Numerical simulation of compressible flows with immersed boundaries using Discontinuous Galerkin methods

Bachelor-Thesis von Simone Katharina Stange
Tag der Einreichung:

1. Gutachten: Prof. Dr.-Ing.habil. Martin Oberlack
2. Gutachten: Dr.-Ing. Björn Müller

TECHNISCHE UNIVERSITÄT DARMSTADT

Computational Engineering
Fachgebiet für Strömungsdynamik (FDY)

Numerical simulation of compressible flows with immersed boundaries using Discontinuous Galerkin methods

Vorgelegte Bachelor-Thesis von Simone Katharina Stange

1. Gutachten: Prof. Dr.-Ing.habil. Martin Oberlack
2. Gutachten: Dr.-Ing. Björn Müller

Tag der Einreichung:

Abstract

Collision handling has been an active research topic in the area of the physically-based simulation of rigid and deformable bodies for many years. A common approach in interactive environments are discrete penalty forces, computing a repulsion forced based on the penetration at one moment in the time step. They provide low computational costs and good scalability, though they suffer from jitter and instability. Tang et al. [18] improved the approach of discrete penalty forces and introduced 2012 the continuous penalty forces (continuous penalty forces), continuously accumulating penalty forces along the penetration trajectory over the whole time step. Thereby, the jitter and instability issues are reduced. Although, the continuous penalty forces show artifacts especially for enduring contacts, precluding the simulation of sliding contacts.

In this thesis, we present a unified system to handle collisions between rigid and deformable bodies with friction. We modify the integration scheme by Bridson et al. [4] to handle rigid and deformable bodies, apply a CCD [16] and handle the detected collisions with continuous penalty forces [18]. We discuss the artifacts arising from the continuous penalty forces algorithm, examine methods to tackle them and apply the new methods to the continuous penalty forces algorithm. Finally, we analyze the results of the continuous penalty forces algorithm in comparison to discrete penalty forces, evaluate our new algorithm to handle the continuous penalty forces artifacts and inspect further improvements.

Contents

1 Introduction **5**
1.1 Motivation and Task 5
1.2 Structure 5

2 Related work **6**

3 Simulation Physics **8**
3.1 Single Body Dynamics 8
3.2 Interacting with Rigid and Deformable Bodies 11
3.3 Time Integration 11

4 Collision Detection **13**
4.1 Broad and Narrow Phase 13
4.2 Discrete Collision Detection 13
4.3 Continuous Collision Detection 14
4.4 CCD Acceleration with Axis Aligned Bounding Boxes 15

5 Collision Handling with Penalty Methods **17**
5.1 Discrete Penalty Forces 17
5.2 Continuous Penalty Forces 18
5.3 Friction 22

6 Robust handling of enduring contacts **23**
6.1 CPF Artifacts 23
6.2 CPF Extensions 26

7 Results **36**
7.1 Stability 36
7.2 Contact Redefinition 42
7.3 Friction 47
7.4 Computational Cost 48

8 Discussion **52**

Bibliography **53**

List of Figures

3.1 Predictor-Corrector scheme 12

4.1 Illustration of the possible feature pairs, VF and EE, for CCD. 14
4.2 2D Illustration of AABB to accelerate CCD by example of a triangle and a vertex. . 16

5.1 2D Illustration of the resolution force computation for CPF and DPF in the VF case. 19

6.1 2D-Illustration of a VF collision, which is considered colliding after it is already resolved 23
6.2 2D-Illustration of a hooked VF collision with tangential forces induced by the CPF algorithm. 24
6.3 2D-Illustration of a VF collision causing impulses at implausible positions. 25
6.4 2D-Illustration of an incorrect VF collision resolution artifact 25
6.5 2D-Illustration of the classification circle. 27
6.6 Comparison of the contact categorization with normals and normal spans. 28
6.7 2D-Illustration of the vertex normal span $\mathbf{N_v}$ of the vertex v. 28
6.8 Comparison of the contact categorization with normals and normal spans for VF collisions with the vertex in an acute-angled, right-angled and obtuse-angled corner. 29
6.9 Illustration of limitation to of the algorithm to non-deep contacts. 30
6.10 2D-Illustration of a barycentric redefinition. Dots indicate the vertices, the red ones are colliding. 31
6.11 Example of a barycentric redefinition in a simulation. 32
6.12 2D-Illustration of the redefinition for a hooked collision. 34
6.13 2D-Illustration of the redefinition for a hooked collision. 35

7.1 The position over time of a particle m=1.0kg dropped from h=0.1m on a plane with continuous (a, b) and discrete (c, d) penalty forces 38
7.2 The position over time of a particle dropped on a plane with continuous and discrete penalty forces with the same time step $\Delta t = 0.01$ and penalty stiffness $k = 1000$ 39
7.3 Illustration of discretizations of the deformable bar on the rigid cube. 40
7.4 Comparison of the repulsion force between a deformable bar and a rigid cube with varying time steps Δt and and constant penalty coefficients $k = 5000$. 40
7.5 The repulsion force between a deformable bar and a rigid cube with varying time steps dt and penalty coefficients k. 41
7.6 The average of the summed penetration depth per time step for a deformable bar dropped on a rigid cube with varying time steps an penalty coefficients. 42
7.7 The penetration depth for CPF and DPF with varying penalty coefficients and constant time step $\Delta t = 0.01s$. 43
7.8 A rigid cube sliding on a deformable bar without the collision redefinition algorithm and without friction. 44

7.9 A rigid cube sliding on a deformable bar with the collision redefinition algorithm and without friction. 44
7.10 A rigid cube sliding on a deformable bar without the collision redefinition algorithm and without friction. 45
7.11 A rigid cube sliding on a deformable barwith the collision redefinition algorithm and without friction. 45
7.12 Contact force for a rigid cube sliding on a deformable bar 46
7.13 A deformable bunny is dropped in a rigid bowl, which is dropped on a static plane. 46
7.14 Inverted tetrahedrons (marked red) occur when the deformable bunny collides with the rigid bowl. 47
7.15 Position and velocity over time of a rigid cube sliding on a slope with different static friction coefficients μ_s. 48
7.16 A rigid cube sliding from the right to the left on a deformable bar with two friction coefficient combinations. 49
7.17 Composition of the computation time for one feature pair 50
7.18 Comparison of the computation time for one impulse with CPF and DPF 51

1 Introduction

1.1 Motivation and Task

Collision handling has been an active research topic in the area of the physically-based simulation of rigid and deformable bodies for many years with several fields of application, for example, engineering, robotics, animation or games. To gain a realistic behavior, it is important to detect collisions between objects and to respond in a way that keeps the objects well separated. There are two main approaches for these responds: constraint-based and impulse-based. An example for a constraint-based approach is the use of discrete penalty forces (DPF). The forces only depend on the penetration depth and the contact normal at one moment in the time step of the simulation. Therefore, they provide low computational costs and good scalability. In its basic formulation, this approach suffers from several problems. In particular, DPF yield discontinuous forces implying jitter and instability, global inconsistency and interpenetrations.
Continuous penalty forces (CPF) [18] improve this approach and compute collision impulses by continuously accumulating penalty forces along the penetration trajectory over the whole time step and thereby reducing the jitter and instability issues. A drawback of the CPF algorithm is that it shows the artifacts of separated features still assumed colliding, additional artificial collisions through discretization and inconsistent feature pair mapping withdrawing a visually satisfying result. Therefore, it is desirable to enhance CPF to handle these situations properly. There is already an extensive amount of work about consistent penetration depth estimation and contact normal computation, nevertheless the CPF algorithm requires its own contact normal definition. Therefore, we want to apply an algorithm keeping the CPF contact normal definition and providing plausible collision feature couples.
This thesis focuses on analyzing the artifacts arising from the CPF algorithm and examining approaches to tackle them. Furthermore, we compare CPF [18] and DPF when resolving collisions between rigid and deformable bodies in interactive multi body framework and investigate the application of a penalty based friction model [23].
In the results we show that our new approach handles the artifacts arising from the CPF algorithm for non-deep contacts. We show that CPF reduce jitter and instability in comparison to DPF and that the penalty-based friction model provides visually sufficient results.

1.2 Structure

This thesis is structured as follows. In chapter 2 we give an overview on the related work. In chapter 3 we provide an introduction to the physical foundations for the simulation of rigid and deformable bodies and explain the integration scheme. The collision detection is outlined in chapter 4. In chapter 5, we describe how to handle collisions with penalty forces and the penalty-based friction model. Chapter 6 analyzes CPF towards artifacts arising from the CCD/CPF formulation and we investigate approaches how to handle them. Subsequently, we present the results in chapter 7 and give a conclusion and outline future work in chapter 8.

2 Related work

The laws of physics provide an exact and comprehensive description of the physical behavior of bodies. Since such an exact and comprehensive representation of these laws in multi-body-simulations is not feasible, due to the complexity of these descriptions approximations and adaptations with regard to the intended use are necessary. Of great importance is Newton's second law of motion and the derived equation $\mathbf{F} = m\mathbf{a}$ providing a relation between force and acceleration. With this relation it is possible to compute the motion of a particle. For the description of rigid bodies, additionally, the rotation needs to be taken into account, yielding three additional degrees of freedom. The necessary relation between angular acceleration and torque is provided by the Euler-Equation [3].

The simulation of deformable bodies adds various degrees of freedom and multiple approaches to model deformable bodies have been investigated by the computer graphics community. Mainly three approaches have become prevalent: mass-spring systems [11], shape matching [15] and finite elements [14]. Mass-spring systems and shape matching are fast and simple but lack of accuracy in comparison to finite elements. With an implicit integration, finite elements are unconditionally stable [20]. Higher accuracy and more complex deformations for finite elements can be obtained by applying quadratic bases as described by Weber et al. [20]. To increase the performance, there is work on the acceleration with efficient GPU data structures by Weber et al. [21]. With these enhancements, finite elements provide fast and stable deformable bodies with the ability to model complex material behavior.

In order to simulate multiple bodies interacting and colliding with each other, it is necessary to detect the collision with a collision detection algorithm and to handle them with a collision handling algorithm. There is a large diversity in collision detection algorithms, varying in necessary pre-processing, provided collision information, performance and processable models. A survey focusing on the detection for different geometric models is provided by Lin and Gottschalk [10] and a survey focusing on deformable bodies is provided by Teschner et al. [19]. Discrete algorithms check for collisions at a specific moment in a time step, though this can lead to missing collisions. Continuous algorithms, as described by Provot [16], take the bodies' approximated trajectory into account. Hence, no collisions are missed and the contact times can be provided.

After the collision has been detected, it is handled in order to restore a penetration free state. Mainly two collision handling approaches have become prevalent in the community: impulse-based [3] and constraint-based methods [13]. Impulse based methods compute impulses which directly change the velocity and by integration modify the positions. They provide an accurate collision response for rigid bodies, but can not be easily extended for deformable bodies. Constraint-based methods define constraints and based on the violation of the constraints contact forces or impulses are computed. A common constraint-based approach are penalty methods.

DPF compute the contact forces only depending on the penetration depth and the contact normal at one moment in the time step. Therefore, they provide low computational cost and good scalability. In its basic formulation, this approach suffers from several problems. Espe-

cially, DPF yield discontinuous forces implying jitter, instability and global inconsistency and interpenetrations.
Continuous penalty forces [18] extend the idea of DPF and compute collision impulses by continuously accumulating penalty forces along the penetration depth over the whole time step and thereby reducing the jitter and instability issues.
Computing consistent penetration depths and contact normals is a closely related topic. An approach for deformable bodies with deep intersections was presented by Keiser et al. [9] and a continuous formulation was investigated by Zhang et al. [24]. The CPF algorithm provides its own definition of contact normal and depth [18].
The collision handling algorithms do not include friction. Friction models are generally based on Coulomb's Friction law, differentiating between dynamic and static friction. Dynamic friction slows down the tangential movement, whereas static friction stops any tangential movement. Static friction is only applied if the tangential velocity is zero and the tangential force smaller than the product of the normal force and the static friction coefficient. A penalty-based friction model is described by Yamane and Nakamura [23], providing a straightforward integration into a penalty based collision resolution, only small computational overhead and a unified handling of rigid and deformable bodies.
The previous presented methods are connected in the integration scheme. An integration scheme approach for the treatment of collision and friction for cloth animation was proposed by Bridson et al. [4]. It can be adapted for the simulation of rigid and deformable bodies with CPF.
In our work we apply finite elements with quadratic bases for the simulation of the deformable bodies [20]. The collisions are detected with CCD [16] and handled with CPF [18], friction is modelled with a penalty-based algorithm [23]. The integration scheme is similar to Bridson et al. [4] with adaptions for the simulation of rigid and deformable bodies with CPF.

