of application are shown.

In contrast to other damping approaches, the optimized damping approach is independent from the deformation model [PB88, BW98], the integration scheme and the structure of the simulated object [NMK*06]. By choosing the damping

parameter always smaller than 1, it is guaranteed that the damping forces do not overshoot.

8.3.1 Iterative force computation

Let $\mathbf{x}_{cm}$ denote the center of mass of an object and $\mathbf{v}_{cm}$ its velocity. $\mathbf{v}_i' = \mathbf{v}_i + \frac{h}{m_i}\mathbf{f}_i$ denotes the predicted future velocity of point $\mathbf{x}_i$, and similarly $\mathbf{v}_e'$ denotes the predicted future relative velocity of distance $\mathbf{x}_e$. Finally, $\mathbf{v}_i^d = \mathbf{v}_i + \frac{h}{m_i}(\mathbf{f}_i + \mathbf{f}_i^d)$ denotes the damped velocity of point $\mathbf{x}_i$.

The neighborhood information can be stored in the *connectivity matrix* $\mathbf{E} \in \mathbb{R}^{m \times n}$, where m is the number of adjacent point pairs in the neighborhood information and n is the number of points of the object. $\mathbf{E}$ is defined as follows: For a distance $\mathbf{x}_e$ with incident points $\mathbf{x}_i$ and $\mathbf{x}_j$, where $i < j$, $\mathbf{E}_{e,i}$ is defined is $\mathbf{E}_{e,i} := 1$ and $\mathbf{E}_{e,j} := -1$. All other values of $\mathbf{E}$ are set to zero.

Similar to $\mathbf{V}$, $\mathbf{V}'$ denotes the set of predicted velocities, and $\mathbf{V}^d$ the damped velocities. $\mathbf{F}^d$ denotes the set of damping forces at the points, and $\mathbf{F}_e^d$ the set of the damping forces at the distances.

In order to show the convergence of iterative spring damping forces, the force computation has to be formulated in matrix-vector-notation, which is done in the following, starting with the standard spring damping step.

$$\begin{aligned} \mathbf{v}_i' &= \mathbf{v}_i + \frac{h}{m_i}\mathbf{f}_i & \mathbf{f}_e^d &= -\gamma \mathbf{v}_e' \\ \mathbf{v}_j' &= \mathbf{v}_j + \frac{h}{m_j}\mathbf{f}_j & \mathbf{f}_i^d &= \mathbf{f}_i^d + \mathbf{f}_e^d \\ \mathbf{v}_e' &= \mathbf{v}_i' - \mathbf{v}_j' & \mathbf{f}_j^d &= \mathbf{f}_j^d - \mathbf{f}_e^d \end{aligned} \tag{8.16}$$

The damped velocities $\mathbf{v}_i^d$ and $\mathbf{v}_j^d$ are given as

$$\begin{aligned} \mathbf{v}_i^d &= \mathbf{v}_i + \frac{h}{m_i}(\mathbf{f}_i^d + \mathbf{f}_i) = \mathbf{v}_i' + \frac{h}{m_i}\mathbf{f}_i^d \\ \mathbf{v}_j^d &= \mathbf{v}_j + \frac{h}{m_j}(\mathbf{f}_j^d + \mathbf{f}_j) = \mathbf{v}_j' + \frac{h}{m_j}\mathbf{f}_j^d. \end{aligned} \tag{8.17}$$

Using the connectivity matrix, (8.16) can be written in matrix-vector-notation:

$$\begin{aligned} \mathbf{V}' &= \mathbf{V} + h\mathbf{M}^{-1}\mathbf{F} \\ \mathbf{F}_e^d &= -\gamma \widetilde{\mathbf{E}}\mathbf{V}' \\ \mathbf{F}^d &= \widetilde{\mathbf{E}}^T \mathbf{F}_e^d \\ &= -\gamma \widetilde{\mathbf{E}}^T \widetilde{\mathbf{E}}\mathbf{V}', \end{aligned} \tag{8.18}$$

with $\widetilde{\mathbf{E}} \in \mathbb{R}^{3n \times 3n}$ arising from $\mathbf{E}$ by replacing each entry y of $\mathbf{E}$ by a 3×3-matrix $\begin{pmatrix} y & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & y \end{pmatrix}$, which accounts for the fact that each point $\mathbf{x}_i$ has three spatial components (x_i, y_i, z_i). Accordingly, (8.17) becomes

$$\begin{aligned} \mathbf{V}^d &= \mathbf{V}' + h\mathbf{M}^{-1}\mathbf{F}^d \\ &\overset{(8.18)}{=} \mathbf{V}' - \gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{V}' \\ \Rightarrow \mathbf{V}^d &= (\mathbf{id} - \gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}})\mathbf{V}' \end{aligned} \tag{8.19}$$

After k iterations, this results in

$$\begin{aligned} \mathbf{V}^{d,k} &= (\mathbf{id} - \gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}})^k\mathbf{V}' \\ &=: \mathbf{D}^k\mathbf{V}', \end{aligned} \tag{8.20}$$

where $\mathbf{V}^{d,k}$ denotes the damped velocities after k iterations.

8.3.2 Convergence of iterative damping forces

In this section, the convergence of (8.20) is shown. The idea is briefly sketched, before the proof is carried out. First, it is shown that $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ is diagonalizable and positive semidefinite, i. e. all eigenvalues are greater than or equal to zero. Then, the convergence of (8.20) can be shown by discussing the eigenvalues of $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ and $\mathbf{D}$. If the absolute values of all eigenvalues are smaller than one, Eq. (8.20) converges for $k \to \infty$.

If $\mathbf{v}'_i = \mathbf{v}'_{cm}$ for all points, $\widetilde{\mathbf{E}}\mathbf{V}' = 0$ and thus, $\mathbf{D}\mathbf{V}' = \mathbf{V}'$. Hence, 0 is always a triple eigenvalue of $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ as $\mathbf{v}'_{cm}$ has three spatial directions, and 1 is always a triple eigenvalue of $\mathbf{D}$.

Let $\mathbf{V}'_{rel}$ be the future velocity of the points relative to the center of mass. Then, $\mathbf{V}'$ can be partitioned into $\mathbf{V}' = \mathbf{V}'_{cm} + \mathbf{V}'_{rel}$, which yields $\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{V}' = \widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{V}'_{rel}$. That means, the damping forces (8.18) are not influenced by the velocity of the center of mass, which implies that it suffices to damp only $\mathbf{V}'_{rel}$ and to add $\mathbf{V}'_{cm}$ afterwards. Hence, $\mathbf{V}'$ in (8.20) can be replaced by $\mathbf{V}'_{rel}$, which results in

$$\begin{aligned} \mathbf{V}^{d,k}_{rel} &= (\mathbf{id} - \gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}})^k\mathbf{V}'_{rel} \\ &= \mathbf{D}^k\mathbf{V}'_{rel}. \end{aligned} \tag{8.21}$$

Thus, for the convergence of the damping forces it suffices to show that (8.21) converges. Hence, the eigenvalue 1 of $\mathbf{D}$, which has no influence on (8.21), can be ignored, and the eigenvalue with the largest absolute value *besides* the eigenvalue 1 has to be considered. This value is called the *key eigenvalue* in the following.

In the following two Lemmata, it is shown that the parameter of the spring damping approach can be chosen such that the absolute value of the key eigenvalue is strictly smaller than one. First, the matrix $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ is proven to be diagonalizable (Lemma 1). Note that although $\mathbf{M}$ and $\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ are symmetric, the product is not necessarily symmetric and therefore, it is not self-evident that it is diagonalizable. Second, it is shown that the damping parameter γ can be chosen such that (8.21) converges (Lemma 2).

Lemma 1. *The matrix $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ is diagonalizable and positive semidefinite.*

Proof. The matrix $\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ is symmetric, and $\mathbf{M}$ can be assumed to be positive definite and symmetric. Hence, $\mathbf{M}$ has a Cholesky factorization $\mathbf{M} = \mathbf{G}\mathbf{G}^T$ [DR08]. It follows that the eigenvalue equation $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{v} = \lambda\mathbf{v}$ is equivalent to $\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{G}^T\mathbf{v} = \lambda\mathbf{G}^T\mathbf{v}$, which can be written as $\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{w} = \lambda\mathbf{w}$ with $\mathbf{w} = \mathbf{G}^T\mathbf{v}$. The matrix $\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}$ is symmetric, hence orthogonally diagonalizable, and positive semidefinite, as $\mathbf{x}^T\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{x} = \|\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{x}\|^2 \geq 0$. Hence, there is a rotation matrix $\mathbf{R}$ such that $\mathbf{R}^T\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{R} = \mathbf{\Lambda}$ has diagonal form, and all diagonal entries are greater than or equal to zero. Defining $\mathbf{P} := \mathbf{G}^{-T}\mathbf{R}$ leads to

$$\begin{aligned}\mathbf{P}^{-1}\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{P} &= \mathbf{R}^T\mathbf{G}^T\mathbf{G}^{-T}\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{R} \\ &= \mathbf{R}^T\mathbf{G}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{G}^{-T}\mathbf{R} \\ &= \mathbf{\Lambda}.\end{aligned} \tag{8.22}$$

It follows that $\mathbf{P}$ diagonalizes $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ and that all eigenvalues are greater than or equal to zero, which concludes the proof. □

Lemma 2. *The damping parameter γ in* (8.21) *can always be chosen such that the absolute value of the key eigenvalue of $\mathbf{D}$ is strictly smaller than one.*

Proof. With the notation of Lemma 1, it follows that $\mathbf{P}^{-1}\mathbf{D}\mathbf{P} = \mathbf{P}^{-1}\mathbf{i}\mathbf{d}\mathbf{P} - \gamma h\mathbf{P}^{-1}\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}\mathbf{P} = \mathbf{id} - \gamma h\mathbf{\Lambda}$ has diagonal form. Thus, the eigenvalues $\mu_i, i = 1, \ldots, n$ of $\mathbf{D}$ can be written as $1 - \lambda_i, i = 1, \ldots, n$, with λ_i being the eigenvalues of $\gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$.

Further, $\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ has rank $3n - 3$ as $\mathbf{E}$ has rank $n - 1$ for connected objects. Therefore, exactly three eigenvalues are 0, which belong to the different directions of $\mathbf{v}'_{cm}$, and all remaining eigenvalues are strictly greater than zero.

Obviously, depending on $\mathbf{M}$ and h, γ can be chosen small enough such that all eigenvalues of $\gamma h\mathbf{M}^{-1}\widetilde{\mathbf{E}}^T\widetilde{\mathbf{E}}$ besides 0 lie in the interval $(0, 2)$. This immediately yields $\mu_i \in (-1, 1)$ for all eigenvalues of $\mathbf{D}$ besides 1, and the absolute value of the key eigenvalue is strictly smaller than one. □

Based on Lemma 2, the convergence of (8.21) can be shown easily.

Theorem 1. *If γ is chosen small enough in* (8.21)*, then $\mathbf{V}_{rel}^{d,k} \to 0$ for $k \to \infty$.*

Proof. Lemma 2 states that γ can be chosen such that the absolute value of the key eigenvalue μ is strictly smaller than one. Thus, it follows $\|\mathbf{V}_{rel}^{d,k}\| \leq \mu\|\mathbf{V}_{rel}^{d,k-1}\| \leq \ldots \leq \mu^k\|\mathbf{V}'_{rel}\|$. This yields $\mathbf{V}_{rel}^{d,k} \to 0$ for $k \to \infty$. □

8.3.3 Direct force computation

As shown in Theorem 1, the iterative spring damping approach leads to $\mathbf{V}^d_{rel} = 0$ for an appropriately chosen γ. Of course, the convergence needs lots of iterations and therefore, it is much less efficient than the simple spring damping approach. However, the limit of the damping forces can be computed directly using Theorem 1, if it is ensured that the computed damping forces yield $\mathbf{V}^d_{rel} = 0$. For a single point, this implies $\mathbf{v}^d_{i,rel} = \mathbf{v}'_{i,rel} + \frac{h}{m_i}\mathbf{f}^d_i = 0$, and the damping force can be computed as

$$\mathbf{f}^d_i = -\frac{m_i}{h}\mathbf{v}'_{i,rel}. \tag{8.23}$$

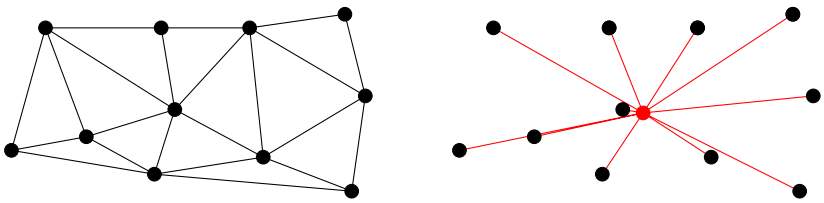
This computation is very simple and efficient, and it is justified by the fact that it is the limit of an infinite number of iterations of the standard spring damping approach.

It is easy to introduce a new damping parameter γ in (8.23) to compute the damping forces as

$$\mathbf{f}^d_i = -\gamma\frac{m_i}{h}\mathbf{v}'_{i,rel}, \tag{8.24}$$

which obviously should not be greater than one. Hence, γ is always within the range between 0 and 1.

Note that the proof of Theorem 1 did not employ any specific information about the structure of $\widetilde{\mathbf{E}}$ except the fact that its rank is $3n-3$. But this is correct if the rank of $\mathbf{E}$ is $n-1$, which is fulfilled if the object is connected. Thus, the proof holds for any type of connectivity structure that defines a connected object. Further, the connectivity information is canceled out in the limit for $k \to \infty$ in (8.21), and as seen in (8.24), it is not needed any more (Fig. 8.5).



(a) (b)

Figure 8.5: The connectivity structure (a) vanishes in the limit of an infinite number of spring damping iterations. The damping forces can be calculated directly with respect to the center of mass (b).

8.3.4 Momentum conservation

As damping forces should not influence the global movement of an object, they must guarantee the conservation of linear and angular momentum. While it is

easy to show that the sum of the damping forces in (8.24) is zero and the linear momentum is conserved, the computation of damping forces commonly results in a nonzero torque, i. e. the condition

$$\sum_{i=1}^{n}(\mathbf{x}_{i,rel} \times \mathbf{f}_i^d) = 0, \tag{8.25}$$

where $\mathbf{x}_{i,rel} := \mathbf{x}_i - \mathbf{x}_{cm}$, is not fulfilled, which has to be handled in a post-processing step. Though first, the conservation of the linear momentum is shown.

$$\begin{aligned}
\sum_{i=1}^{n} \mathbf{f}_i^d &= -\sum_{i=1}^{n} \frac{m_i}{h} \mathbf{v}'_{i,rel} \\
&= -\frac{1}{h} \sum_{i=1}^{n} m_i (\mathbf{v}'_i - \mathbf{v}'_{cm}) \\
&= -\frac{1}{h} \left(\sum_{i=1}^{n} m_i \mathbf{v}'_i - \mathbf{v}'_{cm} \sum_{i=1}^{n} m_i \right) \\
&= 0.
\end{aligned}$$

In the following, different ideas to eliminate the torque are presented. In the simple spring damping approach, damping forces are computed per distance. To cancel out the torque, they can simply be projected onto the distance, which results in zero torque. The novel damping approach in (8.24) directly computes forces per point instead of forces per distance. Therefore, the projection technique is not self-evident and different methods are possible. The usability of the proposed methods will be discussed in Sec. 8.3.6

Force projection onto the distances

The first method is similar to the simple spring damping approach. The forces per point are transformed into forces per distance and are projected similar to the spring damping approach. This is done the following way.

For a distance $\mathbf{x}_e$ with incident points $\mathbf{x}_i$ and $\mathbf{x}_j$, $\mathbf{f}_e^d$ is set to $\mathbf{f}_e^d = \mathbf{f}_i^d - \mathbf{f}_j^d$ and afterwards, it is projected onto $\mathbf{x}_e$ using $\overline{\mathbf{f}_e^d} = (\mathbf{f}_e^d \cdot \mathbf{x}_e) \frac{\mathbf{x}_e}{\|\mathbf{x}_e\|_2^2}$ like in Sec. 8.2.1.

After projecting the force, it is re-distributed to the incident points using temporary variables $\overline{\mathbf{f}_i^d}$ and $\overline{\mathbf{f}_j^d}$:

$$\begin{aligned}
\overline{\mathbf{f}_i^d} &= \overline{\mathbf{f}_i^d} + \overline{\mathbf{f}_e^d} \\
\overline{\mathbf{f}_j^d} &= \overline{\mathbf{f}_j^d} - \overline{\mathbf{f}_e^d}.
\end{aligned} \tag{8.26}$$

This obviously results in zero torque. However, after all distances $\mathbf{x}_e$ have been processed, the magnitude of the force $\overline{\mathbf{f}_i^d}$ can be much higher than the

magnitude of $\mathbf{f}_i^d$. To overcome this problem, the damping forces have to be scaled with a constant factor c which is given by

$$c = \min_{i=1,\dots,n} \frac{\|\mathbf{f}_i^d\|}{\|\overline{\mathbf{f}_i^d}\|}, \tag{8.27}$$

which leads to $\mathbf{f}_i^d = c\overline{\mathbf{f}_i^d}$.

Force projection onto other relative positions

If there is a particular point $\mathbf{x}$ in the connectivity structure, for example the center of mass, it can be more suitable to project the forces onto the relative position $\mathbf{x}_{i,rel} = \mathbf{x}_i - \mathbf{x}$ (Fig. 8.6). Obviously, the torque with respect to $\mathbf{x}$, i. e. $\sum_{i=1}^n \mathbf{x}_{i,rel} \times \mathbf{f}_i^d$, is zero, as $\mathbf{x}_{i,rel}$ and $\mathbf{f}_i^d$ are collinear and thus, their cross product is zero. However, after this projection, the sum of the forces is not necessarily zero. This can be handled the following way. Let $\mathbf{x} = \sum \alpha_i \mathbf{x}_i$ with $\sum \alpha_i = 1$ be the representation of $\mathbf{x}$ using the barycentric coordinates, and let $\mathbf{f} = \sum \mathbf{f}_i^d$ be the possibly non-zero sum of the projected forces. If $\alpha_i \mathbf{f}$ is subtracted from each $\mathbf{f}_i^d$, the sum of the forces is zero again. Also, the torque remains zero:

$$\begin{aligned} \sum_{i=1}^n \mathbf{x}_{i,rel} \times (\alpha_i \mathbf{f}) &= \sum_{i=1}^n (\mathbf{x}_i - \mathbf{x}) \times (\alpha_i \mathbf{f}) \\ &= \sum_{i=1}^n (\alpha_i \mathbf{x}_i) \times \mathbf{f} - \mathbf{x} \times \mathbf{f} \sum_{i=1}^n \alpha_i \\ &= \mathbf{x} \times \mathbf{f} - \mathbf{x} \times \mathbf{f} = 0 \end{aligned} \tag{8.28}$$

Thus, the sum of the damping forces as well as the torque is zero, and both linear and angular momentum are conserved. Note that the last force correction does not impose an additional relative movement onto the affected nodes and therefore, the damping is only affected by the projection which justifies this procedure. The linear momentum is not affected at all. The applicability of this projection approach is discussed in Sec. 8.3.6.

Torque elimination using Linear Programming

Linear Programming can be used as a third possibility to eliminate the torque. To establish the corresponding Linear Program, the cross product can be written as a matrix-vector-multiplication using skew-symmetric matrices. For a vector $\mathbf{a} = (a_x, a_y, a_z)^T$, the matrix $\widetilde{\mathbf{a}}$ defined as

$$\widetilde{\mathbf{a}} := \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{8.29}$$

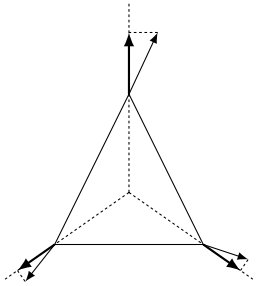


Figure 8.6: Force projection onto relative positions. The fine arrays indicate the computed damping forces, while the bold arrays indicate the projected forces.

allows to replace the cross product $\mathbf{a} \times \mathbf{b}$ by $\widetilde{\mathbf{a}}\mathbf{b}$. The zero-torque-condition (8.25) then becomes

$$\sum_{i=1}^{n} \widetilde{\mathbf{x}}_{i,rel} \mathbf{f}_i^d = 0. \tag{8.30}$$

As this sum is not zero in general, correction forces $\mathbf{f}_i^c$ have to be computed in order to eliminate the torque. Then, the condition becomes

$$\sum_{i=1}^{n} \widetilde{\mathbf{x}}_{i,rel} (\mathbf{f}_i^d + \mathbf{f}_i^c) = 0, \tag{8.31}$$

which is one of the conditions for the Linear Program.

To keep the linear momentum, the sum of the correction forces $\sum_{i=1}^{n} \mathbf{f}_i^c$ has to be zero, because $\sum_{i=1}^{n} \mathbf{f}_i^d$ is already zero. Further, a reasonable objective function is to demand that the correction forces are as small as possible. Thus, the objective function has to be:

$$\min \sum_{i=1}^{n} \|\mathbf{f}_i^c\|_1. \tag{8.32}$$

However, for a Linear Programming problem, the absolute value is not allowed in the objective function, and has to be replaced by a term without absolute values. This can be done by partitioning $\mathbf{f}_i^c = \mathbf{f}_i^{c,+} - \mathbf{f}_i^{c,-}$ into its positive and negative components together with the constraint that $\mathbf{f}_i^{c,+}$ and $\mathbf{f}_i^{c,-}$ are greater than or equal to zero:

$$\begin{aligned} &\min \sum_{i=1}^{n} \|\mathbf{f}_i^{c,+} - \mathbf{f}_i^{c,-}\|_1 \\ =& \min \sum_{i=1}^{n} (\|\mathbf{f}_i^{c,+}\|_1 + \|\mathbf{f}_i^{c,-}\|_1). \end{aligned} \tag{8.33}$$

Eq. (8.33) holds as it is a minimization problem. If, for example, the first component of $\mathbf{f}_i^{c,+}$ and the first component of $\mathbf{f}_i^{c,-}$ were both nonzero, this would not be the optimal solution of the minimization problem. Thus, for the optimal solution, always at least one of the corresponding components of $\mathbf{f}_i^{c,+}$ and $\mathbf{f}_i^{c,-}$ must be zero, implying the correctness of (8.33). Note that (8.33) does not contain absolute values for the components of $\mathbf{f}_i^{c,+}$ and $\mathbf{f}_i^{c,-}$, as $\mathbf{f}_i^{c,+}$ and $\mathbf{f}_i^{c,-}$ are demanded to be greater than or equal to zero.

In total, the Linear Program for torque elimination can be established as follows:

$$\begin{aligned} \min \sum_{i=1}^{n} (\mathbf{f}_i^{c,+} + \mathbf{f}_i^{c,-}) \qquad & (8.34) \\ s.t. \qquad \mathbf{f}_i^{c,+} &\geq 0 \\ \mathbf{f}_i^{c,-} &\geq 0 \\ \sum_{i=1}^{n} \widetilde{\mathbf{x}}_{i,rel} (\mathbf{f}_i^{d} + \mathbf{f}_i^{c,+} - \mathbf{f}_i^{c,-}) &= 0 \\ \sum_{i=1}^{n} (\mathbf{f}_i^{c,+} - \mathbf{f}_i^{c,-}) &= 0 \end{aligned}$$

This Linear Program is feasible, as $\mathbf{f}_i^{c,+} - \mathbf{f}_i^{c,-} = -\mathbf{f}_i^{d}$ is a possible solution, and bounded, as the objective function is always greater than or equal to zero. Thus, it has an optimal solution which eliminates the torque.

8.3.5 Application

In this part, two example settings are described that show how the new damping approach can be applied to compute damping forces. The methods are referred to as global and local damping, as the whole object is damped in the first one and smaller structures are damped locally in the second one. The influence of the torque elimination method on these settings is discussed in Sec. 8.3.6.

Global damping

The global damping approach is the most evident idea following (8.24). In this version, the object is damped as a whole. The damping forces are calculated as given in (8.24) with respect to the center of mass, and the torque is eliminated with one of the above mentioned ideas.

Local damping

In this approach, the object is divided into clusters, and the movement of each cluster is damped relative to its center of mass, but not relative to the center of mass of the whole object. For example, in a tetrahedral mesh, the tetrahedrons can be taken as clusters. To get reasonable damping forces that do not overshoot,

the number of clusters a point lies in has to be taken into account. Therefore, the mass m_i of point $\mathbf{x}_i$ is divided by the number of clusters N_i the point belongs to, and m_i/N_i is used as the mass of $\mathbf{x}_i$ within each of its clusters (cf. [RJ07]).

Of course, it is possible to first perform a global damping step which stabilizes the object in the case of external forces and to perform a local damping step afterwards to reduce oscillations.

8.3.6 Properties of the proposed damping scheme

In this section, the properties of the proposed damping approach and the different torque elimination schemes are discussed.

Obviously, the damping forces (8.23) are optimal in the sense that they damp the whole relative movement. Therefore, they lead to an unconditionally stable simulation. But as they lead to a nonzero torque, it is necessary to handle this drawback. Eliminating the torque obviously introduces some uncontrollable effects, which should be minimized by the torque elimination technique.

The three different approaches that are proposed in Sec. 8.3.4 to eliminate the torque differ in an essential fact. While the first projection technique, which uses the distances, cares for a local torque elimination, the second projection approach and the LP formulation eliminate the torque only globally. Also, they guarantee the conservation of the global linear momentum, while projecting onto the distances or other local structures preserves global and local linear momentum. The experiments showed that the global conservation of linear and angular momentum is not strong enough, as forces and torques at one part of an object can be balanced by forces and torques at another part. This results in undesired artifacts.

Another difference of the approaches is that the projecting methods do not look for a minimum solution, but instead they choose a specific direction. As both methods show better results than the LP formulation, it seems that it is much more important to have any control about the directions of the projected forces than to minimize the magnitude of the correction forces. This is further affirmed by the results that were achieved using the LP formulation for each tetrahedron locally.

Following this, for the global damping approach, projecting the forces onto the distances currently is the best technique to eliminate the torque, as it guarantees the local conservation of linear and angular momentum. Concerning local damping, the second projection method should be preferred, as the directions from the center of mass to the points are more likely to match the directions of the damping forces which are also relative to the center of mass. The locality of momentum conservation is fulfilled self-evidently in the local damping.