3 Simulation Physics

This chapter outlines the physical principles and relevant formulas for the simulation of rigid and deformable bodies. It is essential to understand the modeling of rigid and deformable bodies in order to investigate possibilities to interact with the bodies for collision handling.
We first give a brief overview over the simulation of single bodies and afterwards describe the simulation of rigid and deformable bodies in detail. Subsequently, we discuss how to interact with the bodies for collision handling and outline the integration scheme.

3.1 Single Body Dynamics

The laws of physics provide an exact and comprehensive description of the physical behavior of bodies. Since an exact and comprehensive representation of these laws in multi-body-simulations is not feasible, due to the complexity of these descriptions approximations and adaptations with regard to the intended use are necessary.
In this work, we simulate different types of bodies. We simulate rigid and deformable bodies, additionally we use geometry that does not move, which we call static bodies. Non-static bodies change their motion due to the effects of forces, described by Newton's second law of motion. The way how these changes take place depends on whether the body is deformable or rigid. In contrast to rigid bodies, deformable bodies also change their shape in response to applied forces [3].

3.1.1 Rigid Body Dynamics

A rigid body is an idealized model of a solid body in which deformation is neglected. Thereby, it can by described by only six degrees of freedom: translations along three axes of space and rotations around these three axes [3].
The translation is determined by the body's mass m, position $\mathbf{s}(t)$ and velocity $\mathbf{v}(t)$. The rotation is determined by the body's inertia tensor $\mathbf{J}$, the rotation matrix $\mathbf{R}(t)$ and the angular velocity $\boldsymbol{\omega}(t)$. The translational movement is influenced by the exterior forces $\mathbf{f}_{ext}$ and can be expressed by a system of linear first order differential equations

$$\dot{\mathbf{v}} = \frac{\mathbf{f}_{ext}}{m} \tag{3.1}$$

$$\dot{\mathbf{x}} = \mathbf{v} \tag{3.2}$$

For constant forces, as it is the case within a simulation time step Δt, the system can be solved analytically and no numerical integration is necessary. The velocity can be computed by

$$\mathbf{v}(t_0 + \Delta t) = \mathbf{v}(t_0) + \int_0^{\Delta t} \dot{\mathbf{v}} dt = \mathbf{v}(t_0) + \frac{1}{m}\mathbf{f}_{ext}\Delta t. \tag{3.3}$$

The new position is determined by

$$\mathbf{x}(t_0+\Delta t) = \mathbf{x}(t_0) + \int_0^{\Delta t} \dot{\mathbf{x}} dt = \mathbf{x}(t_0) + \int_0^{\Delta t} \mathbf{v}(t_0) + \frac{1}{m}\mathbf{f}_{ext} t \, dt \tag{3.4}$$

$$= \mathbf{x}(t_0) + \mathbf{v}(t_0)\Delta t + \frac{1}{2m}\mathbf{f}_{ext}\Delta t^2. \tag{3.5}$$

The rotational movement is influenced by the exterior torque $\boldsymbol{\tau}_{ext}$. The change in angular velocity can be expressed by the Euler-Equation

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\boldsymbol{\tau}_{ext} - [\boldsymbol{\omega}(t) \times [\mathbf{J}\boldsymbol{\omega}(t)]]). \tag{3.6}$$

The Euler-Equation is handled by a transformation into the local coordinate system, in which $\mathbf{J}$ is constant and then solved with a numerical integration method. The change in rotation can be expressed in terms of $\mathbf{R}(t)$

$$\dot{\mathbf{R}}(t) = \boldsymbol{\omega}^*(t)\mathbf{R}(t). \tag{3.7}$$

with $\boldsymbol{\omega}^*(t)$ denoting the cross product matrix of $\boldsymbol{\omega}(t)$.
However, representing the rotation with a rotation matrix $\mathbf{R}$ is redundant, since nine values are used to express three degrees of freedom. Furthermore, the formulation is numerically unstable. Alternatively, the rotation can be expressed in a more stable and less redundant way with unit quaternions

$$\dot{\mathbf{q}}(t) = \frac{1}{2}\boldsymbol{\omega}(t) \cdot \mathbf{q}(t). \tag{3.8}$$

A unit quaternion is a normalized 4-element vector and the above used notation is the short form of $(0, \omega_x(t), \omega_y(t), \omega_z(t)) \cdot \mathbf{q}(t)$ [3]. Thus, it is less redundant and more stable. Both equations 3.7 and 3.8 can be solved using numerical integration.
Furthermore, impulses can be applied to the body. For an impulse $\mathbf{I}$ the relation

$$\mathbf{I} = m\mathbf{v} \tag{3.9}$$

applies and similarly for a rotational impulse $\mathbf{l}$

$$\mathbf{l} = \mathbf{J}\boldsymbol{\omega}. \tag{3.10}$$

The previously presented equations apply for centered impulses and forces. If a non-centred force $\mathbf{f}$ is applied, the rotational torque component $\boldsymbol{\tau}$ computes as

$$\boldsymbol{\tau} = \mathbf{d} \times \mathbf{f} \tag{3.11}$$

with the distance vector $\mathbf{d}$ from the center of mass to the point of application. Similarly the rotational impulse $\mathbf{l}$ for a non-centered impulse $\mathbf{I}$ computes as

$$\mathbf{l} = \mathbf{d} \times \mathbf{I}. \tag{3.12}$$

In our framework, the rigid body dynamics are computed with the open source Bullet physics library [6]. Nevertheless, it is important to introduce the formulas as a basis for the collision handling.

3.1.2 Deformable Body Dynamics

Deformable bodies change their shape in response to applied forces, which makes them more complex to simulate than rigid bodies. In our framework, deformable bodies are modeled with finite elements with quadratic bases for the simulation [20]. Nevertheless, as a basis for the collision handling it is sufficient to examine linear finite elements [14]. The method of finite elements for deformable bodies is based on the Lagrange equation for the dynamic behavior of an object. The Lagrange equation can be formulated for the position $\mathbf{x}(t)$ as

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}(\mathbf{x} - \mathbf{x_0}) = \mathbf{f_{ext}}, \tag{3.13}$$

with the first and second order derivatives of the position vector $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$, the mass matrix $\mathbf{M}$, the damping matrix $\mathbf{C}$, the stiffness matrix $\mathbf{K}$ and the external forces $\mathbf{f}_{ext}$. For a system with n vertices, equation 3.13 gives a coupled system of $3n$ linear ordinary differential equations. To compute the stiffness matrix $\mathbf{K}$, we need to understand how the stiffness matrix $\mathbf{K}_e$ for a single tetrahedron is computed.
For a continuous object, the deformation of a point $\mathbf{y} \in \mathbb{R}^3$ is defined by a vector field $\mathbf{u}(\mathbf{y})$ as

$$\mathbf{x}(\mathbf{y}) = \mathbf{y} + \mathbf{u}(\mathbf{y}), \tag{3.14}$$

with the deformed position $\mathbf{x}(\mathbf{y})$. With a linear finite element approach, the deformation is linearly interpolated between the four vertices of the tetrahedron as

$$\mathbf{u}(\mathbf{y}) = \mathbf{H}_e(\mathbf{y})\hat{\mathbf{u}}, \tag{3.15}$$

with the shape function $\mathbf{H}_e(\mathbf{y})$ and the displacement vector $\hat{\mathbf{u}}$, containing the displacement vectors of the four tetrahedron vertices. Using Cauchy's linear strain tensor [5] the strain $\boldsymbol{\varepsilon}$ is

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{y}} + \frac{\partial \mathbf{u}}{\partial \mathbf{y}}^T\right) = \mathbf{B}_e\hat{\mathbf{u}}, \tag{3.16}$$

with the constant matrix $\mathbf{B}_e$ containing derivations of $\mathbf{H}_e(\mathbf{y})$. Applying Hooke's law, the stress $\boldsymbol{\sigma}$ within an element is

$$\boldsymbol{\sigma} = \mathbf{E}\boldsymbol{\varepsilon} = \mathbf{E}\mathbf{B}_e\hat{\mathbf{u}}, \tag{3.17}$$

with the constant matrix $\mathbf{E}$ describing the material's elasticity. For isotropic materials, $\mathbf{E}$ only depends on two scalars, namely the Young's modulus and the Poisson's ratio.
Finally, the stiffness matrix $\mathbf{K}_e$ for one tetrahedron results in

$$\mathbf{f}_e = \mathbf{V}_e\mathbf{B}_e^T\mathbf{E}\mathbf{B}_e\hat{\mathbf{u}} = \mathbf{K}_e\hat{\mathbf{u}}, \tag{3.18}$$

with the tetrahedron's volume $\mathbf{V}_e$ and the elastic forces $\mathbf{f}_e$. Assembling all matrices $\mathbf{K}_e$ gives the final stiffness matrix $\mathbf{K}$.
Now equation 3.13 can be solved with time integration. We use implicit Euler integration, since it provides unconditional stability [20], what is a crucial property in interactive systems.

3.2 Interacting with Rigid and Deformable Bodies

To handle collisions we need to act upon the bodies' dynamics in order to counteract the penetrations.
As we have seen in the previous two sections, the movement of rigid and deformable bodies can be influenced by applying external forces, which directly apply on the acceleration. Furthermore, it is possible to change the velocities by applying impulses or to modify the positions directly.
In the simulation we split the integration in two steps. In the first step, the so called velocity step, we compute the new velocities according to the bodies' interior dynamics and the external forces as presented in section 3.1.1 and section 3.1.2. Afterwards, in the second step, the so called position step, the new positions are computed based on the new velocities. Therefore, it is reasonable to apply additional forces before the velocity step, to modify velocities by applying impulses between the velocity and position step and to modify positions after the position step.
In this work, we apply impulses as suggested by Tang et al. [18], since the result of the CPF algorithm are impulses.
Applying impulses on rigid bodies is straightforward, see section 3.1.1. For deformable bodies applying impulses is more complex. In contrast to rigid bodies, deformable bodies do not have one global velocity, but each vertex has its own velocity and its associated mass. Therefore, we can apply impulses at the vertices, analogously to the formula for centered impulses for rigid bodies, see equation 3.9. If the point of application is not a vertex it is necessary to determine the triangle including the point of application. Thereby, the barycentric weights can be computed and the impulse is divided on the triangle vertices according to the barycentric weights [18].