The projection methods can be applied to any set of forces that should guarantee the preservation of linear and angular momentum. For example, they could be applied to constraint forces in order to preserve the global movement of an object. Even the point damping forces would yield reasonable damping forces after applying the torque elimination procedure.

For the local damping approach, it turned out that omitting the projection

results in great improvement of stability for the drawback that the global orientation is affected. However, there are scenarios where this fact is accepted because of the great improvement of stability.

8.3.7 Force propagation

Due to the discrete time integration, the propagation of external forces, e. g. collision, constraint or user interaction forces, proceeds slowly. For example, a user interaction force that is applied to a point $\mathbf{x}_i$ causes only this point to move within the following time step. In the subsequent time step, this causes forces on all neighboring points of $\mathbf{x}_i$ by the deformation model. However, in each time step, the force is propagated to the neighboring vertices only. This behavior is independent from the object's stiffness, but depends on the discretization, and causes an unrealistically weak impression compared to the normally expected perception. Further, for large external forces, the displacement of $\mathbf{x}_i$ causes high strain values, affecting the stability of the simulation. It should be emphasized that this is not a problem of the deformation model, but of the discrete time integration, which leads to a non-optimal interpolation of the displacement field.

To overcome this limit concerning external forces, the iterative damping approach can be applied. Fig. 8.7 sketches a few iterations of the spring damping scheme in a one-dimensional example. An external force $\mathbf{f}$ is applied at mass point 6. In the first iteration of the spring damping, there is a non-zero relative velocity between points 5 and 6, but zero relative velocities elsewhere. Thus, this results in a non-zero damping force only at the points 5 and 6.

In the second iteration, also a non-zero relative velocity between the points 4 and 5 is observed, which results in a damping force at these points. Also, the relative velocity between 5 and 6 is non-zero again, even though it is smaller than in the first iteration. Obviously, the damping force between 4 and 5, which was caused by the damping force between 5 and 6, in turn influences the damping force between 5 and 6 in the subsequent iterations. This procedure is continued, thus in the next iteration, there will be a damping force between 3 and 4, in turn influencing the damping force between 4 and 5.

In each iteration, the force is propagated one element further. Moreover, each force influences the one it was caused by. Thus, in the third iteration, the relative velocity between 5 and 6 could possibly be greater than in the second iteration, as the damping force between 4 and 5 reduces the resulting damping force at 5.

The first aspect is desired, as some amount of the external force $\mathbf{f}$ is propagated to vertex 1 after 5 damping iterations, but within one time step. However, the second aspect seems a little disturbing, as the iterative procedure seems to oscillate. However, in Sec. 8.3.2, it has been shown that this is not the case and the procedure converges.

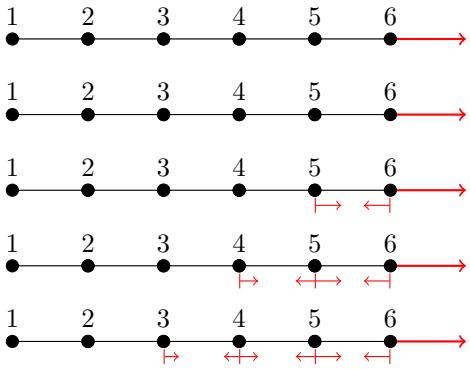


Figure 8.7: Illustration of the first few spring damping iterations. The red arrows illustrate the external force **f** and the sketched damping forces.

8.4 Results

The general benefit of damping in dynamic simulations and the spring damping approach are illustrated in Fig. 8.4. In this section, various scenes show the improvements obtained by the proposed damping approach. The scenes can be simulated stably with the proposed damping approach, but get unstable with the standard spring damping approach. Thus, the proposed approach allows for larger time steps, which results in more efficient simulations. The objects consist of tetrahedral meshes generated by [SWT06]. Collisions are detected using [THM*03], and contacts are handled using [HTK*04].

Fig. 8.8 illustrates the force propagation of the proposed damping scheme. An external force is applied to the right hand side of a bar which is shown by the tetrahedral mesh that is used for simulation (Fig. 8.8 (a)). Points where a damping force is computed are marked with red color depending on the magnitude of the force. Using the global damping approach without force projection, the force is propagated within one time step (Fig. 8.8 (b)). Including the force projection, in this example it takes about 20 time steps to propagate the external force (Fig. 8.8 (c)). The resulting forces are illustrated after 10 and 20 time steps, respectively. In contrast, the spring damping approach cannot propagate external forces comparably fast (Fig. 8.8 (d)). Here, the resulting forces after 10, 20 and 50 time steps are shown, which still is not sufficient to propagate the external force to the left end of the bar. Moreover, the magnitudes of the propagated forces are smaller compared to the global damping approach.

Fig. 8.9 illustrates that the proposed damping approach is able to handle scenes where the spring damping approach cannot keep the simulation stable. Here, a global damping step is performed which is followed by a local damping step. This combination yielded the best results. Note that before the local damping is performed, the global damping forces are already projected using the first projection method. The setting of the experiment is quite simple: A sphere falls down onto a membrane. While the simulation fails using the spring

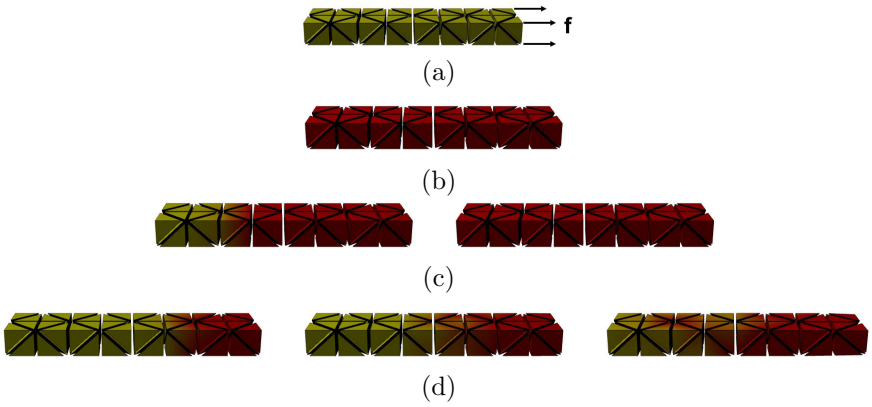


Figure 8.8: Illustration of the force propagation using the optimized damping. (a) In the initial position, an external force is applied to the right part of the bar. (b) Using the global damping without force projection, it is propagated within one time step. (c) Using the global damping with force projection, it takes 20 time steps to propagate the force. The snapshots are given after 10 and 20 time steps. (d) The non-iterative spring damping approach takes much longer to propagate the force. The snapshots are given after 10, 20 and 50 time steps.

damping approach, it remains perfectly stable with the optimized damping approach.

Fig. 8.10 shows two cubes colliding with high relative velocities. Thus, large collision response force occur. Using the spring damping approach, the simulation fails (Fig. 8.10 (b)), while it remains stable using the optimized damping approach (Fig. 8.10 (c)). Again, a global damping step is performed followed by a local damping step.

Fig. 8.11 illustrates the propagation of user interaction forces. A rope which is lying on the floor is picked up by some draggers. The simulation remains stable using the optimized damping approach (Fig. 8.11 (a)). In the situation of the lower picture in Fig. 8.11 (a), it is switched to the standard spring damping approach, and the simulation gets unstable (Fig. 8.11 (b)).

Fig. 8.12 illustrates another scenario where the proposed approach approach allows for a larger time step compared to existing solutions. Like in Fig. 8.9 and Fig. 8.10, both a global and a local damping step are applied. The rope bridge consists of many tetrahedral meshes that are connected with local constraints [GBT06]. Due to the locality and magnitude of the constraint forces, a fast force propagation is very important to keep the object stable, which is implicitly done by the optimized damping approach.

8.5 Conclusion

In this section, the benefit of velocity-dependent damping forces in numerical integration schemes has been illustrated. Further, these benefit has also been shown in the context of dynamic simulations, where especially damping approaches using relative velocities are useful. The spring damping approach has been shown to keep simulations stable using comparably large time steps.

Moreover, a new damping approach has been introduced that is inspired by the idea that an iterative computation of damping forces further improves the stability. The iterative spring damping approach converges if the damping parameter is chosen appropriately, and the limit can be computed directly without actually performing iterations. The approach is independent of the deformation model and the integration scheme and does not need any connectivity information. Thus, it can be applied for arbitrary object representations, e. g. meshless approaches [MHTG05, RJ07]. While the linear momentum is automatically conserved, the torque has to be eliminated in a postprocessing step, for which three possible methods have been introduced. These approaches can also be applied to any other set of forces that have to guarantee the conservation of angular momentum. Different applications of the approach as global damping to the object as a whole or as local damping to separate clusters of the object have been illustrated. Further, the approach simplifies the parameter setting, as the damping constant is always within the range between 0 and 1. In the result section, it has been shown that the damping approach allows for larger time steps in dynamic simulations.

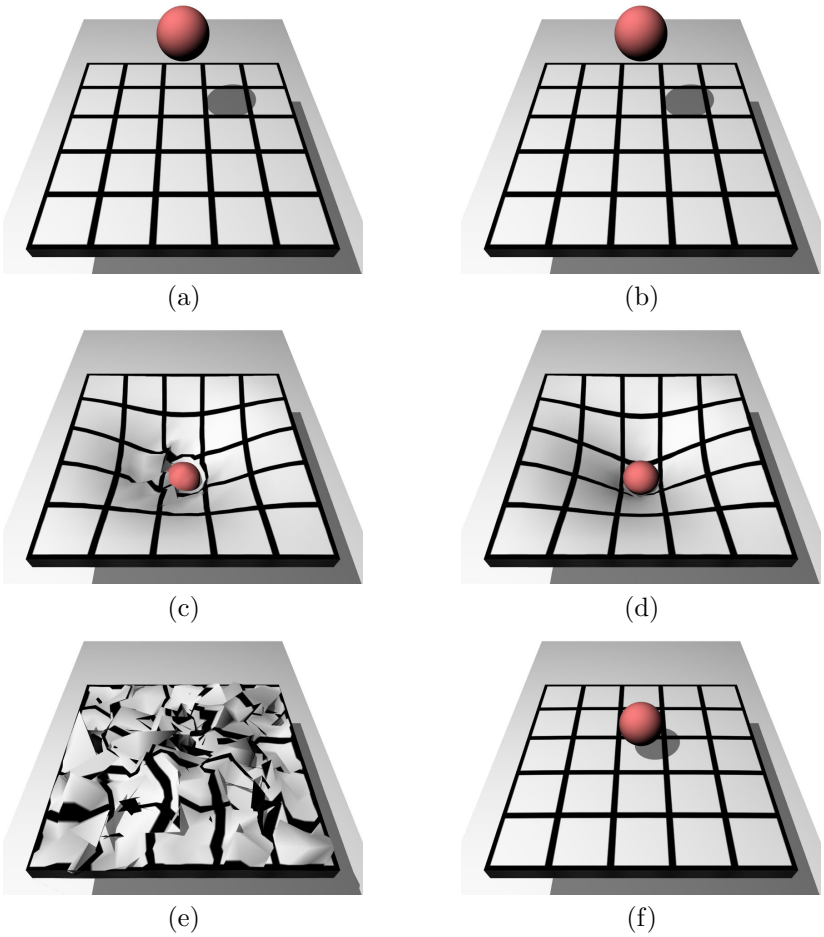
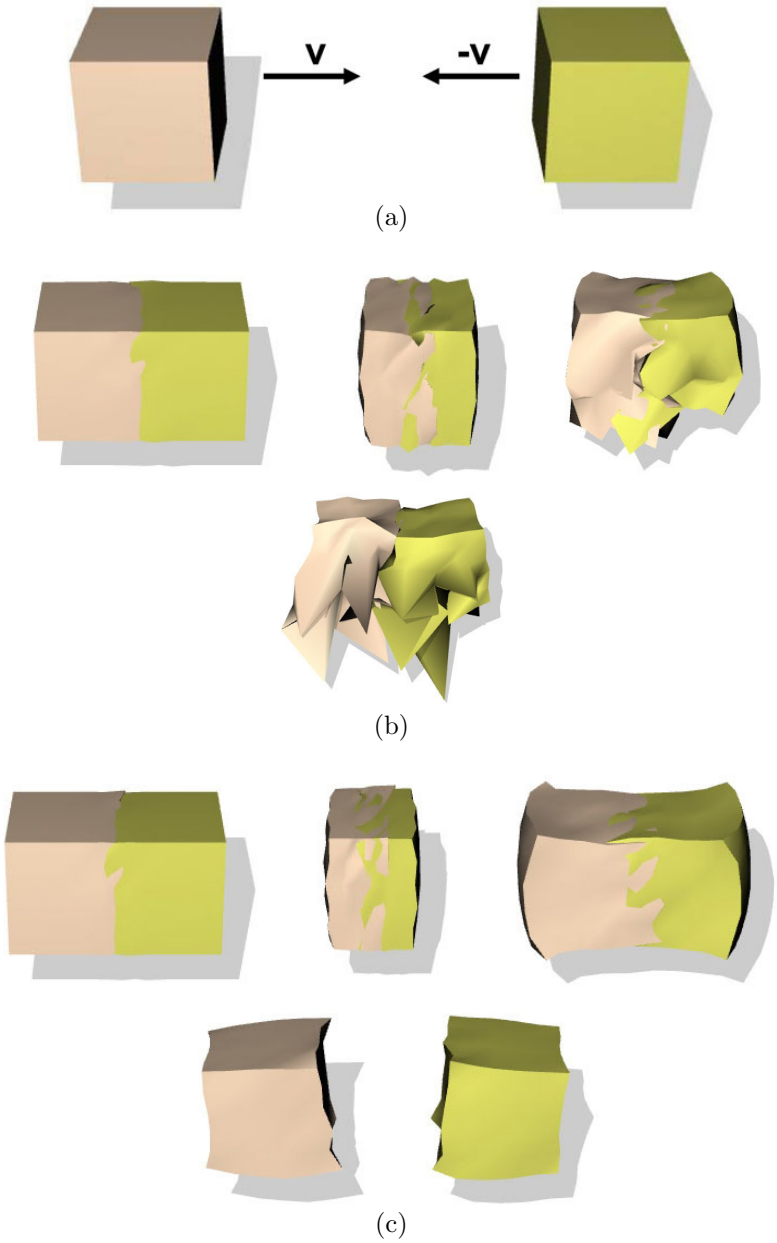


Figure 8.9: This experiment illustrates the differences between the spring damping approach (left column) and the optimized approach (right column). The compared pictures are always chosen at the same simulation step. (a,b) Configuration of the experiment: A sphere falls down onto the membrane. (c) Using the spring damping approach, the membrane does not remain stable. (d) The membrane shows no artifacts using the optimized damping approach. (e) After the collision contact, the membrane does not return to its resting state using the spring damping. (f) The sphere bounces back using the optimized approach.

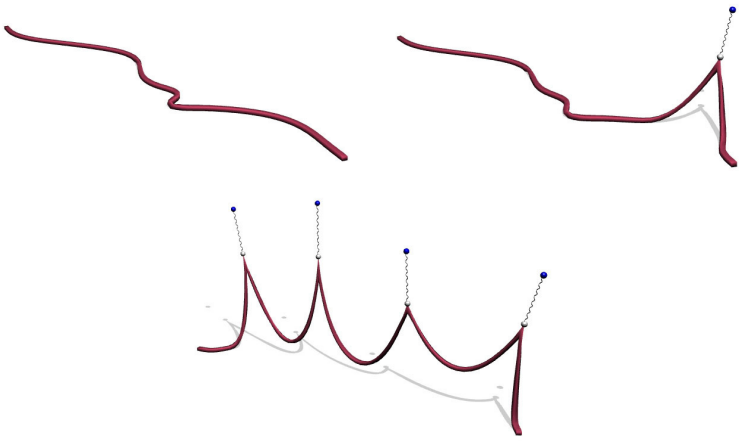


(a)

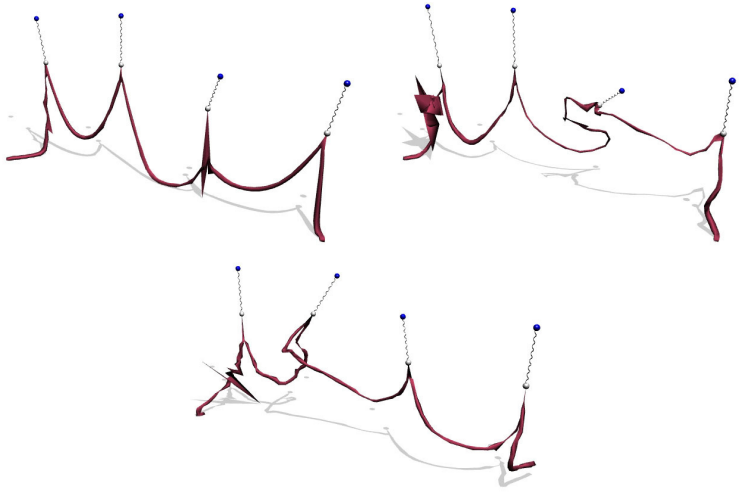
(b)

(c)

Figure 8.10: This example illustrates that large collision response forces can be handled. (a) Configuration of the experiment: Two cubes with high relative velocities. (b) Using the spring damping approach, the simulation fails during the collision response. (c) The simulation keeps stable using the optimized damping approach.



(a)



(b)

Figure 8.11: Propagation of user interaction forces. (a) A rope lying on the floor is picked up by some draggers. Using the optimized damping approach, the simulation is stable due to the fast force propagation. (b) Switching to the standard spring damping approach in the situation of the lower picture in (a), the simulation becomes unstable and finally fails.

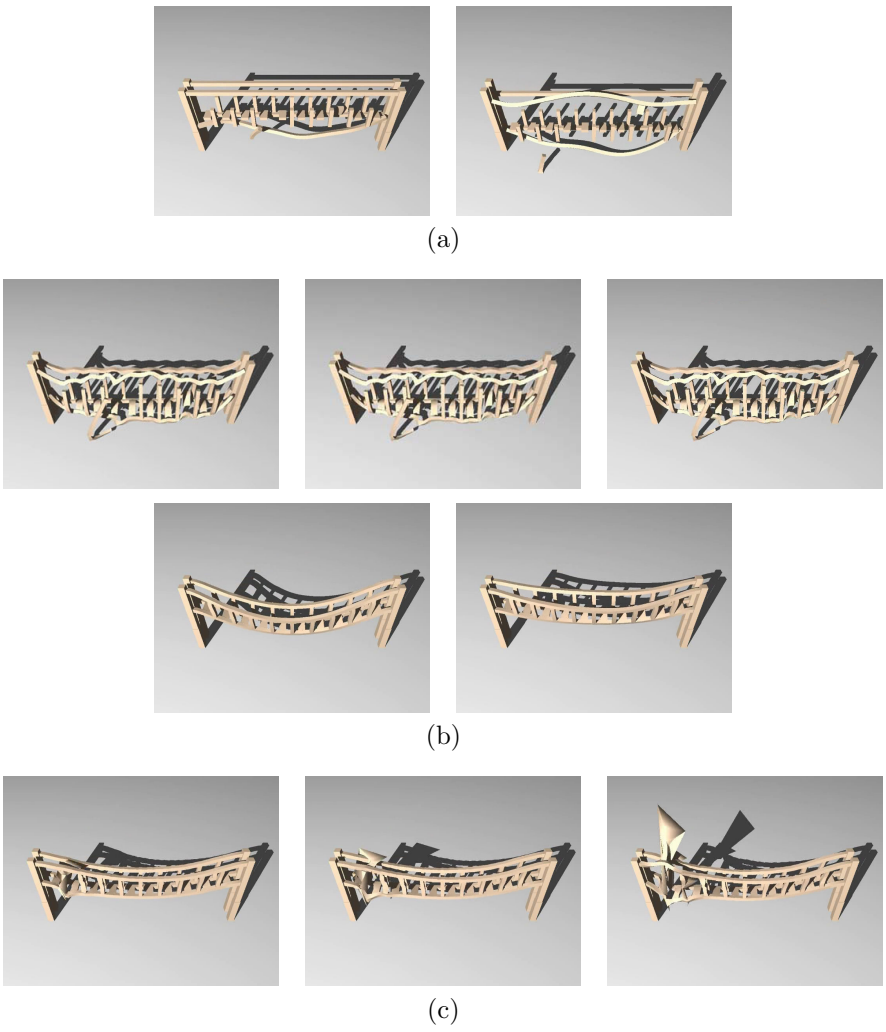


Figure 8.12: This experiment illustrates a scenario where a larger time step can be chosen using the optimized damping approach. (a) Setting of the experiment: A bridge is established of several single elements using constraint forces. Without the constraint forces, the objects can move freely. (b) Switching the constraints on, the bridge is established by the constraint forces. The simulation remains stable due to the fast force propagation by the optimized damping approach. (c) Switching to the standard spring damping approach after the bridge is established, the simulation fails immediately as the constraint forces cannot be handled robustly.

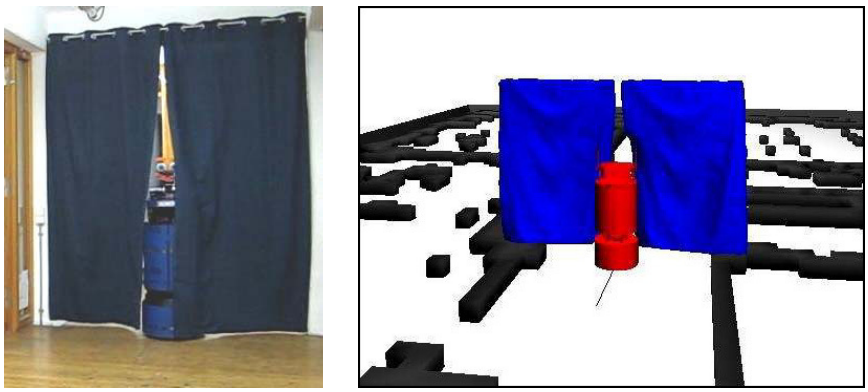
9 Application

The simulation environment has been integrated into a path planning algorithm for mobile robots in [FBS*08a, FSS*09]. Path planning is one of the fundamental problems of mobile robotics, as the ability of a safe navigation in space is a precondition for autonomous robots. The objective is to find a collision-free trajectory from a starting position to a goal position in a given environment. This fundamental task has been well-studied in the past, and numerous approaches have been developed [Lat91, CLH*05, LaV06].

Though few path planning approaches consider deformable robots [KLH98, LK01, GSLM05, MK06], the majority of algorithms has concentrated on environments with rigid obstacles [Lat91, CLH*05, LaV06]. On the contrary, environments in daily life contain non-rigid objects like curtains, cloth, or plants, and humans employ their knowledge about the deformability of such objects in their actions. Thus, also mobile navigation algorithms benefit from the possibility to include information about deformable objects. An example is illustrated in Fig. 9.1, where a robot has to pass through a curtain in order to reach its goal position. Approaches considering rigid obstacles only will either fail, if no collision-free path exists, or have to take a costly detour. In contrast, planning approaches considering deformable objects are able to trade-off the travel cost and deformation energy in order to find a faster or a more energy-efficient path.

Besides the planning process, considering deformable objects also poses a new challenge during the path execution. In rigid environments, the planned paths are demanded to be collision-free. Nevertheless, the robot could collide with the environment e. g. due to inaccuracies in motion or localization, or it might face unforeseen obstacles. In both scenarios, collisions have to be strictly avoided and the robot has to be able to detect possible collisions in order to stop its motion until the obstacle is away or to replan its path. To detect collisions, there are sensor-based approaches like potential fields [Kha86, KC95] or dynamic windows [FBT97, BK99] as well as methods based on the environment map [FBTC98, SA92]. In environments with deformable objects, this problem gets even worse as the possibility to interact with deformable objects weakens the constraint of collision avoidance. Collisions with deformable objects are permitted or even desired on optimal paths. Nevertheless, unforeseen rigid objects have to be reliably detected for a safe navigation.

In this chapter, two approaches are illustrated that account for these tasks.



(a) (b)

Figure 9.1: A robot passing through a curtain. (a) shows the robot in the real world moving on its planned path. (b) illustrates a model of the robot in the simulation environment where the deformation cost is estimated.

In Sec. 9.1, a path planning approach from [Fra07, FBS*08a] is illustrated that allows for environments with deformable objects and estimates the deformation energy using the simulation environment introduced in Chapter 3. In Sec. 9.2, an approach from [FSS*09] for the safe navigation in environments with deformable objects is sketched. It is based on a learned sensor model that allows to distinguish measurements that are caused by deformable objects from measurements originated by other obstacles.

9.1 Path planning in environments containing deformable objects

A planning approach that accounts for deformable objects has been published in [Fra07, FBS*08a]. The approach estimates the deformation energy along a path by simulating the robot's movement in the simulation environment and thus, it achieves a trade-off between path and travel cost. In contrast to earlier approaches [KLH98, BLA02, RLA06], the planning approach can be executed online as the computationally expensive computation of the deformation energy is done in a preprocessing step [FBS*08b] which allows for an accurate approximation during the path planning.

The approach employs the Probabilistic Roadmap Method (PRM) [KSL*96], which works within the configuration space $\mathcal{C}$. A configuration q of a robot is given by a vector denoting its position and orientation, or the position and orientation of joints, for example. $\mathcal{C}$ contains all possible configurations of the robot and is divided into the set $\mathcal{C}_{free}$ of collision-free states and the obstacle region $\mathcal{C}_{obs} := \mathcal{C} \setminus \mathcal{C}_{free}$, i. e. the configurations that lead to a collision of the robot with an obstacle in the real world. Thus, a feasible path of a robot in the real world corresponds to a continuous path from a starting configuration q_{start}

to a goal configuration q_{goal} within $\mathcal{C}_{free}$. The Probabilistic Roadmap Method establishes a graph in $\mathcal{C}_{free}$ by using random samples as vertices which are connected by a local planner if there is a collision-free path between them. The optimal path for the robot is computed by employing a shortest path algorithm on that graph.