3.3 Time Integration

To step the simulation forward in time, we separate the bodies' internal dynamics from the collision processing algorithm similarly to Bridson et al. [4]. The collisions are handled in a predictor-corrector fashion (see figure 3.1). We first predict the velocities (1) and positions at the end of the time step, then we use the predicted information to check for collisions (2). If collisions are detected, we apply the collision resolution algorithm to compute the repulsion impulses and corrected velocities (3) and use the corrected velocities to compute the final positions (4).
In contrast to Bridson we do not check for proximity since we do not differ between collision and contact and apply friction together with the repulsion impulses. So the integration scheme can be expressed as

1. Advance simulation by one time step Δt and set: $t^{n+1} = t^n + \Delta t$
2. Predict new candidate velocities $\mathbf{v}_p^{n+1}$ according to external forces and bodies' interior dynamics
3. Check linear trajectories for collisions between t^n and t^{n+1} using the position from the previous step $\mathbf{x}^n$ and $\mathbf{v}_p^{n+1}$
4. Repeat until no new collisions appear, maximum 5 iterations
   4.1. Compute repulsion and friction impulses for unhandled collisions

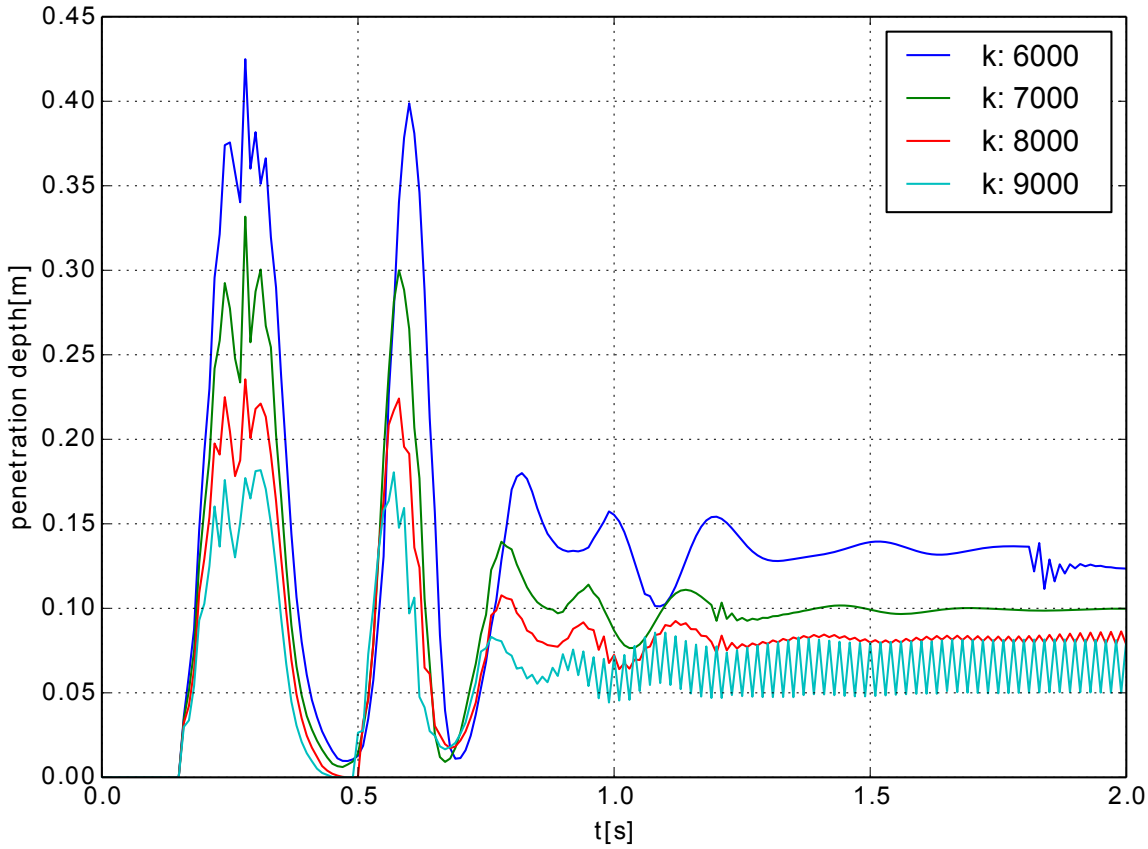


Figure 3.1: Illustration of the predictor-corrector scheme showing a particle colliding with a plane. 1) Prediction of the new velocity 2) Check for collisions along the new trajectory 3) Applying correction impulses yields the final velocity 4) Position update

4.2. Update $\mathbf{v}_p^{n+1}$ with the repulsion and friction impulses

4.3. Check linear trajectories for collisions between t^n and t^{n+1} using the position from the previous step $\mathbf{x}^n$ and $\mathbf{v}_p^{n+1}$

5. Update velocity $\mathbf{v}^{n+1} = \mathbf{v}_p^{n+1}$

6. Update positions $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}$

7. Check for collision resolution and for redefinition

8. Rendering

4 Collision Detection

To provide a plausible accurate collision handling, it is essential to determine all colliding elements. Information about occurring collisions are provided by a collision detection algorithm. We will present two different algorithms for computing intersections between triangle meshes. We first introduce a discrete collision test which tests for collisions at one moment in the time step. Afterwards, we extend the algorithm to a continuous formulation, by taking the feature trajectories into account. Thereby, we do not miss any collisions and additionally provide the contact times which are required for the CPF algorithm. Finally, we have a look at how the collision detection can be sped up with bounding volume hierarchies.

4.1 Broad and Narrow Phase

To check two bodies for intersections with elementary collision tests requires to test each element of the first body against all elements of the second body. Therefore, it scales in $O(nm)$ with n and m corresponding to the numbers of the surface vertices of the objects. Since the amount of necessary tests is huge and performing all of them is expensive, most collision detections are divided into two stages. The first stage, called "Broad Phase", uses a fast method to narrow the possibly colliding feature pairs down. The second stage, called "Narrow Phase" applies an elemental test only to the remaining feature pairs. The division in these two steps provides a significant speed-up [3]. In this thesis only a narrow phase is used.

4.2 Discrete Collision Detection

An elementary discrete collision detection (DCD) test is to test for triangle-triangle collisions [12], which can be reduced to edge-triangle tests (based on [1]). All triangles of a body A are tested against the edges of a body B and vice versa. For a triangle T with the vertices $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, the plane t defined by the triangle T can be expressed as

$$t(\omega_b, \omega_c) = (1 - \omega_b - \omega_c)\mathbf{a} + \omega_b \mathbf{b} + \omega_c \mathbf{c}, \tag{4.1}$$

with the parameters ω_b and ω_c corresponding to the barycentric coordinates.
For an edge E with the vertices $\mathbf{p}$ and $\mathbf{q}$ the line e defined by the edge E is

$$e(\omega_q) = \mathbf{p} + \omega_q(\mathbf{q} - \mathbf{p}), \tag{4.2}$$

with the parameter ω_q corresponding to the barycentric coordinate. The edge E and the triangle T collide if and only if $\exists \omega_q, \omega_b, \omega_c \in [0,1]$ such that

$$\mathbf{p} + \omega_q(\mathbf{q} - \mathbf{p}) = (1 - \omega_b - \omega_c)\mathbf{a} + \omega_b \mathbf{b} + \omega_c \mathbf{c}. \tag{4.3}$$

If there exists a solution to the linear system 4.3, the plane and the line intersect. To ensure that the intersection is in the edge interval of the line, ω_q, ω_b and ω_c need to be in the interval $[0,1]$.
To solve equation 4.3 we transform it into a matrix formulation

$$\mathbf{p}-\mathbf{a}=\begin{pmatrix}\mathbf{p}-\mathbf{q} & \mathbf{b}-\mathbf{a} & \mathbf{c}-\mathbf{a}\end{pmatrix}\begin{pmatrix}\omega_q\\ \omega_b\\ \omega_c\end{pmatrix} \tag{4.4}$$

which can be solved using Cramer's rule.

4.3 Continuous Collision Detection

Continuous collision detection (CCD) algorithms do not only detect whether two features are colliding but also when they are colliding, which is a property required by the CPF algorithm. Most CCD algorithms approximate the motion linearly, as described by Provot [16], but higher order formulations are also possible and provide a more accurate estimation of the collision times.
For triangle meshes there are only two possible elementary collisions. Either a vertex goes through a triangle face resulting in a vertex-face (VF) collision or an edge goes through another edge resulting in an edge-edge (EE) collision, as shown in figure 4.1.
We now describe how to compute collisions for both cases.

(a) Vertex-face case (b) Edge-edge case

Figure 4.1: Illustration of the possible feature pairs, VF and EE, for CCD.

Vertex-Face

For a moving vertex $\mathbf{p}(t)$ and a moving triangle T with the vertices $\mathbf{a}(t)$, $\mathbf{b}(t)$ and $\mathbf{c}(t)$ the constant velocities associated to the linear motion are denoted as $\mathbf{v}_p$, $\mathbf{v}_a$, $\mathbf{v}_b$ and $\mathbf{v}_c$. For t in the interval $[t_0, t_0+\Delta t]$, the vertex trajectories are $\mathbf{p}(t)=\mathbf{p}_0+t\mathbf{v}_p$, $\mathbf{a}(t)=\mathbf{a}_0+t\mathbf{v}_a$, $\mathbf{b}(t)=\mathbf{b}_0+t\mathbf{v}_b$ and $\mathbf{c}(t)=\mathbf{c}_0+t\mathbf{v}_c$. The plane $f(t)$ defined by the triangle T at the time t can be expressed as

$$f(\omega_b,\omega_c,t)=(1-\omega_b-\omega_c)\mathbf{a}(t)+\omega_b\mathbf{b}(t)+\omega_c\mathbf{c}(t). \tag{4.5}$$

The vertex $\mathbf{p}(t)$ and the triangle T collide, if at one point in the time step $\mathbf{p}(t)$ is within T, which holds if and only if $\exists t\in[t_0,t_0+\Delta t]$, $\exists\ \omega_b,\omega_c\in[0,1]$, $\omega_b+\omega_c\leq 1$ such that

$$\mathbf{p}_0+t\mathbf{v}_p=(1-\omega_b-\omega_c)\mathbf{a}(t)+\omega_b\mathbf{b}(t)+\omega_c\mathbf{c}(t). \tag{4.6}$$

In contrast to the DCD the resulting equation system is non linear due to the new time dependency. Therefore, we introduce the additional condition that at the time of collision, the vertex $\mathbf{p}(t)$ is coplanar to the triangle T. The additional condition is satisfied if $\mathbf{p}(t)-\mathbf{a}(t)$ is perpendicular to the triangle normal $\mathbf{n}_t(t) = (\mathbf{b}(t)-\mathbf{a}(t)) \times (\mathbf{c}(t)-\mathbf{a}(t))$ giving

$$(\mathbf{p}(t)-\mathbf{a}(t)) \cdot \mathbf{n}_t(t) = \mathbf{0}. \tag{4.7}$$

This equation gives a third order polynomial with up to three real solutions t_i. Any real t_i in the interval $[t_0, t_0 + \Delta t]$ will be inserted into condition 4.6, yielding a system of linear equations. Any t_i satisfying both conditions is considered as a moment of collision.

Edge-Edge

For a two moving edges E_{ab} and E_{cd} with the corresponding vertices $\mathbf{a}(t)$, $\mathbf{b}(t)$, $\mathbf{c}(t)$ and $\mathbf{d}(t)$ the constant velocities are $\mathbf{v}_a$, $\mathbf{v}_b$ $\mathbf{v}_c$ and $\mathbf{v}_d$. For t in the interval $[t_0, t_0 + \Delta t]$ the vertex trajectories are $\mathbf{a}(t) = \mathbf{a}_0 + t\mathbf{v}_a$, $\mathbf{b}(t) = \mathbf{b}_0 + t\mathbf{v}_b$, $\mathbf{c}(t) = \mathbf{c}_0 + t\mathbf{v}_c$ and $\mathbf{d}(t) = \mathbf{d}_0 + t\mathbf{v}_d$. The lines $e_{ab}(t)$ and $e_{cd}(t)$ defined by E_{ab} and E_{cd} can be expressed as

$$e_{ab}(\omega_b, t) = \mathbf{a}(t) + \omega_b(\mathbf{b}(t)-\mathbf{a}(t)) \tag{4.8}$$
$$e_{cd}(\omega_d, t) = \mathbf{c}(t) + \omega_d(\mathbf{d}(t)-\mathbf{c}(t)) \tag{4.9}$$

with the barycentric weights ω_b and ω_d.
Edge $E_{ab}(t)$ and edge $E_{cd}(t)$ collide if and only if $\exists t \in [t_0, t_0 + \Delta t]$, $\exists\ \omega_b, \omega_d \in [0,1]$ such that

$$\mathbf{a}(t) + \omega_b(\mathbf{b}(t)-\mathbf{a}(t)) = \mathbf{c}(t) + \omega_d(\mathbf{d}(t)-\mathbf{c}(t)) \tag{4.10}$$

Again this gives a non linear system and a further equation is necessary. At the time of collision, both edges are coplanar, which is fulfilled if the cross product of the edge vectors is perpendicular to an arbitrary vector from E_{ab} to E_{ac}, which can be written

$$((\mathbf{b}(t)-\mathbf{a}(t)) \times (\mathbf{d}(t)-\mathbf{c}(t))) \cdot (\mathbf{c}(t)-\mathbf{a}(t)) = \mathbf{0} \tag{4.11}$$

Again this equation gives a third order polynomial with up to three solutions t_i. Any t_i in the interval $[t_0, t_0 + \Delta t]$ will be inserted into condition 4.10, yielding a system of linear equations. Any t_i satisfying both conditions is considered as a moment of collision.

4.4 CCD Acceleration with Axis Aligned Bounding Boxes

The elementary collision tests for CCD require more than 300 floating point operations per feature pair [8]. In order to speed up the CCD, we apply axis aligned bounding boxes (AABB), since they provide a small computational overhead and narrow the necessary elementary tests. For CCD AABB span cuboids around the object primitives' trajectories, with the cuboid edges aligned to the coordinate axes [16], see figure 4.2. To test for intersections between two AABB, the minimum and maximum values for each axis are sorted in ascending order. As long as for at least one axis the minimum and maximum coordinates for both AABB are consecutive there is no intersection. Otherwise, the AABB intersect and the elementary test needs to be applied.
We do not apply a hierarchical order of the AABB, as it is common used for bounding volume hierarchies [19].

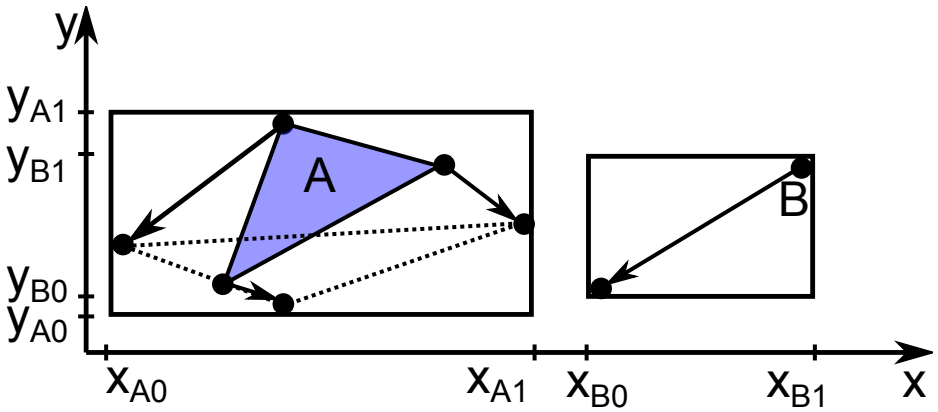


Figure 4.2: 2D Illustration of AABB to accelerate CCD by example of a triangle and a vertex. The arrows indicate the trajectory of the vertices within the time step. The boxes surrounding the features are the according AABB. The minimum and maximum coordinates for both AABB are consecutive in the x-direction. Therefore, there is no intersection.

5 Collision Handling with Penalty Methods

This chapter describes the collision handling with penalty methods. We will first introduce the DPF and then extend the algorithm to a continuous formulation the CPF, taking the whole feature trajectory into account and thereby increasing stability and reducing jitter. Afterwards, we present the penalty-based friction formulation.
Penalty methods handle constraints in a multi body system by applying forces opposed to the constraint violation to correct the velocity and position of the object. The strength of the force depends on the severeness of the constraint violation [3] and can be compared to a spring taking effect until the constraint is complied.

5.1 Discrete Penalty Forces

The method of DPF defines a force field based on the penetration depth of two intersecting bodies. There are multiple definitions for penetration depth and the force field computation, we use the formulation presented by Tang et al. [18]. For a point $\mathbf{x}$, belonging to object A and intersecting object B, the penetration depth $\delta(\mathbf{x})$ is defined as the distance to the closest point on the surface of B. Based on the penetration depth and the penalty coefficient k, the penalty energy is defined by

$$E(\mathbf{x}) = \frac{1}{2}k\delta(\mathbf{x})^2. \tag{5.1}$$

The formulation is similar to the energy $E_s(\mathbf{x})$ of a spring system [7]

$$E_s(\mathbf{x}) = \frac{1}{2}c\ ||\mathbf{x}-\mathbf{x_0}||^2, \tag{5.2}$$

with the initial position $\mathbf{x}_0$ and the spring stiffness c. The penalty coefficient k corresponds to the spring stiffness c and the penetration depth $\delta(\mathbf{x})$ corresponds to the spring expansion $||\mathbf{x}-\mathbf{x_0}||$. The penalty coefficient k needs to be chosen in an empirical manner. According its analogy to the spring stiffness, a similar behaviour can be observed. High penalty coefficients provide large contact forces and small intersections, however they also lead to unstable systems. On the other hand low penalty coefficients provide more stable systems with smaller forces but larger intersections. There is no exact rule how to choose the penalty coefficient, although we will show later (see section 7.1) that the relation $\Delta t \sim \sqrt{\frac{m}{k}}$ can be used as a rule of thumb.
According to the mechanical law of energy conservation [22] the negative gradient of the penalty energy is the penalty force

$$F(\mathbf{x}) = -\nabla E(\mathbf{x}) = -k\delta(\mathbf{x})\nabla\delta(\mathbf{x}). \tag{5.3}$$

With the definition of the penetration depth given above, the negative gradient of the penetration depth is equal to the unit surface normal $\mathbf{n}$ at the closest point on the surface of the penetrated object. The penalty force results in

$$F(\mathbf{x}) = k\delta(\mathbf{x})\mathbf{n} \tag{5.4}$$

The impulse caused by DPF during a time step Δt is the integral along time according to Newton's second law of motion

$$I = \int_{t_i}^{t_i+\Delta t} F(\mathbf{x})dt = \Delta t F(\mathbf{x}) = \Delta t k \delta(\mathbf{x})\mathbf{n}. \tag{5.5}$$

The impulse can be applied to the vertices as described in section 3.2.

5.2 Continuous Penalty Forces

This section outlines the algorithm CPF presented by Tang et al. [18], which reduces instability and jitter in comparison to DPF and is the basis for this thesis. The general impulse formula for the vertex-face and edge-edge case will be derived and the exact formula for the case of linear interpolated trajectories will be presented.

5.2.1 CCD and Penetration Depth

To simplify the collision response with CPF, each simulation time step is normalized to the unit interval $[0,1]$. The collisions are detected using a CCD algorithm, as described in section 4.3, considering all possible pairwise features, VF and EE. Furthermore, the CCD provides the contact times for each feature pair and the collision points $\mathbf{p}(t)$ and $\mathbf{q}(t)$ at the first time-of-contact. The points $\mathbf{p}(t)$ and $\mathbf{q}(t)$ are traced along their trajectories until the collision is resolved.
In contrast to DPF, the penetration depth is defined by a formulation utilizing the information provided by CCD. A contact normal is defined for each contact pair. In the VF case, the time-dependent contact normal $\mathbf{n}(t)$ is the normal of the colliding triangle. In the EE case, the time-dependent contact normal $\mathbf{n}(t)$ is the cross product of the two edge vectors. The penetration depth is defined as the vector between the two contact points $\mathbf{p}(t)$ and $\mathbf{q}(t)$ projected onto the contact normal

$$\delta(t) = \mathbf{n}(t) \cdot (\mathbf{p}(t) - \mathbf{q}(t)) \tag{5.6}$$

A feature pair is considered colliding, from the first time of contact until the penetration depth is strictly positive. Within a normalized simulation time step the penetration time interval $[t_a^i, t_b^i] \in [0,1]$ is the interval in which the feature pair is colliding. With linear trajectories, up to two penetration time intervals can occur within one time step, since the elementary collision test can return up to three real roots. For higher-degree trajectories even more time intervals can occur.

5.2.2 Continuous Penalty Forces

The CPF approach extends the idea of DPF and computes collision impulses by continuously accumulating penalty forces along the penetration depth over the whole time step. Thereby, not only the position at one moment but also the trajectory is included in the computation of the impulse.

The CPF $\mathbf{F}(t)$ is computed analogously to the DPF, but now with time-dependent definitions for the normal $\mathbf{n}(t)$, penetration depth $\delta(t)$

$$\mathbf{F}(t) = k\delta(t)\mathbf{n}(t) = k\mathbf{n}(t)\cdot(\mathbf{p}(t)-\mathbf{q}(t))\mathbf{n}(t). \tag{5.7}$$

The resulting impulse is computed with regard to the penetration intervals

$$\mathbf{I} = \int_{t_i}^{t_i+\Delta t} \mathbf{F}(t)dt = \sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} \mathbf{F}(t)dt = k\sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} \mathbf{n}(t)\cdot(\mathbf{p}(t)-\mathbf{q}(t))\mathbf{n}(t)dt. \tag{5.8}$$

An illustration of the impulse computation is provided in figure 5.1. The linear trajectory can be approximated exactly by the CPF, see figure 5.1 a) and c). Whereas, the DPF overestimate the penetration depth as long as the penetration depth is increasing, see figure 5.1 b). The DPF underestimated the penetration depth as long as it is decreasing, see figure 5.1 d).

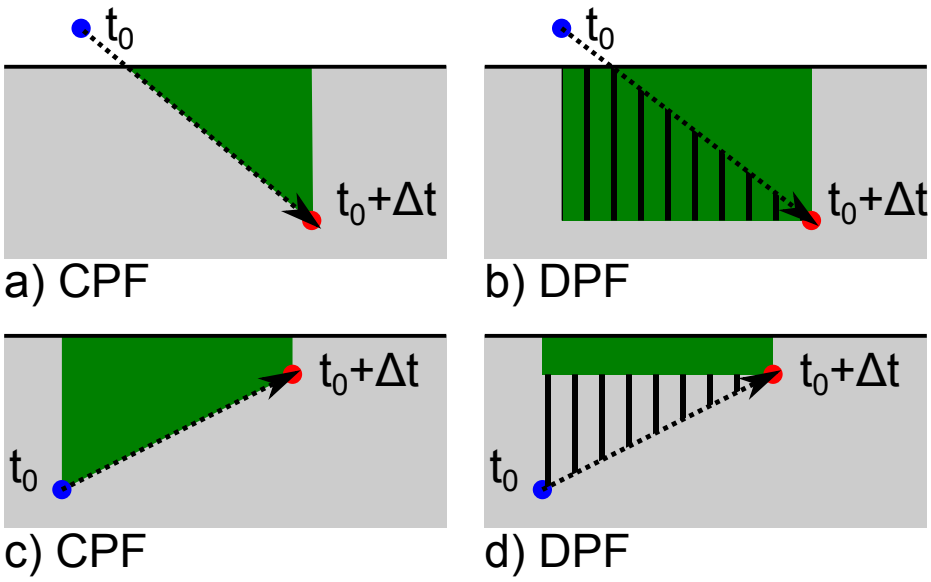


Figure 5.1: 2D Illustration of the resolution force computation for CPF and DPF in the VF case. The blue dot marks the position of the vertex of object A at the beginning of the time step and the red dot the predicted end position. The dotted arrow indicates the linearly approximated trajectory. The grey area is the inside of object B. The green area corresponds to the integral over the force and thereby the resolution impulse. The black striped areas indicate the estimation errors of the DPF forces. It is noticeable that the DPF overestimates the penetration depth as long as the penetration depth is increasing, see subfigure b), and underestimates the penetration depth as long as it is decreasing, see subfigure d).