[Fra07, FBS*08a] extend the feasible configurations $\mathcal{C}_{free}$ by the set $\mathcal{C}_{def}$ of configurations where the robot is in contact with a deformable object, and employ the Probabilistic Roadmap Method within $\mathcal{C}_{free} \cup \mathcal{C}_{def}$. To achieve a trade-off between travel and deformation cost, the deformation energy along an edge i of the probabilistic roadmap is estimated by simulating the robot's movement along the corresponding path within the simulation environment (Chapter 3). The underlying deformation model has been presented in Chapter 5, and the deformation parameters can be estimated using the approach proposed in Sec. 7.1. The deformation energy returned by the simulation environment is computed according to Eq. (5.50). The total cost of each edge i is defined as a weighted sum of deformation and travel cost.

Using this cost function, the Probabilistic Roadmap Method computes a path with an optimal trade-off between deformation and travel cost (Fig. 9.2). However, due to the computational requirements in the simulation process, answering path queries takes a long time. Thus, the deformation cost is estimated based on precomputed energy values [FBS*08b] in order to allow for an online execution of the planning approach. Therefore, the deformable objects are assumed to be fixed in the environment so that the deformation cost mainly depends on the trajectory of the robot relative to an object, and the deformation cost is precomputed for a number of line segments through each object. Then, for an arbitrary edge in the roadmap, the deformation energy is estimated as a weighted sum over all neighboring paths for which the deformation energy was precomputed. This results in a great speedup, as path queries can typically be answered within few hundred milliseconds, while the costly precomputation step has to be done only once for each object. The experiments in [FBS*08a] show that the computed paths are similar to an exact deformation computation, which justifies the deformation cost approximation due to the enormous gain of planning performance.

9.2 Collision avoidance in environments containing deformable objects

Fig. 9.3 illustrates the execution of the path planned in Fig. 9.2 (a). In this situation, a collision with the curtain is allowed, while collisions with rigid obstacles have to be strictly avoided. Therefore, in a sensor-based collision avoidance routine, measurements stemming from the curtain can be ignored, while measurements concerning nearby rigid obstacles have to be taken into account. Thus, the collision detection routine has to figure out whether a measurement is caused by a deformable or by a rigid object. This problem has been tackled in [FSS*09].

(a) (b)

Figure 9.2: Illustration of a planned path in the environment shown in Fig. 9.1. (a) The approach trades off deformation and travel cost and avoids a detour through the upper part of the environment. (b) When the curtains are moved, the most energy-efficient path also changes [FSS*09].

(a) (b) (c)

Figure 9.3: (a)-(c) A robot moves through a curtain along the path planned in Fig. 9.2 (a).

As the positions of deformable objects in the environment are known, the distinction is not too difficult if neither the robot nor any unforeseen dynamic obstacles are close to a deformable object. However, the problem gets worse if the robot is in contact with a deformable object. In this case, the sensor data is influenced by the deformed object, but cannot be ignored safely, as rigid objects can be near and have to be detected. [FSS*09] present a learned sensor model which combines the knowledge about deformable objects in the environment with range scans during deformations. This way, it allows to distinguish between measurements stemming from a deformable object and other measurements while the robot is in contact with a deformable object.

In [FSS*09], the robot employs a laser scanner for the collision detection process. Like the deformation cost, the obtained sensor data does not only depend on the deformable object and the robot's position, but also on the trajectory relative to the object. Thus, the sensor data is collected for several trajectories relative to a deformable object, and the reflected beams of the scanner are labeled manually as "reflected by a rigid object" or "reflected by a deformable object". This allows to establish a sensor model in the presence of deformable objects. During path execution, the sensor model is used to compute the proba-

bility that a measurement is caused by a deformable object, and measurements with a high probability are disregarded in the collision detection routine.

This way, the approach filters the measurements that belong to deformable objects, and the remaining measurement values can be used for a standard collision avoidance routine like [FBT97, MM00]. Fig. 9.4 illustrates that the approach is able to detect a person standing in front of or behind the curtain, and navigates safely in the given environment.

(a) (b) (c)

Figure 9.4: The robot moves along the path planned in Fig. 9.2. (a) A person stands close to the curtain. The robot recognizes the obstacle and stops. (b) After the person has gone away, the robot moves through the curtain. (c) On its way back, the measurements are influenced by the curtain. Nevertheless, the person is recognized and the robot stops.

9.3 Conclusion

In this chapter, the integration of a simulation environment into a mobile navigation system for autonomous robots has been illustrated. In Sec. 9.1, an approach has been summarized that takes the deformation energy into account while maintaining fast path queries. The approach uses precomputed values of the deformation energy for different paths relative to a deformable object and uses weighted average values in the planning process to compute the cost of object deformations, which allows for an online execution. The planning algorithm finds paths that trade off deformation and travel cost, which avoids costly detours or even allows to find paths in environments where rigid planning approaches would fail.

In Sec. 9.2, an approach for the safe navigation in environments with deformable objects has been sketched. It estimates the probability that a measurement is caused by a deformable object by a learned sensor model. This allows the robot the get close to deformable objects, while still being able to detect unforeseen obstacles.

Thus, the application of simulation environments enhances the planning capabilities for autonomous robots in environments with deformable objects, without losing the ability to safely navigate in space.

10 Conclusion

In this thesis, several contributions in the area of acquisition and simulation of deformable objects in Computer Graphics have been presented. In the acquisition process, the global registration of point clouds and the estimation of deformation parameters have been investigated in order to acquire realistic models. Concerning the simulation of deformable objects, a novel inversion handling scheme and the optimized damping approach contribute to the stability of dynamic simulations and allow for larger time steps. Finally, the simulation system has been integrated into a planning approach for autonomous robots that is able to safely navigate in environments containing deformable objects.

Object acquisition

For the acquisition of 3D models, the transformed polynomials approach for the global registration of partially overlapping point clouds has been introduced in Chapter 4. It provides an appropriate initial configuration for further processing by the ICP algorithm, which optimizes the alignment locally. This allows for an automatic registration of depth scans, which is essential for the automatic reconstruction of 3D models of objects. The approach computes surface-approximating polynomials of different objects and finds possible matches based on rotationally invariant features. For matching candidates, the polynomials are transformed into their respective coordinate systems, which allows to compare the surfaces in more detail. This results in more discriminating information than relying on features only and allows the simultaneous registration of several scans of different similarly shaped objects. Further, it has been shown that the transformed polynomials approach locates a nearly-optimal transformation and requires only minor improvements by the ICP algorithm.

Parameter estimation

As the simulation of deformable objects requires appropriately chosen deformation parameters, a parameter estimation approach has been introduced in Chapter 7. It is based on indentation tests to obtain a force-displacement-relation which allows to deduce the physical parameters of the co-rotational

Finite Element Method. Therefore, a mobile robot has been equipped with a force-feedback-sensor and a depth camera which observe the applied forces and the deformed surface. Applying the measured force within a simulation environment allows to adapt the physical parameters by a gradient descent scheme in order to match the simulated and the measured surface. It has been shown that the system is able to capture realistic physical parameters which can be used in the path planning algorithm illustrated in Chapter 9. Thus, it enables an autonomous exploration of environments containing deformable objects. Although the estimation approach has been formulated for the co-rotational Finite Element Method, the general idea can be applied to arbitrary deformation models. Further, an estimation approach for the illumination parameters has been outlined in Chapter 7 to account for an appropriate visual appearance of reconstructed models.

Inversion handling

As the co-rotational Finite Element Method (Chapter 5) does not handle inverted elements correctly, a separate inversion handling scheme has been introduced in Chapter 6. As it is not possible to determine the inversion direction by considering the current deformation state only, the approach is based on the heuristic assumption that elements are as uninverted as possible. Based on this idea, the approach locates the direction that causes minimum movement to uninvert the tetrahedron. Inversions are resolved efficiently within a small number of simulation steps. Moreover, the computed inversion direction can be stored for a consistent processing in subsequent time steps. The approach has been combined with an efficient handling of degenerated elements and improves the robustness and stability of Finite Element simulations. Further, the approach can be implemented within other constitutive models.

Optimized damping

To further improve the stability of dynamic simulations, a novel damping approach has been developed in Chapter 8. The optimized damping is based on the commonly employed spring damping, whose benefits on dynamic simulations have been illustrated. The novel approach further improves the stability of simulations by an iterative application of spring damping. It has been shown that the iterative procedure converges and that the limit can be computed directly, which allows for an efficient calculation of damping forces. The optimized damping approach simplifies the parameter setting, as the damping coefficient can always be chosen within the range between 0 and 1. Moreover, it permits larger time steps in dynamic simulations compared to previous approaches. Further, it has been applied for the fast propagation of external forces like collision or user interaction.

10.1 Outlook

Finally, some ideas for future work within the presented areas are given. Concerning the global registration approach, scaling of objects could be integrated by considering ratios of coefficients. Further applications in object recognition and symmetry detection are also possible areas of ongoing work. Moreover, it could be investigated if the approach could be transferred to non-rigid registration. In the area of parameter estimation, future work could be concerned with the application to different deformation models and inhomogeneous materials. The stability and efficiency of dynamic simulations is still an active area of research. The proposed approaches for inversion handling and damping are general methods that can be integrated in various deformation models. Ideas like modal analysis concentrate on the deformation model to avoid instabilities [MC11]. Also, the combination of different deformation models as in [FSAH11] can lead to stable simulations of ill-shaped meshes. Other approaches try to mix explicit and implicit integration schemes for large time steps [FSH11]. Apart from stability and the choice of the time step, more efficient simulations can also be reached by so called time-critical approaches in collision detection [GST09], where some computation time in dynamic simulations can be saved.

Curriculum vitae

Personal data

Name	Rüdiger Schmedding
Date of birth	April 16, 1982
Place of birth	Emmendingen, Germany

Education

2007-2011	Research assistant and Ph.D. candidate, Albert-Ludwigs-Universität Freiburg
2007	Diploma in Mathematics (Dipl.-Math.)
2003-2007	Studies in Mathematics, Rheinische Friedrich-Wilhelms-Universität Bonn
2001-2003	Studies in Mathematics, Albert-Ludwigs-Universität Freiburg
2001	German Abitur, Goethe-Gymnasium Emmendingen

Publications

[**ST08**] Ruediger Schmedding, Matthias Teschner, **Inversion Handling for Stable Deformable Modeling**, *The Visual Computer*, vol. 24, no. 7-9 (CGI 2008 Special Issue), pp. 625-633, 2008.

[**SGT09**] Ruediger Schmedding, Marc Gissler, Matthias Teschner, **Optimized Damping for Dynamic Simulations**, *Proc. Spring Conference on Computer Graphics*, Budmerice, Slovakia, pp. 205-212, April 23-25, 2009.

[**SGT10**] Ruediger Schmedding, Marc Gissler, Matthias Teschner, **Fast Force Propagation in Dynamic Simulations Using Optimized Damping**, *Journal of Computer Graphics & Geometry*, vol. 12, no.2, pp. 60-77, 2010.

[**SFBT11**] Ruediger Schmedding, Barbara Frank, Wolfram Burgard, Matthias Teschner, **Transformed Polynomials for the Global Registration of Point Clouds**, *Proc. Spring Conference on Computer Graphics*, Viničné, Slovakia, pp. 157-164, April 28-30, 2011.

[**GST09**] Marc Gissler, Ruediger Schmedding, Matthias Teschner, **Time-critical Collision Handling for Deformable Modeling**, *Journal of Computer Animation and Virtual Worlds (CAVW)*, vol. 20, no. 2-3, pp. 355-364, CASA 2009 Special Issue, 2009.

[**FSS*09**] Barbara Frank, Cyrill Stachniss, Ruediger Schmedding, Matthias Teschner, Wolfram Burgard, **Real-world Robot Navigation amongst Deformable Obstacles**, *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 1649-1654, May 12-17, 2009.

[**FSS*10a**] Barbara Frank, Ruediger Schmedding, Cyrill Stachniss, Matthias Teschner, Wolfram Burgard, **Learning Deformable Object Models**

for Mobile Robot Path Planing using Depth Cameras, *Proc. Workshop RGB-D: Science and Systems RSS*, 2010.

[**FSS*10b**] Barbara Frank, Ruediger Schmedding, Cyrill Stachniss, Matthias Teschner, Wolfram Burgard, **Learning the Elasticity Parameters of Deformable Objects with a Manipulation Robot**, *Proc. IEEE / RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1877-1883, 2010.

Bibliography

[ABB*08] Avril S., Bonnet M., Bretelle A.-S., Grédiac M., Hild F., Ienny P., Latourte F., Lemosse D., Pagano S., Pagnacco E., Pierron F.: Overview of Identification Methods of Mechanical Parameters Based on Full-field Measurements. *Experimental Mechanics 48*, 4 (2008), 381–402.

[ABCO*01] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C. T.: Point Set Surfaces. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 21–28.

[ABK98] Amenta N., Bern M., Kamvysselis M.: A New Voronoi-Based Surface Reconstruction Algorithm. In *Proc. of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 415–421.

[AK04] Amenta N., Kil Y. J.: Defining Point-Set Surfaces. *ACM Trans. Graph. 23* (2004), 264–270.

[AMCO08] Aiger D., Mitra N. J., Cohen-Or D.: 4-points Congruent Sets for Robust Pairwise Surface Registration. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–10.