5.2.3 Vertex-Face Contact Forces

In the VF case, the contact point $\mathbf{p}(t)$ is the position of the colliding vertex V and the contact point $\mathbf{q}(t) = \omega_a\mathbf{a}(t)+\omega_b\mathbf{b}(t)+\omega_c\mathbf{c}(t)$ is the representation of the collision point on the triangle T in barycentric coordinates with the vertex positions $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ and their corresponding weights

ω_a, ω_b and ω_c. The contact normal $\mathbf{n}_T(t) = (\mathbf{b}(t) - \mathbf{a}(t)) \times (\mathbf{c}(t) - \mathbf{a}(t))$ is the normal of the triangle. With equation 5.8, the impulse on the vertex p is

$$\mathbf{I}_p = k \sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} \mathbf{n_T}(t) \cdot (\mathbf{p}(t) - \omega_a \mathbf{a}(t) - \omega_b \mathbf{b}(t) - \omega_c \mathbf{c}(t)) \mathbf{n_T}(t) dt. \tag{5.9}$$

The impulse on the triangle is distributed among the three vertices according to the barycentric weights

$$\mathbf{I}_a = -\omega_a \mathbf{I}_p, \mathbf{I}_b = -\omega_b \mathbf{I}_p, \mathbf{I}_c = -\omega_c \mathbf{I}_p. \tag{5.10}$$

For linear interpolated motions, the velocity is constant and can be computed with the vertices' start and end position $(\mathbf{p}_0, \mathbf{p}_1)$, $(\mathbf{a}_0, \mathbf{a}_1)$, $(\mathbf{b}_0, \mathbf{b}_1)$ and $(\mathbf{c}_0, \mathbf{c}_1)$ as $\mathbf{v}_p = \mathbf{p}_1 - \mathbf{p}_0$ and similarly for $\mathbf{v}_a$, $\mathbf{v}_b$ and $\mathbf{v}_c$. With the second order Bernstein polynomials

$$B_i^2(t) = \frac{2!}{i!(2-i)!} t^i (1-t)^{2-i} \tag{5.11}$$

and the constant basis coefficients $\mathbf{n}_0 = (\mathbf{b}_0 - \mathbf{a}_0) \times (\mathbf{c}_0 - \mathbf{a}_0)$, $\mathbf{n}_1 = \frac{\mathbf{n}_0 + \mathbf{n}_2 - (\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_c - \mathbf{v}_a)}{2}$ and $\mathbf{n}_2 = (\mathbf{b}_1 - \mathbf{a}_1) \times (\mathbf{c}_1 - \mathbf{a}_1)$ and the normalization factor $L(t)$, the time varying contact normal is

$$\mathbf{n_T}(t) = \frac{\mathbf{n_0} B_0^2(t) + \mathbf{n_1} B_1^2(t) + \mathbf{n_2} B_2^2(t)}{L(t)}. \tag{5.12}$$

The normalization factor $L(t)$ is composed as

$$L(t) = \sqrt{(L_0 L_1 ... L_4)(B_0^4(t) B_1^4(t) ... B_4^4(t))}, \tag{5.13}$$

with the factors L_i, which are given in the supplementary material of [18].
To compute the impulse exactly, it would be necessary to integrate the rational function n_T which is rather complicated because of the denominator $L(t)$ [18]. Tang et al. suggest to approximate $L(t)$ by a constant L_k

$$L(t) \approx L_k = \sqrt{\frac{L_0 + L_1 + L_2 + L_3 + L_4}{5}}. \tag{5.14}$$

With this approximation the normal is a quadratic term and the computation of the integral is simplified, resulting in speed-ups up to an order of magnitude without noticeable differences in the simulation results [18].

5.2.4 Edge-Edge Contact Forces

In the EE case, the contact points $\mathbf{p}(t) = \omega_a \mathbf{a}(t) + \omega_b \mathbf{b}(t)$ and $\mathbf{q}(t) = \omega_c \mathbf{c}(t) + \omega_d \mathbf{d}(t)$ are the barycentric representation of the collision points on the edges E_1 and E_2 with the vertex positions $\mathbf{a}(t)$, $\mathbf{b}(t)$, $\mathbf{c}(t)$ and $\mathbf{d}(t)$ and their corresponding weights ω_a, ω_b, ω_c and ω_d. The

contact normal $\mathbf{n_E}(t) = (\mathbf{b}(t) - \mathbf{a}(t)) \times (\mathbf{d}(t) - \mathbf{c}(t))$ is the cross product of the two edge vectors. With equation 5.8 the impulse on the edge E_1 is

$$\mathbf{I}_E = k \sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} \mathbf{n_E}(t) \cdot (\omega_a \mathbf{a}(t) + \omega_b \mathbf{b}(t) - \omega_c \mathbf{c}(t) - \omega_d \mathbf{d}(t)) \mathbf{n_E}(t) dt. \tag{5.15}$$

The impulse is distributed among the four vertices according to the barycentric weights

$$\mathbf{I}_a = \omega_a \mathbf{I}_E, \mathbf{I}_b = \omega_b \mathbf{I}_E, \mathbf{I}_c = -\omega_c \mathbf{I}_E, \mathbf{I}_d = -\omega_d \mathbf{I}_E. \tag{5.16}$$

Similar to the VF case, for linear interpolated motion the velocity is constant and the velocity can be computed from the vertices' start and end positions $(\mathbf{a}_0, \mathbf{a}_1)$, $(\mathbf{b}_0, \mathbf{b}_1)$, $(\mathbf{c}_0, \mathbf{c}_1)$ and $(\mathbf{d}_0, \mathbf{d}_1)$ as $\mathbf{v}_a = \mathbf{a}_1 - \mathbf{a}_0$ and analogously for $\mathbf{v}_b$, $\mathbf{v}_c$ and $\mathbf{v}_d$. With the second order Bernstein polynomials

$$B_i^2(t) = \frac{2!}{i!(2-i)!} t^i (1-t)^{2-i} \tag{5.17}$$

and the constant basis coefficients $\mathbf{n}_0' = (\mathbf{b}_0 - \mathbf{a}_0) \times (\mathbf{d}_0 - \mathbf{c}_0)$, $\mathbf{n}_1' = \frac{\mathbf{n}_0' + \mathbf{n}_2' - (\mathbf{v}_b - \mathbf{v}_a) \times (\mathbf{v}_d - \mathbf{v}_c)}{2}$ and $\mathbf{n}_2' = (\mathbf{b}_1 - \mathbf{a}_1) \times (\mathbf{d}_1 - \mathbf{c}_1)$ and the normalization factor $L(t)$, the time varying contact normal can be expressed as

$$\mathbf{n_T}(t) = \frac{\mathbf{n}_0' B_0^2(t) + \mathbf{n}_1' B_1^2(t) + \mathbf{n}_2' B_2^2(t)}{L(t)} \tag{5.18}$$

The normalization factor $L(t)$ can be computed as in the VF case.
We now give an example how to apply CPF for collisions for a static ground plane. Ground collisions are a common scenario in many environments and due to the static plane the formula can be simplified.
For collisions with the ground plane in an environment with gravity in the negative y-direction, the contact normal $\mathbf{n_P} = (0, 1, 0)$ is constant. With the position of the colliding vertex $\mathbf{p}(t)$ the impulse is

$$\mathbf{I}_p = k \sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} \mathbf{n_P} \cdot \mathbf{p}(t) \mathbf{n_P} dt = k \sum_{i=0}^{n} \int_{t_a^i}^{t_b^i} (0, p_y(t), 0) dt, \tag{5.19}$$

with $p_y(t)$ the $\mathbf{p}(t)$ component in y-direction.
Again we assume linear interpolated motion for the particle $\mathbf{p}(t) = \mathbf{p_0} + t\mathbf{v_p}$. Solving the integral gives a quadratic expression for the vertex impulse

$$\mathbf{I}_p = (0, (t_a - t_b) k p_y(t_0) - \frac{t_b^2 - t_a^2}{2} k v_y, 0). \tag{5.20}$$

No outer sum is required, since maximum one penetration interval can occur.

5.3 Friction

We model friction with a penalty-based model by Yamane and Nakamura[23], providing a straightforward integration in the penalty based collision resolution, only small computational overhead and an unified handling for rigid and deformable bodies. The penalty-based formulation differs between dynamic and static friction, similar to Coulomb's friction law. Coulomb's friction law [23] defines the friction force $\mathbf{f}_f$ as

$$\mathbf{f}_f = \begin{cases} \mathbf{f}_s & \text{if } |v_{slip}| = 0 \text{ and } |\mathbf{f}_s| \leq \mu_S |\mathbf{f}_n| \\ \mu_D \mathbf{f}_n & \text{else} \end{cases} \tag{5.21}$$

with the static and dynamic friction coefficients μ_s and μ_d and the normal force $\mathbf{f}_n$. For the model by Yamane and Nakamura, we first compute the static friction force $\mathbf{f}_s$ as

$$\mathbf{f}_s = (k_{FP}(\mathbf{p}_{ref} - \mathbf{p}) - k_{FD}\mathbf{v}_{slip}) \tag{5.22}$$

with the penalty coefficients k_{FP} and k_{FD}, the current position $\mathbf{p}$ and the reference position $\mathbf{p}_{ref}$, where $\mathbf{p}$ is to be fixed. $\mathbf{p}_{ref}$ is initialized when the features come into contact with the contact point and is updated to the current position $\mathbf{p}$ each time dynamic friction is applied. Similar to Coulombs law, we apply static friction, if $|\mathbf{f}_s| \leq \mu_s |\mathbf{f}_n|$. Otherwise we apply dynamic friction as

$$\mathbf{f}_d = -\mu_d |\mathbf{f}_n| \frac{\omega(|\mathbf{v}_{slip}|)}{|\mathbf{v}_{slip}|} \mathbf{v}_{slip}, \tag{5.23}$$

with the continuous weighting function $\omega(x)$. $\omega(x)$ is defined by

$$\omega(x) = 1 - e^{-k_\omega x} \tag{5.24}$$

with the positive constant k_ω. $\omega(x)$ satisfies $\omega(0) = 0$ and $\lim_{x \to +\infty} \omega(x) = 1$. Furthermore, $\omega(x)$ solves the potential singularity arising from the division by $|\mathbf{v}_{slip}|$ for small $|\mathbf{v}_{slip}|$, since

$$\lim_{x \to 0} \frac{\omega(x)}{x} = k_\omega. \tag{5.25}$$

Therby, for $|v_{slip}| \ll 1$ the dynamic friction can be approximated as

$$\mathbf{f}_d = -\mu_d |\mathbf{f}_n| k_\omega \mathbf{v}_{slip}. \tag{5.26}$$

The static friction model is similar to applying a damped spring at the contact point. If the spring force is too large, dynamic friction is applied.

6 Robust handling of enduring contacts

This chapter outlines the artifacts we observed emerging from the CPF algorithm [18] and our approaches to treat them. We will first give a brief overview on the artifacts and then describe them in detail. Subsequently, we will describe our approaches to handle the artifacts.

6.1 CPF Artifacts

The CPF collision handling as described by Tang et al. [18] shows artifacts, in particular for enduring contacts. Enduring contacts are contacts, which are not resolved in the time step they occur. We observed three major artifacts, separated features still assumed colliding, additional artificial collisions through discretization and inconsistent feature pair mapping. The artifacts emerge if collisions from previous time steps are not completely resolved and additionally a colliding features performs a motion tangential to the surface it is intersecting. In particular, such a behavior can be observed for sliding and deep collisions.

6.1.1 Separated Features Still Assumed Colliding

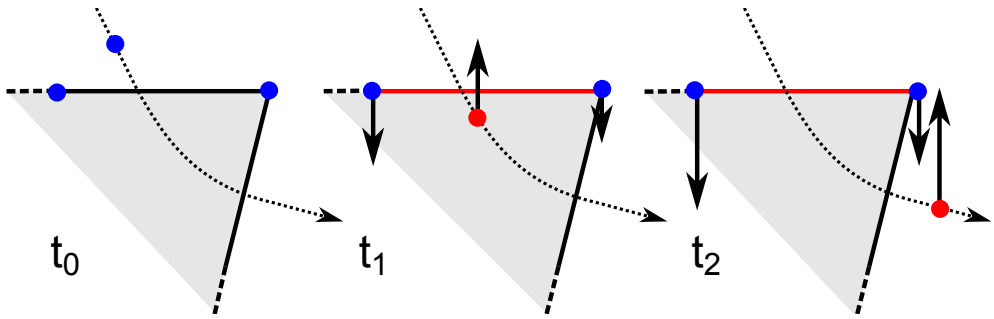


Figure 6.1: 2D-Illustration of a VF collision, which is considered colliding after it is already resolved. Dots indicate the vertices, the red ones are colliding. The straight lines indicate the the surface triangles, the red ones are colliding. The black arrows indicating the applied impulses. t_0 no collision; t_1 the single vertex is colliding with the red triangle; t_2 the single vertex is still colliding with the red triangle, although it left the body through the black face.

A feature pair is considered colliding as long as the penetration depth is strictly negative [18]. This can lead to feature pairs, which are considered colliding even if the bodies are already separated. The collision resolution is missed if a feature is colliding with another feature and leaves the body by passing through a third feature, without reducing the original penetration depth to zero. A good example for such a scenario is shown in figure 6.1. A vertex enters a cube through the top face and leaves the cube through a side face, however the separated features are still assumed to be colliding.

For EE collisions a similar behavior can be observed. For EE collisions it is additionally possible that an edge leaves the body without colliding with another edge.

6.1.2 Additional Artificial Collisions Through Discretization

Consider a cube colliding with an analytical static plane. Correctly there are only impulses applied to the object in the direction normal to the plane and the cube's tangential velocity does not change. If we swap the analytical plane for a discretized triangle surface mesh, again we would expect that only impulses are applied which show in the direction normal to the plane. However, collisions arise with normals showing in a direction tangential to the plane. They stop the tangential movement directly and and hook the body in the surface, as shown in figure 6.2. These collisions arise between the edges and vertices of the plane with the edges and faces of the cube. However, the edges and vertices of the plane are only a product of the discretization and should not induce tangential forces.
The artifact can be described more generic as follows. When two bodies collide with non-zero tangential velocity or with deep intersections, collisions arise with contact normals showing in a direction tangential to the surfaces. This makes the objects collide sidewards, stops the tangential movement and hooks the objects between surface elements.
In our simulations the consequences of this artifact were severe to such an extent that a simulation of collisions with even only small tangential movement was not possible.

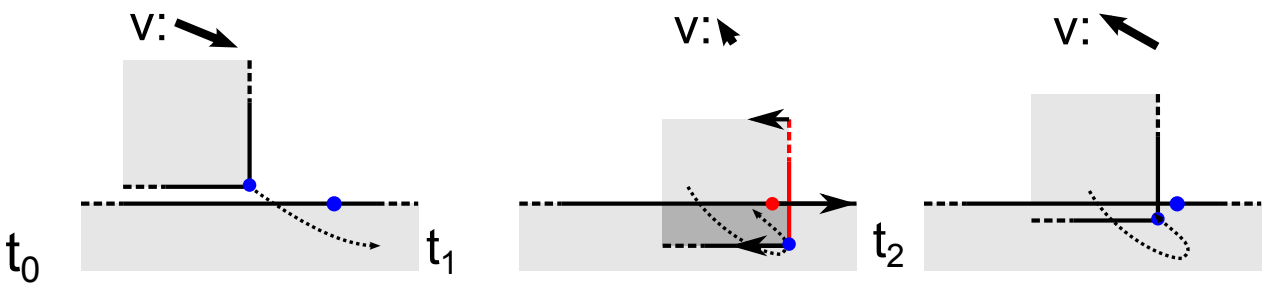


Figure 6.2: 2D-Illustration of a hooked VF collision with tangential forces induced by the CPF algorithm. For the sake of clarity, only the artifact collision is highlighted. t_0 no collision; t_1 the red face collides with the surface vertex of the lower body resulting in forces almost tangential to the surface of the lower body and reverting the tangential movement; t_2 the artifact collision is resolved, however the colliding object completely reverted its velocity, similar to a collision with a artificial wall.

6.1.3 Inconsistent Feature Pair Mapping

When two features collide, the feature pair and the barycentric weights are stored for further collision treatment and used until the collision is resolved [18].
For the VF case, this can lead to implausible feature pairs when a vertex moves tangential to the surface of the body it is intersecting. The implausible feature pairs arise if the vertex continues its tangential movement and moves below a neighboring triangle, as shown in Fig 6.3. As a result there are large distances between the vertex and the colliding triangle, which results in

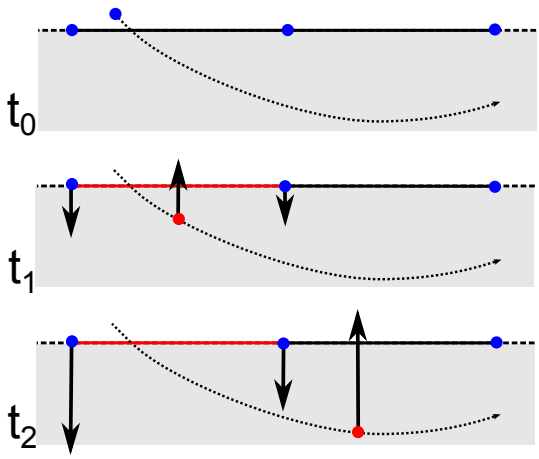


Figure 6.3: 2D-Illustration of a VF collision causing impulses at implausible positions. At time t_0 there is no collision. At time t_1 the single vertex is colliding with the red triangle. At time t_2 the single vertex is still colliding with the red triangle, although the vertex shifted under the triangle right to the red one.

impulses at distant vertex positions. This leads to implausible behavior as an object is deformed at positions distant from the collision.
In the case of EE collisions, a similar behavior can be observed. When an edge moves tangential to the surface of the body it is intersecting and the edge slides below another edge, impulses are applied at distant positions.
Furthermore, the implausible feature pairs can lead to wrong resolutions of collisions, as shown in figure 6.4. With a large distance between the two colliding features, only a minor rotation of the colliding face or an colliding edge is necessary to reduce the penetration depth to zero. Thereby the resolution condition (given in section 5.2.1) is fulfilled, whereas the feature might still be intersecting the body. This unwanted resolution appears for example when a rigid cube

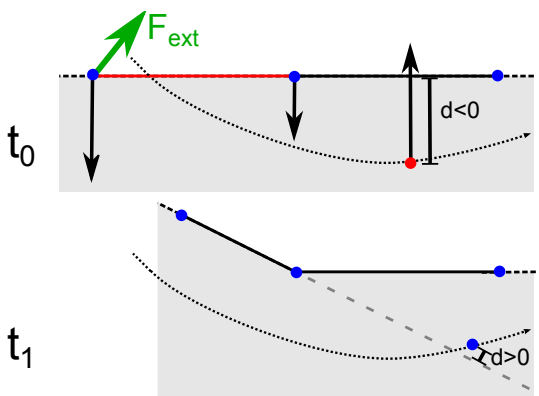


Figure 6.4: 2D-Illustration of an incorrect VF collision resolution artifact. At time t_0 the single vertex is colliding with the red triangle and a external force is applied to the red triangle. At time t_1 the red triangle rotated, the distance of the vertex to the triangle plane is now positive and thereby the collision resolution condition is incorrectly satisfied.

is dropped on a deformable bar and afterwards the rigid cubes slides on the bar. Due to impact of the rigid body, the surface of the deformable body oscillates and the surface triangles perform small rotations. These rotations are sufficient to fulfill the resolution condition. Thereby the

collision is "resolved", whereas the vertex should still be colliding since it is intersecting the body.

6.2 CPF Extensions

To maintain plausible feature pairs and correct resolution impulses, extensions of the original CPF algorithm are necessary. In total we introduce three additional steps solving and reducing the issues described in the previous section.

6.2.1 Advanced Resolution Condition

To accomplish a correct resolution detection and to avoid the separated features still assumed colliding, we extend the resolution condition by additionally taking collisions of the features' vertices into account.
For the VF case, we consider a collision resolved, if either the penetration depth is positive or the colliding vertex collides with another face of the intersecting body. This condition is sufficient, since no collision is missed due to the CCD. Therefore, when the collision resolves the vertex passes the surface of the body it is intersecting and thereby a triangle vertex contact is detected.
For the EE case we consider a collision resolved, if either the penetration depth is positive or when both vertices of an edge are resolved and the edge is not intersecting a face of the previously intersected body. This condition is sufficient, since when the edge is colliding, but none of the edge's vertices is colliding, the edge pierces the surface and thereby the edge intersects a triangle. Otherwise, the edge is not colliding.

6.2.2 Contact Categorization

As described in section 6.1.2, some collisions generate artifact forces when the tangential velocity is non-zero. To avoid the artifact forces we need to distinguish these collisions from the regular collisions. Therefore, we introduce conditions in order to categorize the collisions either as "regular", which can be handled with the regular algorithm or as "hooked", which generate artifact forces and require special handling. The categorization is limited to non-deep contacts, though we will discuss possibilities to extend the categorization to deep collisions at the end of the section.
A **regular** collision is considered as the common case, with the two bodies being pushed away from each other. This applies for collisions with the surface normals of the colliding features showing in opposite directions and the resolution impulses showing inside the according bodies. Regular collisions are handled with the original algorithm.
A **hooked** collision appears for sliding contacts or deep intersections. The surface normals are closer to an orthogonal state than to an opposing state. This results in impulses tangential to a surface. These collisions are implausible since the tangential motion should not change without friction. The body hooks in the surface, as it can be seen in figure 6.2. Therefore, we redefine the feature pairs of hooked collisions in order to reduce the tangential impulses and to gain regular collisions.
In the following we will discuss the formulation with an emphasis on the VF case. Afterwards, we will give the definition for the EE case, which is analogous.

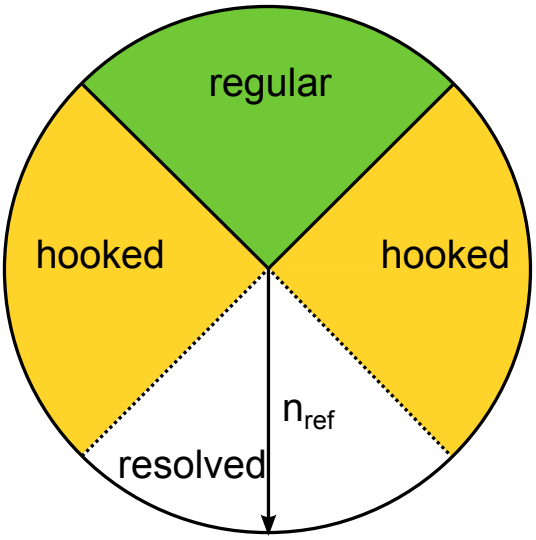


Figure 6.5: The classification circle is an illustration of the condition for a hooked collision. Depending on the surface normal n_v, as shown as example by the grey arrow, and the fixed contact normal $\mathbf{n}_c$it is possible to determine whether a collision is hooked or regular.

The previous description yields the basic formulation of the condition for a hooked collision: A collision is hooked, if the proportion of the contact normal $\mathbf{n_c}$ in a direction tangential to the surface of a feature is larger than the proportion in the direction normal to the surface. Otherwise, the collision is regular. The contact normal is equal to the direction of the applied collision impulse. To express the surface direction of a vertex, we apply the averaged vertex normal $\mathbf{n_v}$ as the average of the adjacent face normals and handle it as a surface normal. The condition for a hooked collision can now be expressed with the dot product as

$$-\mathbf{n_v} \cdot \mathbf{n_c} < \cos(45°) = \frac{\sqrt{2}}{2}, \tag{6.1}$$

with a negative sign before the dot product since the two normals are opposing. This relation is illustrated in the classification circle, see figure 6.5. It is not necessary to additionally check the triangle normal against the contact normal since the contact normal is equal to the triangle normal. Thereby, they are per definition always showing in the same direction.
The basic formulation is insufficient for vertices in acute-angled corners, as shown in the following. Vertices in acute-angled corners of a tetrahedron can collide in a larger area of contact angles than vertices in obtuse-angled corners, as shown in figure 6.6. The figure compares collisions of vertices in acute-angled and an obtuse-angled corners. The subfigures a) and b) have the same relation between the contact normals and the averaged vertex normal. Therefore, the basic formulation yields that both collisions are hooked. However, for subfigure a) the contact normal is almost parallel to the two face normals, therefore we would expect a regular collision. Furthermore, the angle between the averaged vertex normal the normal of the adjacent faces is large, it is close to 90°. Thereby, the averaged vertex normal does not provide a sufficient expression of the surface normal for acute-angled corners.
Therefore, we need to introduce a construct taking the angles between adjacent face normals into account. Accordingly, we introduce a cone alike span $\mathbf{N}$ (see figure 6.7) spanned by the surface normals of the directly adjacent faces of the vertex or edge. It will be called "normal span"

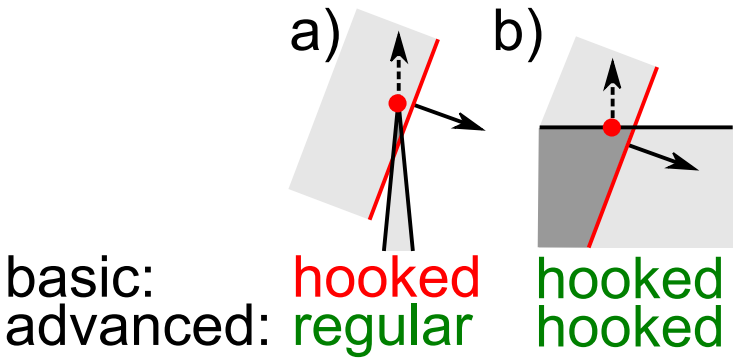


Figure 6.6: Comparison of collisions of vertices in acute-angled and an obtuse-angled corners. The subfigures a) and b) have the same relation between the contact normals and the averaged vertex normal. Therefore, the basic formulation yields that both collisions are hooked. However, for a) the contact normal is almost parallel to the two face normals, therefore we would expect an regular collision.

in the following. The normal span is large for acute-angled corners and small for obtuse-angled

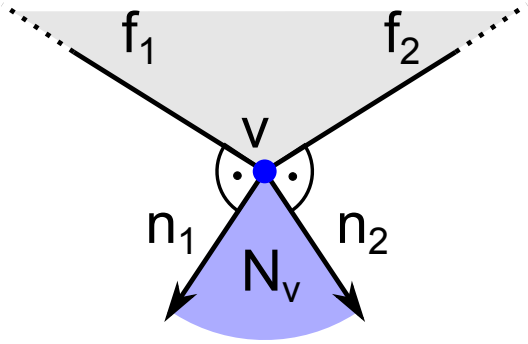


Figure 6.7: 2D-Illustration of the vertex normal span $\mathbf{N}_v$ of the vertex v. The normals of the faces $\mathbf{f}_1$ and $\mathbf{f}_2$ are indicated $\mathbf{n}_1$ and $\mathbf{n}_2$. The vertex normal span is indicated in light blue.

corners. Thereby, the normal span reflects the collision behavior for vertices in such corners and a proper contact categorization is possible (see figure 6.8). The illustration compares the collision category evaluation for the collisions with averaged normals and normal spans. It emphasizes that with normal spans an accurate collision categorization can be provided, regardless of the angle in the vertex corner.