[AMHH08] Akenine-Möller T., Haines E., Hoffman N.: *Real-Time Rendering*, 3rd ed. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[AOW*08] Adams B., Ovsjanikov M., Wand M., Seidel H.-P., Guibas L. J.: Meshless Modeling of Deformable Shapes and their Motion. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 77–86.

[AW01] Adhikari S., Woodhouse J.: Identification of Damping: Part 1, Viscous Damping. *Journal of Sound and Vibration 243*, 1 (2001), 43–61.

[Bat96] Bathe K.-J.: *Finite Element Procedures*. Prentice Hall, 1996.

[BB75] Becker E., Bürger W.: *Kontinuumsmechanik*. Verlag B.G. Teubner, Stuttgart, 1975.

[BB08] Bender J., Bayer D.: Impulse-based simulation of inextensible cloth. In *Computer Graphics and Visualization (CGV 2008) - IADIS Multi Conference on Computer Science and Information Systems* (Amsterdam, Netherlands, 2008).

[BBK09] Bronstein A. M., Bronstein M. M., Kimmel R.: Topology-Invariant Similarity of Nonrigid Shapes. *Int. J. Comput. Vision 81*, 3 (2009), 281–301.

[BBO*09] Bickel B., Bächer M., Otaduy M. A., Matusik W., Pfister H., Gross M.: Capture and Modeling of Non-Linear Heterogeneous Soft Tissue. *ACM Trans. Graph. 28*, 3 (2009), 89:1–89:9.

[BC00] Bourguignon D., Cani M.-P.: Controlling Anistotropy in Mass-Spring Systems. In *Proc. Computer Animation and Simulation* (2000), pp. 113–123.

[BCP*08] Burion S., Conti F., Petrovskaya A., Baur C., Khatib O.: Identifying physical properties of deformable objects by using particle filters. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* (2008), pp. 1112–1117.

[BDB09] Bayer D., Diziol R., Bender J.: Optimized impulse-based dynamic simulation. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Karlsruhe, Germany, 2009), pp. 125–133.

[BDB11] Bender J., Diziol R., Bayer D.: Simulating inextensible cloth using locking-free triangle meshes. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Lyon, France, 2011), pp. 11–17.

[Ben07] Bender J.: Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 225–233.

[BH11] Batty C., Houston B.: A Simple Finite Volume Method for Adaptive Viscous Liquids. In *Proc. of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 111–118.

[BHS03] Bianchi G., Harders M., Székely G.: Mesh Topology Identification for Mass-Spring Models. In *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI '03)* (2003), vol. 1, pp. 50–58.

[BIT09] Becker M., Ihmsen M., Teschner M.: Corotated SPH for Deformable Solids. In *Proc. Eurographics Workshop on Natural Phenomena* (2009), pp. 27–34.

[BK99] Brock O., Khatib O.: High-Speed Navigation Using the Global Dynamic Window Approach. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* (1999), pp. 341–346.

[BLA02] Bayazit O., Lien J.-M., Amato N.: Probabilistic Roadmap Motion Planning for Deformable Objects. In *Proc. of the IEEE Int. Conf. on Robotics & Automation* (2002), pp. 2126–2133.

[Bla04] Blais F.: Review of 20 Years of Range Sensor Development. *J. Electronic Imaging 13*, 1 (2004), 231–240.

[BM92] Besl P., McKay N.: A Method for Registration of 3-D Shapes. *IEEE Trans. PAMI 14*, 2 (1992), 239–256.

[BMR*99] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The Ball-Pivoting Algorithm for Surface Reconstruction. *Transactions on Visualization and Computer Graphics 5*, 4 (1999), 349–359.

[BR02] Bernardini F., Rushmeier H.: The 3D Model Acquisition Pipeline. *Computer Graphics Forum 21*, 2 (2002), 149–172.

[BR07] Brown B. J., Rusinkiewicz S.: Global Non-Rigid Alignment of 3-D Scans. *ACM Trans. Graph. 26*, 3 (2007), 21:1–21:9.

[BSSH04] Bianchi G., Solenthaler B., Székely G., Harders M.: Simultaneous Topology and Stiffness Identification for Mass-Spring Models Based on FEM Reference Deformations. In *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI '04)* (2004), vol. 2, pp. 293–301.

[BT07] Becker M., Teschner M.: Robust and Efficient Estimation of Elasticity Parameters Using the Linear Finite Element Method. In *Proc. of Simulation and Visualization* (2007), pp. 15–28.

[BTFN*08] Brown B. J., Toler-Franklin C., Nehab D., Burns M., Dobkin D., Vlachopoulos A., Doumas C., Rusinkiewicz S., Weyrich T.: A System for High-volume Acquisition and Matching of Fresco Fragments: Reassembling Theran Wall Paintings. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–9.

[BTT09] Becker M., Tessendorf H., Teschner M.: Direct Forcing for Lagrangian Rigid-Fluid Coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503.

[BW98] Baraff D., Witkin A.: Large Steps in Cloth Simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 43–54.

[CB02] Chandrupatla T. R., Belegundu A. D.: *Introduction to Finite Elements in Engineering*, 3rd ed. Prentice Hall, Inc., Upper Saddle River, New Jersey 07458, 2002.

[CBS00] Chen F., Brown G. M., Song M.: Overview of three-dimensional shape measurement using optical methods. *Optical Engineerings 39* (2000), 10–22.

[CDA00] Cotin S., Delingette H., Ayache N.: Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feedback for Surgery Training and Simulation. *The Visual Computer 16*, 8 (2000), 437–452.

[CGC*02] Capell S., Green S., Curless B., Duchamp T., Popović Z.: Interactive Skeleton-Driven Dynamic Deformations. *ACM Trans. Graph. 21* (2002), 586–593.

[CHC98] Chen C.-S., Hung Y.-P., Cheng J.-B.: A Fast Automatic Method for Registration of Partially-Overlapping Range Images. In *Proc. ICCV* (1998), pp. 242–248.

[CHC99] Chen C., Hung Y., Cheng J.: RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images. *IEEE Trans. PAMI 21*, 11 (1999).

[CK05] Choi M. G., Ko H.-S.: Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 91–101.

[CL96] Curless B., Levoy M.: A Volumetric Method for Building Complex Models from Range Images. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 303–312.

[CLH*05] Choset H., Lynch K. M., Hutchinson S., Kantor G. A., Burgard W., Kavraki L. E., Thrun S.: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[CM91] Chen Y., Medioni G.: Object Modeling by Registration of Multiple Range Images. In *Proc. of IEEE Int. Conf. on Robotics and Automation* (1991), pp. 2724–2729.

[CP03] Cazals F., Pouget M.: Estimating Differential Quantities using Polynomial Fitting of Osculating Jets. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 177–187.

[CPP10] Cretu A.-M., Payeur P., Petriu E. M.: Learning and Prediction of Soft Object Deformation Using Visual Analysis of Robot Interactions. In *Proc. of the 6th Int. Conf. on Advances*

in visual computing - Volume Part II (Berlin, Heidelberg, 2010), ISVC'10, Springer-Verlag, pp. 232–241.

[Cur99] Curless B.: From Range Scans to 3D Models. *SIGGRAPH Comput. Graph. 33*, 4 (1999), 38–41.

[CYMTT92] Carignan M., Yang Y., Magnenat-Thalmann N., Thalmann D.: Dressing Animated Synthetic Actors with Complex Deformable Clothes. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), ACM, pp. 99–104.

[CZ92] Chen D., Zeltzer D.: Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method. *Proc. of ACM SIGGRAPH* (1992), 89–98.

[CZ05] Choi A., Zheng Y.: Estimation of Young's modulus and Poisson's ratio of soft tissue from indentation using two different-sized indentors: Finite element analysis of the finite deformation effect. *Medical and Biological Engineering and Computing 43*, 2 (2005), 258–264. 10.1007/BF02345964.

[CZKM98] Chen Y., Zhu Q., Kaufmann A., Muraki S.: Physically-based Animation of Volumetric Objects. In *Proc. of IEEE Computer Animation* (1998), pp. 154–160.

[DBB09a] Diziol R., Bayer D., Bender J.: Simulating almost incompressible deformable objects. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Karlsruhe, Germany, 2009), pp. 31–37.

[DBB09b] Diziol R., Bender J., Bayer D.: Volume conserving simulation of deformable bodies. In *Proceedings of Eurographics* (Munich, Germany, 2009).

[DBB11] Diziol R., Bender J., Bayer D.: Robust real-time deformation of incompressible surface meshes. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), ACM, pp. 237–246.

[DDCB01] Debunne G., Desbrun M., Cani M.-P., Barr A. H.: Dynamic Real-Time Deformations using Space & Time Adaptive Sampling. *Proc. annual conference on Computer Graphics and iterative techniques, ACM SIGGRAPH* (2001), 31–36.

[DJ05] Donner C., Jensen H. W.: Light Diffusion in Multi-Layered Translucent Materials. *ACM Trans. Graph. 24* (2005), 1032–1039.

[DKT95] Deussen O., Kobbelt L., Tücke P.: Using Simulated Annealing to Obtain Good Nodal Approximations of Deformable Bodies. In *Proc. Sixth Eurographics Workshop on Simulation and Animation* (1995), pp. 30–43.

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Proc. of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[DR08] Dahmen W., Reusken A.: *Numerik für Ingenieure und Naturwissenschaftler*, 2nd ed. Springer-Verlag, 2008.

[EGS03] Etzmuss O., Gross J., Strasser W.: Deriving a Particle System from Continuum Mechanics for the Animation of Deformable Objects. *IEEE Transactions on Visualization and Computer Graphics 9* (2003), 538–550.

[EKS03] Etzmuss O., Keckeisen M., Strasser W.: A Fast Finite Element Solution for Cloth Modelling. In *Proc. 11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 244–251.

[ERM*03] Escobar J. M., Rodríguez E., Montenegro R., Montenero G., González-Yuste J. M.: Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Meth. in Appl. Mech. and Eng. 192* (2003), 2775–2787.

[FB81] Fischler M. A., Bolles R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM 24*, 6 (1981), 381–395.

[FBS*08a] Frank B., Becker M., Stachniss C., Burgard W., Teschner M.: Efficient Path Planning for Mobile Robots in Environments with Deformable Objects. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* (2008), pp. 3737–3742.

[FBS*08b] Frank B., Becker M., Stachniss C., Teschner M., Burgard W.: Learning Cost Functions for Mobile Robot Navigation in Environments with Deformable Objects. In *Workshop on Path Planning on Cost Maps at the IEEE International Conference on Robotics and Automation (ICRA)* (2008).

[FBT97] Fox D., Burgard W., Thrun S.: The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics & Automation Magazine 4*, 1 (1997).

[FBTC98] Fox D., Burgard W., Thrun S., Cremers A.: A Hybrid Collision Avoidance Method For Mobile Robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation* (1998).

[FCOS05] Fleishman S., Cohen-Or D., Silva C. T.: Robust Moving Least-squares Fitting with Sharp Features. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 544–552.

[FH86] Faugeras O., Hébert M.: The Representation, Recognition, and Locating of 3D shapes from range data. *The International Journal of Robotics Research 5*, 3 (1986), 27–52.

[Fon09] Fong P.: Sensing, Acquisition, and Interactive Playback of Data-based Models for Elastic Deformable Objects. *Int. J. Rob. Res. 28*, 5 (May 2009), 630–655.

[Fra07] Frank B.: *3D-Simulation von Roboternavigation in Umgebungen mit deformierbaren Objekten.* Masters thesis, University of Freiburg, 2007.

[FSAH11] Fierz B., Spillmann J., Aguinaga I., Harders M.: Maintaining large time steps in explicit finite element simulations using shape matching. *IEEE Transactions on Visualization and Computer Graphics* (2011).

[FSH10] Fierz B., Spillmann J., Harders M.: Stable explicit integration of deformable objects by filtering high modal frequencies. *Journal of WSCG 18*, 1-3 (2010), 81–88.

[FSH11] Fierz B., Spillmann J., Harders M.: Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 257–266.

[FSS*09] Frank B., Stachniss C., Schmedding R., Teschner M., Burgard W.: Real-world Robot Navigation amongst Deformable Obstacles. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* (2009), pp. 1649–1654.

[FSS*10a] Frank B., Schmedding R., Stachniss C., Teschner M., Burgard W.: Learning Deformable Object Models for Mobile Robot Path Planing using Depth Cameras. In *Proc. Workshop RGB-D: Science and Systems RSS* (2010).

[FSS*10b] Frank B., Schmedding R., Stachniss C., Teschner M., Burgard W.: Learning the Elasticity Parameters of Deformable Objects with a Manipulation Robot. In *Proc. IEEE / RSJ Int. Conf. on Intelligent Robots and Systems IROS* (2010), pp. 1877–1883.

[GBT06] Gissler M., Becker M., Teschner M.: Local Constraint Sets for Deformable Objects. In *Proc. Virtual Reality Interactions and Physical Simulations VRIPHYS* (2006), pp. 25–32.

[GCO06] Gal R., Cohen-Or D.: Salient Geometric Features for Partial Shape Matching and Similarity. *ACM Trans. Graph. 25*, 1 (2006), 130–150.