To implement the normal span in the simulation, we need to discretized it. Therefore, we approximate the normal span by the set of triangle normals adjacent to the vertex or edge. For edges only the two directly adjacent faces are considered. For the approximation of larger normal cones we additionally add the averaged vertex normal to the set.

For the contact categorization the normal cone is an often used element. To avoid a large computation overhead due to searching of neighboring elements in order to compute their normal, it is necessary to apply a data structure providing fast neighborhood information. We use a data structure by Serna et al. [17] providing the necessary fast neighborhood information.

The previously introduced category definitions are implemented as follows. A VF collision is hooked, if the proportion of the contact normal showing in the direction tangential to the vertex

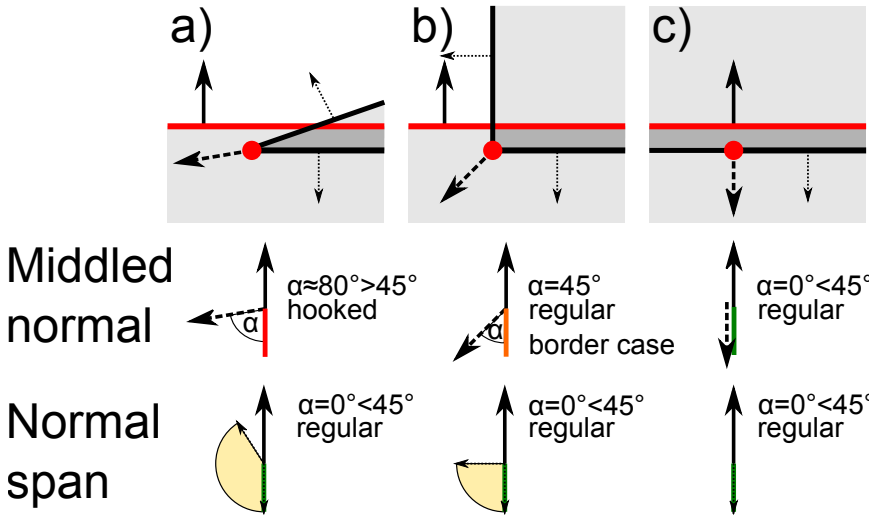


Figure 6.8: Comparison of the contact categorization with normals and normal spans for VF collisions with the vertex in an acute-angled, right-angled and obtuse-angled corner. a) shows a acute-angled corner of a tetrahedron colliding with a flat surface. The normal span indicates correctly a regular collisions, in contrast to the normals indicating a hooked collision. b) shows a right-angled corner of a cube colliding with a flat surface. The normal span indicates correctly a regular collisions, for the normals it is exactly the border between hooked and valid. c) shows a vertex of a flat surface colliding with a flat surface. Both, the normal span and the normals indicate the collision correctly as regular.

surface is larger than proportion showing in the normal direction, expressed similar to the basic formulation as

$$\min(\{\mathbf{n_{Vi}} \cdot \mathbf{n_{cv}}\}) > -\frac{\sqrt{2}}{2} \mid \mathbf{n_{Vi}} \in \mathbf{N_v}, \tag{6.2}$$

with the normal span of the vertex $\mathbf{N_v}$ and the contact normal $\mathbf{n_{cv}}$. This formulation is similar to measuring the minimum angle between the normal span and the negative contact normal. If this angle is larger than 45° the impulse proportion tangential to the face is larger than normal proportion and the collision is hooked. Otherwise the collision is regular.
A EE collision is hooked, if the proportion of the contact normal showing in the direction tangential to the surface is larger than proportion showing in the normal direction for both edges, expressed similar to the basic formulation as

$$\min(\{\mathbf{n_{Eai}} \cdot \mathbf{n_{ce}}\}) > -\frac{\sqrt{2}}{2} \wedge \max(\{\mathbf{n_{Ebi}} \cdot \mathbf{n_{ce}}\}) < \frac{\sqrt{2}}{2} \mid, \mathbf{n_{Eai}} \in \mathbf{N_{Ea}},\ \mathbf{n_{Ebi}} \in \mathbf{N_{Eb}} \tag{6.3}$$

with the normal spans $\mathbf{N}_{Ea}$ and $\mathbf{N}_{Eb}$ of the edges and the contact normal $\mathbf{n_{ce}}$.
If the collision is not hooked, it is regular.
The handling of regular collisions is straightforward. Regular collisions are handled with the CPF algorithm without modification. The handling on hooked contacts is based on the algorithm for the redefinition of feature pairs in the next section 6.2.3. Therefore, the handling of hooked contacts is described in section 6.2.4.

In the beginning of this section we have stated that the categorization is only applicable for non-deep collisions. Now we will investigate the problem arising from deep collisions and how it could be solved.
This collision definition is limited to non-deep intersections, since we can not differentiate between a hooked contact and a deep intersection with only the local collision data, see figure 6.9. On the left side the local collision arrangement is shown and on the right side three possible global arrangements for the local collision arrangement are shown. With only the local collision data it is not possible to differentiate between the three cases. However, it is necessary

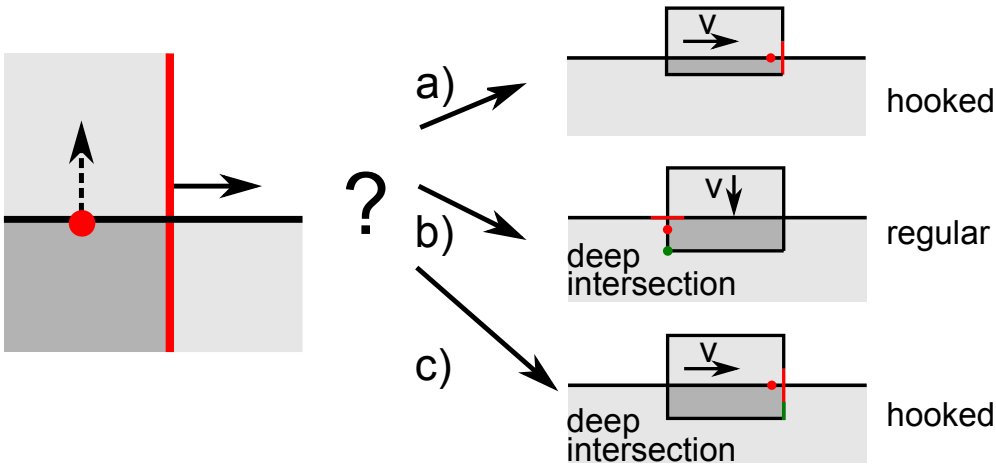


Figure 6.9: Illustration of limitation to of the categorization algorithm to non-deep contacts, since otherwise the collision arrangement is ambiguous, as shown by the possible arrangements a)-c).

to differentiate between the three cases in order to yield a distinct categorization since arrangements a) and c) are hooked collisions whereas b) is a regular collisions. As an interim solution we restricted the algorithm to non-deep collisions, thereby only arrangement a) is possible and the local arrangement is unambiguous. Furthermore, non-deep collisions are sufficient in many cases, since deep collisions often provide a visually unsatisfying result.
Nevertheless, the capability to handle deep contacts is desirable in order to simulate high-resolution meshes and to increase the overall robustness of the algorithm. To handle deep contacts additional information about the collision arrangement is required. This additional information could be provided by a global intersection analysis, similar to Baraff et al. [2], thereby it would be possible to differentiate between the sliding and the regular collisions for deep collisions. Alternatively, checking the neighbors of the new colliding vertex for collisions and taking their contact normals as a reference, would provide the necessary information and would provide less computational overhead.

6.2.3 Redefinition of Feature Pairs

To ensure plausible feature pairs and to avoid an inconsistent feature pair mapping, we trace the collisions and check each simulation step for possibly necessary redefinitions. A redefinition is possibly necessary if the first feature of the feature pair projected onto the second feature does not provide a geometrical overlap. A redefinition is only "possibly" necessary since for concave surfaces it is possible that the projections of a feature do not overlap with any other feature.

Imagine the 2D case of a vertex in the center of a five-pointed star, none of the sides overlaps with the projection of the vertex on the lines defined by the sides.

For VF collisions the assumption holds that as long as a vertex intersects a body, it is colliding with exact one face of the intersecting body. The redefinition for the VF case can be subdivided into two steps. The first step is to check if a redefinition is possibly necessary. If a redefinition is possibly necessary, we apply a second step and search for the correct face corresponding to the colliding vertex. An example of a VF redefinition is presented in figure 6.10 and in figure 6.11.

In contrast to the VF case, no similar assumption holds for the EE case. An edge intersecting with another body can be colliding with one edge, with multiple edges or even with no edge. Hence, there is no coherence between the elimination and new definition of a feature pair. Therefore, the elimination and new definition are treated separately. We check for eliminations at the end of each simulation step. For the new EE definitions we leverage that the new EE definitions go along with VF redefinitions. Thus, we check for new EE definitions whenever we redefine a VF collision.

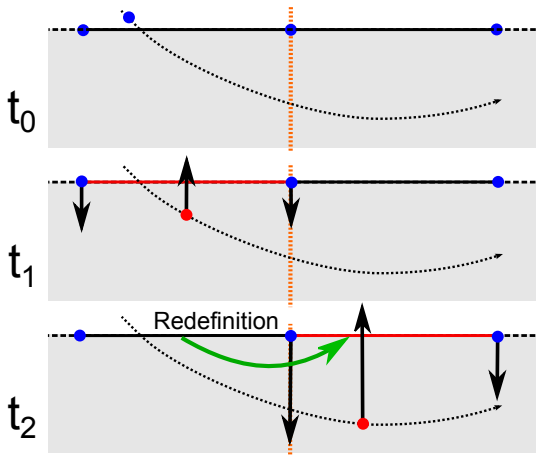


Figure 6.10: 2D-Illustration of a barycentric redefinition. t_0 no collision, t_1 the single vertex is colliding with the left triangle, indicated red, t_2 the single vertex is now still colliding with the right triangle. The colliding face was redefined as indicated by the green arrow. The redefinition border is indicated with the orange-dotted line.

Vertex Face Case

For the VF case, we project the colliding vertex V onto the plane of the colliding face F and compute the barycentric coordinates ω_{fa}, ω_{fb} and ω_{fc}, see algorithm 6.1. The projected vertex is within the triangle if all barycentric weights are positive. Such collision pairs are considered valid and no redefinition is necessary. Otherwise, it is possibly necessary to redefine the pair and we assume that the vertex went on to a face adjacent to the initial face. With the direction information of the previously computed barycentric coordinates, we only need to check the faces adjacent to the vertices with positive weights. Therefore, we project the vertex V on the faces adjacent to vertices with positive weights.