[GM01] Gerthsen C., Meschede D.: *Gerthsen Physik*, 21st ed. Springer-Lehrbuch Series. Springer, 2001.

[GMGP05] Gelfand N., Mitra N. J., Guibas L. J., Pottmann H.: Robust Global Registration. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, pp. 197–206.

[GP07] Gross M., Pfister H.: *Point-based graphics*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2007.

[Gre03] Greve R.: *Kontinuumsmechanik*. Springer, 2003.

[GSCO07] Gal R., Shamir A., Cohen-Or D.: Pose-Oblivious Shape Signature. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 261–271.

[GSLM05] Gayle R., Segars P., Lin M. C., Manocha D.: Path Planning for Deformable Robots in Complex Environments. In *Proc. of Robotics: Systems and Science (RSS)* (2005), pp. 225–232.

[GST09] Gissler M., Schmedding R., Teschner M.: Time-critical Collision Handling for Deformable Modeling. *Journal of Computer Animation and Virtual Worlds (CAVW) 20*, 2-3 (CASA 2009 Special Issue) (2009), 355–364.

[GTT89] Gourret J.-P., Thalmann N. M., Thalmann D.: Simulation of Object and Human Skin Deformations in a Grasping Task. *SIGGRAPH Comput. Graph. 23* (July 1989), 21–30.

[GW05] Georgii J., Westermann R.: Mass-Spring Systems on the GPU. *Simulation Modelling Practice and Theory 13* (2005), 693–702.

[GW08] Georgii J., Westermann R.: Corotated Finite Elements Made Fast and Stable. In *Proc. of the 5th Workshop On Virtual Reality Interaction and Physical Simulation* (2008), pp. 11–19.

[Har67] Hart E. W.: Theory of the tensile test. *Acta Metallurgica 15*, 2 (1967), 351–355.

[HAWG08] Huang Q.-X., Adams B., Wicke M., Guibas L. J.: Non-Rigid Registration Under Isometric Deformations. *Computer Graphics Forum 27*, 5 (2008), 1449–1457.

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface Reconstruction from Unorganized Points. *SIGGRAPH Comput. Graph. 26* (1992), 71–78.

[HES03] Hauth M., Etzmuss O., Strasser W.: Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7 (2003), 581–600.

[HFG*06] Huang Q.-X., Flöry S., Gelfand N., Hofer M., Pottmann H.: Reassembling Fractured Objects by Geometric Matching. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 569–578.

[HFS*01] Hirota G., Fisher S., State A., Lee C., Fuchs H.: An Implicit Finite Element Method for Elastic Solids in Contact. In *Proc. Computer Animation* (2001), pp. 136–254.

[HK06] Hornung A., Kobbelt L.: Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 41–50.

[HKHM72] Hayes W. C., Keer L. M., Herrmann G., Mockros L. F.: A mathematical analysis for indentation tests of articular cartilage. *Journal of Biomechanics 5*, 5 (1972), 541–551.

[HLZ*09] Huang H., Li D., Zhang H., Ascher U., Cohen-Or D.: Consolidation of Unorganized Point Clouds for Surface Reconstruction. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–7.

[Hor87] Horn B. K. P.: Closed-form Solution of Absolute Orientation using Unit Quaternions. *J. Opt. Soc. Amer. A 4*, 4 (1987), 629–642.

[HS04] Hauth M., Strasser W.: Corotational Simulation of Deformable Solids. *Journal of WSCG 12* (2004), 137–145.

[HSO03] Hauser K. K., Shen C., O'Brien J. F.: Interactive Deformation Using Modal Analysis with Constraints. *Graphics Interface* (2003), 247–256.

[HTK*04] Heidelberger B., Teschner M., Keiser R., Müller M., Gross M.: Consistent Penetration Depth Estimation for Deformable Collision Response. In *Proc. Vision, Modeling, Visualization VMV* (2004), pp. 339–346.

[IABT11] Ihmsen M., Akinci N., Becker M., Teschner M.: A Parallel SPH Implementation on Multi-core CPUs. *Computer Graphics Forum 30*, 1 (2011), 99–112.

[IAGT10] Ihmsen M., Akinci N., Gissler M., Teschner M.: Boundary Handling and Adaptive Time-stepping for PCISPH. In *Proc. VRIPHYS* (2010), pp. 79–88.

[IBAT11] Ihmsen M., Bader J., Akinci G., Teschner M.: Animation of Air Bubbles with SPH. In *Int. Conf. on Computer Graphics Theory and Applications GRAPP* (2011), pp. 225–234.

[ITF04] Irving G., Teran J., Fedkiw R.: Invertible Finite Elements For Robust Simulation of Large Deformation. *Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), 131–140.

[JBS06] Jones M. W., Bærentzen J. A., Sramek M.: 3D Distance Fields: A Survey of Techniques and Applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 581–599.

[JH97] Johnson A. E., Hebert M.: Surface Registration by Matching Oriented Points. In *Proc. Int. Conf. On Recent Advances in 3-D Digital Imaging and Modeling* (1997), pp. 121–128.

[JMLH01] Jensen H. W., Marschner S. R., Levoy M., Hanrahan P.: A Practical Model for Subsurface Light Transport. In *Proc. of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 511–518.

[Joh97] Johnson A.: *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[JP99] James D. L., Pai D. K.: ArtDefo: Accurate Real Time Deformable Objects. In *Proc. of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 65–72.

[JP02] James D., Pai D.: Real Time Simulation of Multizone Elastokinematic Models. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* (2002), pp. 927–932.

[JP03] James D. L., Pai D. K.: Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects. *ACM Trans. Graph. 22*, 1 (2003), 47–82.

[Ju09] Ju T.: Fixing Geometric Errors on Polygonal Models: A Survey. *Journal of Computer Science and Technology 24*, 1 (2009), 19–29.

[JZ06] Jain V., Zhang H.: Robust 3D Shape Correspondence in the Spectral Domain. In *Proc. of the IEEE Int. Conf. on Shape Modeling and Applications 2006* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 19–30.

[JZ07] Jain V., Zhang H.: A Spectral Approach to Shape-based Retrieval of Articulated 3D Models. *Computer-Aided Design 39*, 5 (2007), 398–407.

[JZvK07] Jain V., Zhang H., van Kaick O.: Non-Rigid Spectral Correspondence of Triangle Meshes. *Int. Journal on Shape Modeling 13*, 1 (2007), 101–124.

[KAHD10] Kordelas G., Agapito J. D. P.-M., Hernandez J. M. V., Daras P.: State-of-the-art Algorithms for Complete 3D Model Reconstruction. In *"Engage" Summer School* (2010).

[Kan94] Kanatani K.: Analysis of 3-D Rotation Fitting. *IEEE Transactions on pattern analysis and machine intelligence 16* (1994), 543–549.

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson Surface Reconstruction. In *Proc. of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70.

[KC95] Khatib M., Chatila R.: An extended potential field approach for mobile robot sensor-based motions. In *Proc. Int. Conf. on Intelligent Autonomous Systems (IAS'4)* (1995).

[KCATCO*10] Kin-Chung Au O., Tai C.-L., Cohen-Or D., Zheng Y., Fu H.: Electors Voting for Fast Automatic Shape Correspondence. *Computer Graphics Forum 29*, 2 (2010), 645–654.

[KFR04] Kazhdan M., Funkhouser T., Rusinkiewicz S.: Symmetry Descriptors and 3D Shape Matching. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 115–123.

[Kha86] Khatib O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The Int. Journal of Robotics Research 5*, 1 (1986), 90–98.

[KL04] Kajberg J., Lindkvist G.: Characterisation of materials subjected to large strains by inverse modelling based on in-plane displacement fields. *Int. Journal of Solids and Structures 41*, 13 (2004), 3439–3459.

[KLH98] Kavraki L. E., Lamiraux F., Holleman C.: Towards Planning for Elastic Objects. In *Robotics: The Algorithmic Perspective*. A. K. Peters, Natick, MA, 1998, pp. 313–325. Proc. of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[KP11] Kim J., Pollard N. S.: Fast Simulation of Skeleton-Driven Deformable Body Characters. *ACM Trans. Graph. 30*, 5 (2011), 121:1–121:19.

[KS05] Kim J., Srinivasan M.: Characterization of Viscoelastic Soft Tissue Properties from *In Vivo* Animal Experiments and Inverse FE Parameter Estimation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2005*, Duncan J., Gerig G., (Eds.), vol. 3750 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 599–606.

[KSL*96] Kavraki L., Svestka P., Latombe J.-C., , Overmars M.: Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation 12*, 4 (1996), 566–580.

[KVD*01] Kauer M., Vuskovic V., Dual J., Szekely G., Bajka M.: Inverse Finite Element Characterization of Soft Tissues. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, Niessen W., Viergever M., (Eds.), vol. 2208 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 128–136.

[KYOK09] Kim H.-J., Yoo W.-S., Ok J.-K., Kang D.-W.: Parameter identification of damping models in multibody dynamic simulation of mechanical systems. *Multibody System Dynamics 22*, 4 (2009), 383–398.

[Lan01] Lang J.: *Deformable Model Acquisition and Validation*. PhD thesis, The University of British Columbia (Canada), 2001.

[Lat91] Latombe J.-C.: *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[LaV06] LaValle S. M.: *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[LCOL07] Lipman Y., Cohen-Or D., Levin D.: Data-Dependent MLS for Faithful Surface Approximation. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 59–67.

[LCOLTE07] Lipman Y., Cohen-Or D., Levin D., Tal-Ezer H.: Parameterization-free Projection for Geometry Reconstruction. *ACM Trans. Graph. 26*, 3 (2007), 22.

[LEF95] Lorusso A., Eggert D. W., Fisher R. B.: A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proc. of the 1995 British conference on Machine vision (Vol. 1)* (Surrey, UK, UK, 1995), BMVC '95, BMVA Press, pp. 237–246.

[Lev98] Levin D.: The Approximation Power of Moving Least-squares. *Mathematics of Computation*, 224 (1998), 1517–1531.

[Lev04] Levin D.: Mesh-Independent Surface Interpolation. In *Geometric Modeling for Scientific Visualization*, Brunnett G., Hamann B., Mueller K., Linsen L., (Eds.). Springer-Verlag, 2004, pp. 37–50.

[LF09] Lipman Y., Funkhouser T.: Möbius Voting for Surface Correspondence. *ACM Trans. Graph. 28*, 3 (2009), 1–12.

[LG05] Li X., Guskov I.: Multi-scale Features for Approximate Alignment of Point-based Surfaces. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, pp. 217–226.

[LK01] Lamiraux F., Kavraki L. E.: Planning Paths for Elastic Objects under Manipulation Constraints. *Int. Journal of Robotics Research 20*, 3 (2001), 188–208.

[LPC*00] Levoy M., Pulli K., Curless B., Rusinkiewicz S., Koller D., Pereira L., Ginzton M., Anderson S., Davis J., Ginsberg J., Shade J., Fulk D.: The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proc. of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 131–144.

[LPW02] Lang J., Pai D. K., Woodham R. J.: Acquisition of Elastic Models for Interactive Simulation. *The Int. Journal of Robotics Research 21*, 8 (2002), 713–733.

[LSH07] Lloyd B., Szekely G., Harders M.: Identification of Spring Parameters for Deformable Object Simulation. *Visualization*

and Computer Graphics, IEEE Transactions on 13, 5 (2007), 1081–1094.

[LZSCO09] Liu R., Zhang H., Shamir A., Cohen-Or D.: A Part-Aware Surface Metric for Shape Analysis. *Computer Graphics Forum (Special Issue of Eurographics 2009) 28*, 2 (2009), 397–406.

[MBT03] Maciel A., Boulic R., Thalmann D.: Deformable Tissue Parameterized by Properties of Real Biological Tissue. In *IS4TM* (2003), Springer-Verlag, pp. 74–87.

[MC11] Müller M., Chentanez N.: Solid Simulation with Oriented Particles. *ACM Trans. Graph. 30*, 4 (August 2011), 92:1–92:10.

[MDM*02] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable Real-Time Deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM, pp. 49–54.

[MG04] Müller M., Gross M.: Interactive Virtual Materials. In *Proc. of Graphics Interface* (2004), pp. 239–246.

[MGAT09] Metzger M. C., Gissler M., Asal M., Teschner M.: Simultaneous Cutting of Coupled Tetrahedral and Triangulated Meshes and its Appl. in Orbital Rec. *Int. Journal of Computer Assisted Radiology and Surgery 4*, 5 (2009), 409–416.

[MGP06] Mitra N. J., Guibas L. J., Pauly M.: Partial and Approximate Symmetry Detection for 3D Geometry. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 560–568.

[MGP07] Mitra N. J., Guibas L. J., Pauly M.: Symmetrization. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, pp. 63:1–63:8.

[MGPG04] Mitra N. J., Gelfand N., Pottmann H., Guibas L.: Registration of Point Cloud Data from a Geometric Optimization Perspective. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 22–31.

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position Based Dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118.

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless Deformations Based on Shape Matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 471–478.

[MK06] Moll M., Kavraki L. E.: Path Planning for Deformable Linear Objects. *IEEE Transactions on Robotics 22*, 4 (Aug. 2006), 625–636.

[MKN*04] Müller M., Keiser R., Nealen A., Pauly M., Gross M., Alexa M.: Point Based Animation of Elastic, Plastic and Melting Objects. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 141–151.

[MM00] Minguez J., Montano L.: Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)* (2000), pp. 21–26.

[MRO08] Misra S., Ramesh K. T., Okamura A. M.: Modeling of Tool-Tissue Interactions for Computer-Based Surgical Simulation: A Literature Review. *Presence: Teleoper. Virtual Environ. 17*, 5 (October 2008), 463–491.

[NMK*06] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836.

[NvdS01] Nienhuys H.-W., van der Stappen A. F.: A Surgery Simulation Supporting Cuts and Finite Element Deformation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, Niessen W., Viergever M., (Eds.), vol. 2208 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 145–152.

[OBH02] O'Brien J. F., Bargteil A. W., Hodgins J. K.: Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21* (2002), 291–294.

[OH99] O'Brien J. F., Hodgins J. K.: Graphical Modeling and Animation of Brittle Fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 137–146.

[OSG08] Ovsjanikov M., Sun J., Guibas L.: Global Intrinsic Symmetries of Shapes. In *SGP '08: Proceedings of the Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2008), Eurographics Association, pp. 1341–1348.

[Osw09] Oswald E.: *Parameterschätzung für das Phong-Beleuchtungsmodell unter Verwendung von 3D Oberflächenrekonstruktionen.* Bachelor thesis, University of Freiburg, 2009.

[PB88] Platt J. C., Barr A. H.: Constraints Methods for Flexible Models. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 279–288.

[PBL09] Paulsen R. R., Bærentzen J. A., Larsen R.: Regularisation of 3D Signed Distance Fields. In *SCIA '09: Proceedings of the 16th Scandinavian Conference on Image Analysis* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 513–519.

[PDA01] Picinbono G., Delingette H., Ayache N.: Non-linear and anisotropic elastic soft tissue models for medical simulation. *IEEE Int. Conf. Robotics and Automation* (2001).

[PGK02] Pauly M., Gross M., Kobbelt L. P.: Efficient Simplification of Point-Sampled Surfaces. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 163–170.

[Pho75] Phong B. T.: Illumination for Computer Generated Pictures. *Commun. ACM 18*, 6 (June 1975), 311–317.

[PMG*05] Pauly M., Mitra N. J., Giesen J., Gross M., Guibas L. J.: Example-based 3D Scan Completion. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, pp. 23–32.

[PMW*08] Pauly M., Mitra N. J., Wallner J., Pottmann H., Guibas L. J.: Discovering Structural Regularity in 3D Geometry. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–11.

[PO09] Parker E. G., O'Brien J. F.: Real-Time Deformation and Fracture in a Game Environment. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 156–166.

[Pul99] Pulli K.: Multiview Registration for Large Data Sets. In *Proc. of Second Int. Conf. on 3-D Digital Imaging and Modeling* (1999), pp. 160–168.

[PW89] Pentland A., Williams J.: Good Vibrations: Modal Dynamics for Graphics and Animation. *SIGGRAPH Comput. Graph. 23*, 3 (1989), 207–214.

[PWY*07] Pottmann H., Wallner J., Yang Y.-L., Lai Y.-K., Hu S.-M.: Principal Curvatures from the Integral Invariant Viewpoint. *Computer Aided Geometric Design 24*, 8-9 (2007), 428–442.

[RBBK10] Raviv D., Bronstein A. M., Bronstein M. M., Kimmel R.: Full and Partial Symmetries of Non-rigid Shapes. *Int. J. Comput. Vision 89*, 1 (2010), 18–39.

[RCM*01] Rocchini C., Cignoni P., Montani C., Pingi P., Scopigno R.: A low cost 3D scanner based on structured light. *Computer Graphics Forum 20*, 3 (2001), 299–308.

[RHHL02] Rusinkiewicz S., Hall-Holt O., Levoy M.: Real-time 3D Model Acquisition. *ACM Trans. Graph. 21*, 3 (2002), 438–446.

[RJ07] Rivers A. R., James D. L.: FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM Trans. Graph. 26* (2007), 82:1–82:6.

[RL01] Rusinkiewicz S., Levoy M.: Efficient Variants of the ICP Algorithm. *Int. Conf. on 3D Digital Imaging and Modeling* (2001), 145–152.

[RLA06] Rodríguez S., Lien J.-M., Amato N.: Planning Motion in Completely Deformable Environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)* (2006), pp. 2466–2471.

[SA92] Schmidt G., Azarm K.: Mobile Robot Navigation In A Dynamic World Using An Unsteady Diffusion Equation Strategy. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (1992), pp. 642–647.

[SB08] Syllebranque C., Boivin S.: Estimation of Mechanical Parameters of Deformable Solids from Videos. *The Visual Computer 24* (2008), 963–972.

[SBT07] Spillmann J., Becker M., Teschner M.: Non-iterative Computation of Contact Forces for Deformable Objects. *Journal of WSCG 15*, 1-3 (2007), 33–40.

[SFBT11] Schmedding R., Frank B., Burgard W., Teschner M.: Transformed Polynomials for Global Registration of Point Clouds. In *Proc. Spring Conference on Computer Graphics* (2011), pp. 157–164.

[SGN*05] Soza G., Grosso R., Nimsky C., Hastreiter P., Fahlbusch R., Greiner G.: Determination of the elasticity parameters of brain tissue with combined simulation and registration. *Int. Journal of Medical Robotics and Computer Assisted Surgery 1*, 3 (2005), 87–95.

[SGT09] Schmedding R., Gissler M., Teschner M.: Optimized Damping for Dynamic Simulations. In *Proc. Spring Conference on Computer Graphics* (2009), pp. 205–212.

[SGT10] Schmedding R., Gissler M., Teschner M.: Fast Force Propagation in Dynamic Simulations Using Optimized Damping. *Journal of Computer Graphics & Geometry 12*, 2 (2010), 60–77.

[She02] Shewchuk J.: What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measure. In *Proc. of 11th Int. Meshing Roundtable* (2002), pp. 115–126.

[SLS*07] Sharf A., Lewiner T., Shklarski G., Toledo S., Cohen-Or D.: Interactive Topology-aware Surface Reconstruction. *ACM Trans. Graph. 26* (2007), 43.

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and Approximating Implicit Surfaces from Polygon Soup. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 896–904.

[SP09] Solenthaler B., Pajarola R.: Predictive-Corrective Incompressible SPH. *ACM Trans. Graph. 28*, 3 (July 2009), 40:1–40:6.

[SSBT08] Stumpp T., Spillmann J., Becker M., Teschner M.: A Geometric Deformation Model for Stable Cloth Simulation. In *Proc. VRIPHYS* (2008), pp. 39–46.

[SSSCO08] Shalom S., Shapira L., Shamir A., Cohen-Or D.: Part Analogies in Sets of Objects. In *Proc. of Eurographics Symposium on 3D Object Retrieval* (2008), pp. 33–40.

[ST05] Spillmann J., Teschner M.: Contact Surface Computation for Coarsely Sampled Deformable Objects. In *Proc. Vision, Modeling, Visualization VMV'05* (2005), pp. 289–296.

[ST08] Schmedding R., Teschner M.: Inversion Handling for Stable Deformable Modeling. *The Visual Computer 24*, 7-9 (CGI 2008 Special Issue) (2008), 625–633.

[SVAC11] San-Vicente G., Aguinaga I., Celigueta J. T.: Cubical Mass-Spring Model Design Based on a Tensile Deformation Test and Nonlinear Material Model. *IEEE Transactions on Visualization and Computer Graphics* (2011).

[SWT06] Spillmann J., Wagner M., Teschner M.: Robust Tetrahedral Meshing of Triangle Soups. In *Proc. Vision, Modeling, Visualization VMV* (2006), pp. 9–16.

[SZ92] Schnur D. S., Zabaras N.: An inverse method for determining elastic material properties and a material interface. *International Journal for Numerical Methods in Engineering 33*, 10 (1992), 2039–2057.

[TBHF03] Teran J., Blemker S., Hing V. N. T., Fedkiw R.: Finite Volume Methods for the Simulation of Skeletal Muscle. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 68–74.

[TBW*09] Tevs A., Bokeloh M., Wand M., Schilling A., Seidel H.-P.: Isometric Registration of Ambiguous and Partial Data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)* (Miami Beach, Florida, USA, 2009), IEEE Computer Society, pp. 1185–1192.

[TF88a] Terzopoulos D., Fleischer K.: Deformable Models. *The Visual Computer 4* (1988), 306–331.

[TF88b] Terzopoulos D., Fleischer K.: Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 269–278.

[TFBW*10] Toler-Franklin C., Brown B., Weyrich T., Funkhouser T., Rusinkiewicz S.: Multi-feature Matching of Fresco Fragments. *ACM Trans. Graph. 29*, 6 (2010), 185:1–185:12.

[THM*03] Teschner M., Heidelberger B., Müller M., Pomeranets D., Gross M.: Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Proc. Vision, Modeling, Visualization VMV* (2003), pp. 47–54.

[THMG04] Teschner M., Heidelberger B., Müller M., Gross M.: A Versatile and Robust Model for Geometrically Complex Deformable Solids. *Proc. Computer Graphics International* (2004), 312–319.

[TKH*05] Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Fuhrmann A., Cani M.-P., Faure F., Magnenat-Thalmann N., Strasser W., Volino P.: Collision Detection for Deformable Objects. *Computer Graphics Forum 24*, 1 (2005), 61–81.

[TO02] Turk G., O'Brien J. F.: Modelling with Implicit Surfaces that Interpolate. *ACM Trans. Graph. 21*, 4 (2002), 855–873.

[TPBF87] Terzopoulos D., Platt J., Barr A., Fleischer K.: Elastically Deformable Models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM, pp. 205–214.

[TW88] Terzopoulos D., Witkin A.: Physically Based Models with Rigid and Deformable Components. *Computer Graphics and Applications, IEEE 8*, 6 (1988), 41–51.

[Umm08] Ummenhofer B.: *Finite Element Method on Convex Polyhedra in DefCol Studio*. Bachelor thesis, University of Freiburg, 2008.

[VBS09] Vrubel A., Bellon O. R. P., Silva L.: A 3D Reconstruction Pipeline for Digital Preservation. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR* (2009), pp. 2687–2694.

[vG98] van Gelder A.: Approximate Simulation of Elastic Membranes by Triangulated Spring Meshes. *Journal of Graphics, GPU, and Game Tools 3*, 2 (1998), 21–41.

[VGS04] Vachal P., Garimella R. V., Shashkov M. J.: Untangling of 2D meshes in ALE simulations. *Journal of Computational Physics 196* (2004), 627–644.

[vKZHCO11] van Kaick O., Zhang H., Hamarneh G., Cohen-Or D.: A Survey on Shape Correspondence. *Computer Graphics Forum 30*, 6 (2011), 1681–1707.

[WBG07] Wicke M., Botsch M., Gross M.: A Finite Element Method on Convex Polyhedra. *Computer Graphics Forum 26*, 3 (2007), 355–364.

[Wey06] Weyrich T.: *Acquisition of Human Faces Using A Measurement-Based Skin Reflectance Model*. PhD thesis, ETH Zürich, 2006.

[WMP*06] Weyrich T., Matusik W., Pfister H., Bickel B., Donner C., Tu C., McAndless J., Lee J., Ngan A., Jensen H. W., Gross M.: Analysis of Human Faces using a Measurement-Based Skin Reflectance Model. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 1013–1024.

[WMW06] Winkelbach S., Molkenstruck S., Wahl F. M.: Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In *DAGM-Symposium* (2006), pp. 718–728.

[Woo80] Woodham R. J.: Photometric Method for Determining Surface Orientation from Multiple Images. *Optical Engineerings 19*, 1 (1980), 139–144.

[Woo84] Woodham R. J.: *Photometric Method for Determining Shape from Shading*. Tech. rep., University of British Columbia, Vancouver, BC, Canada, Canada, 1984.

[ZCK98] Zhu Q.-h., Chen Y., Kaufman A.: Real-time Biomechanically-based Muscle Volume Deformation using FEM. *Computer Graphics Forum 17*, 3 (1998), 275–284.

[ZCLH09] Zheng Y. P., Choi A. P. C., Ling H. Y., Huang Y. P.: Simultaneous estimation of Poisson's ratio and Young's modulus using a single indentation: a finite element study. *Meas. Sci. Technol. 20*, 4 (2009), 045706:1–045706:9.

[Zha94] Zhang Z.: Iterative Point Matching for Registration of Free-Form Curves and Surfaces. *Int. Journal of Computer Vision 13*, 2 (1994), 119–152.

[ZST*10] Zheng Q., Sharf A., Tagliasacchi A., Chen B., Zhang H., Sheffer A., Cohen-Or D.: Consensus Skeleton for Non-Rigid Space-Time Registration. *Computer Graphcis Forum (Special Issue of Eurographics) 29*, 2 (2010), 635–644.

[ZZ94] Zantout R. N., Zheng Y. F.: Geodesics: A Tool for Solving Material Properties Inverse Problems. In *Proc. of the IEEE International Conference on Industrial Technology* (1994), pp. 391–394.

[ZZ95] Zieliński P., Ziqtak K.: The Polar Decomposition - Properties, Applications and Algorithms. *Matematyka Stoswana 38* (1995), 23–40.