Again, we check if the projected vertices are inside the new triangles by computing the barycentric weights. All faces F_i having only positive barycentric weights are candidates for the redefinition. In order the gain a regular collision, we apply the classification (see section 6.2.2) and keep only the regular collisions as candidates. For the remaining candidates, we compute

(a) before the redefinition

(b) after the redefinition

Figure 6.11: Example of a barycentric redefinition in a simulation. Subfigure a) shows the situation before the redefinition. Subfigure b) shows the same situation one time step later. The left vertex shifted under the next face, the collision is redefined and thereby updated to the next face .

the penetration depths and choose the candidate with the smallest penetration depth. If no candidate was found, we do not redefine the collision.
To handle concave faces, we need to extend the algorithm. For concave meshes it is possible that there is no face giving positive barycentric weights in the direct neighborhood. So we need to modify the barycentric weight condition. We define the barycentric tolerance ε_v as

$$\varepsilon_v = \max\{\begin{cases} -\omega_i & \text{if } \omega_i < 0 \\ \omega_i - 1 & \text{if } \omega_i > 1 \\ 0 & \text{else} \end{cases} \mid \omega_i \in \{\omega_a, \omega_b, \omega_c\}\} \tag{6.4}$$

with the initial barycentric weights ω_{fa}, ω_{fb} and ω_{fc}. ε_v expresses the maximum discrepancy of the barycentric weights to the valid interval $[0, 1]$.
In the algorithm we modify the condition for redefinition candidates. We now do not only add all vertices having all their barycentric coordinates in the interval $[0, 1]$, but also the vertices in the interval $[-\varepsilon_v, 1 + \varepsilon_v]$.

Edge Edge Case

For the EE case, in each simulation step before we compute the collision impulses, we check whether the collision needs to be dismissed. Therefore, we project the edge E_B on the plane defined by the Edge E_A with the contact normal as plane and projection normal. We compute the barycentric coordinates ω_{A0}, ω_{A1}, ω_{B0} and ω_{B1} of the point of intersection between the projected E_A with E_B. If the edges overlap, all barycentric coordinates are positive and the collision will be handled. To avoid the missing of EE collisions, we apply a relaxed condition; if the barycentric weights are in the interval $[-\varepsilon_E, 1 + \varepsilon_E]$ with $\varepsilon_E = 0.3$ the collision is kept, but not handled. For the case that not even the relaxed condition is fulfilled, the collision is dismissed.
Due to the circumstance that there is not always a coherence between the dismissing of an EE collision and the appearance of a new EE collision, checking the dismissed edges' neighbors is not sufficient for an accurate EE redefinition. In consideration of the circumstance that EE redefinitions go along with VF redefinitions we couple the EE redefinitions to the VF redefinitions instead of EE dismissals.

Algorithm 6.1 Vertex Face Feature Pair Redefinition for convex meshes

```
Input: Vertex V, Face F, List Collisions, List ResolvedCollisions, ω_FA, ω_FB, ω_FC, n_F
Result: List Collisions, List ResolvedCollisions
List RedefinitionCandidates
if ω_FA > 0 then
    RedefinitionCandidates.add(neighbouringFaces(V_A))
end if
if ω_FB > 0 then
    RedefinitionCandidates.add(neighbouringFaces(V_B))
end if
if ω_FC > 0 then
    RedefinitionCandidates.add(neighbouringFaces(V_C))
end if
currentBestDepth = −∞
Collision currentCollision
for all Face F_i in RedefinitionCandidates do
    ω_IA, ω_IB, ω_IC ⇐ computedProjectedBarycentrics(V,F_i)
    if ω_IA > 0 ∧ ω_IB > 0 ∧ ω_IC > 0 then
        if regular==collisontype(Collision(V,F_i)) then
            n_i =faceNormal(F_i)
            d_i =penetrationdepth(V,F_i)
            if d_i >currentBestDepth ∧ d_i < 0) then
                currentBestDepth= d_i
                currentCollision=Collision(V,F_i)
            end if
        end if
    end if
end for
if currentBestDepth> −∞ then
    Collisions.add(currentCollision)
    ResolvedCollisions.add(Collision(V,F))
end if
```

We consider a VF redefinition of a vertex V_S which is redefined from the face F_{old} with the face normal n_{ref} to the adjacent face F_{new}. All neighboring edges of V_S are saved in a candidate list L_A. All edges adjacent to at least one of the mutual vertices of F_{old} and F_{new} are saved in a second candidate list L_B.
We test all possible EE feature pairs for new collisions, by testing each edge in L_A against each edge in L_B. If a feature pair is according to section 6.2.2 regular and the projected barycentric coordinates are in the relaxed interval $[-\varepsilon_E, 1+\varepsilon_E]$ we keep the feature pair as a collision.

6.2.4 Redefinition of Hooked Contacts

We consider hooked collisions as aberrant and correct these collisions. A nearby attempt would be to directly modify the direction of the collision impulse. However, the direction is defined by the contact normal, which is an essential element of the CPF algorithm. Therefore, we redefine the hooked collision pairs in order to reduce the tangential impulses and to gain regular collision pairs, as shown in figure 6.12. For the EE case these redefinitions are already covered by the collision redefinition in section 6.2.3. For the VF case we handle the hooked contacts with a redefinition of the feature pair similar to the implausible feature pairs case, see section 6.2.3. In contrast to the implausible feature pairs case, the barycentric weights for the initial face generally are positive. This allows us to use algorithm 6.1 with only a small change in the first step. We now search in all triangles adjacent to the triangle vertices and not only the triangles adjacent to vertices with positive weights. This change is necessary since we do not have a direction information, as in the case of barycentric redefinition provided by violation of [0,1] interval by the barycentric weights.

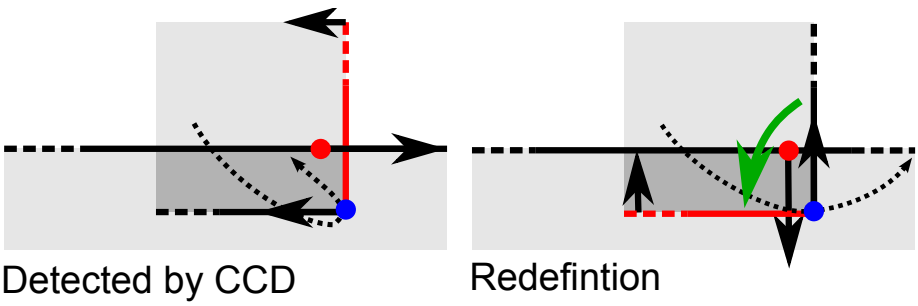


Figure 6.12: 2D-Illustration of the redefinition for a hooked collision. The redefinition is marked with a green arrow. Before the redefinition, the red face collides with the surface vertex of the lower body resulting in forces almost tangential to the surface of the lower body and reverting the tangential movement. After the redefinition, the vertex collides with the redefined bottom sided face, resulting in forces in the normal direction of the lower surface and the tangential movement is preserved.

In sum, the redefinition algorithm for a hooked collision can be reduced to four (simplified) steps (see figure 6.13). In the first step, the triangles adjacent to the colliding triangle are saved as candidates. In the second step, the colliding vertex is projected onto the planes defined by the candidate triangles and it is checked whether the projection is inside the triangles. All candidates with the projection outside the triangle are dismissed. In the third step, the remaining candidates are tested for artifact collisions and all candidates which would yield artifact collisions are dismissed. In the fourth and final step, generally, only one triangle is left. However, it

is possible that multiple triangles are left therefore we choose the candidate with the smallest intersection for the redefinition.

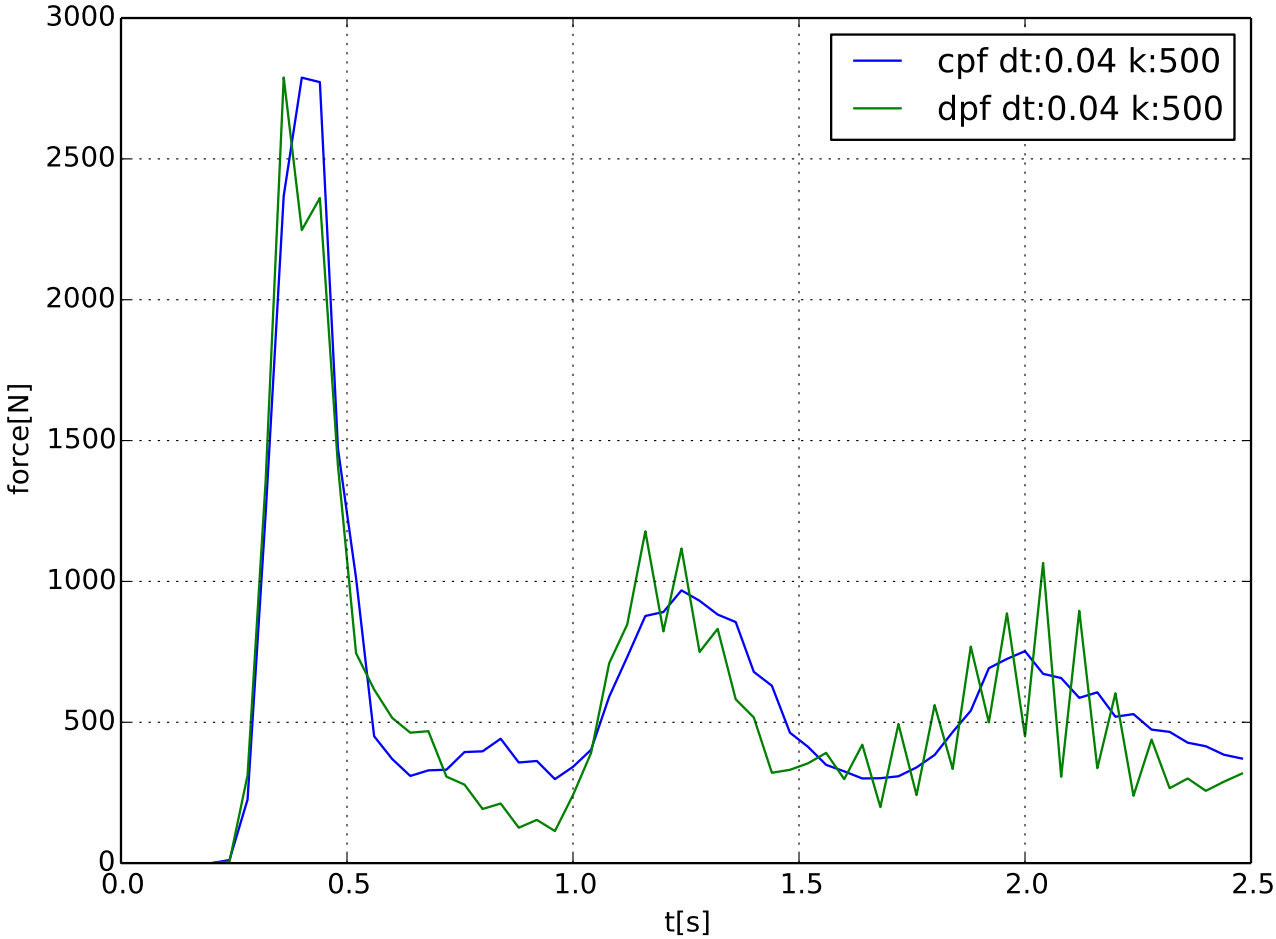


Figure 6.13: 2D-Illustration of the redefinition for a hooked collision. 1) Neighboring triangles are candidates 2) Dismissal of the candidates with the vertex projection outside the triangle 3) Dismissal of the artifact candidates 4) Redefinition to the final candidate

7 Results

In this chapter, we analyze the CPF algorithm and discuss the results in comparison to a DPF algorithm. Furthermore, we analyze and discuss the newly introduced contact redefinition and the friction formulation under use of varying parameters and scenarios.
We will begin with the analysis of a 1D particle on plane contact, providing the possibility to formulate the integration scheme and compute the stable time steps in an analytic fashion. Thereafter, we will apply the preliminary findings to investigate a more complex scenario, the collision between a deformable bar and a rigid cube with a focus on stability, jitter and penetration depth. Afterwards, we will examine the applied friction model regarding static and dynamic friction. Subsequently, we will show the results of the introduced contact redefinition algorithm and discuss the benefits. Finally, we will investigate the overall computational cost of the collision resolution with a special regard to the additional cost induced by the new redefinition algorithm.
In order to provide a matchable result of the comparison of DPF and CPF, in all tests DPF are computed with CCD using the penetration depth at t=1.

7.1 Stability

In general, it is difficult to measure the correct behavior of a multi-body simulation, since it only provides physically plausible instead of physically exact results. Therefore, we need to couple an analytical and visual evaluation.
An essential property for collision handling methods is stability. Otherwise, the energy in the system increases and collision resolutions provide irritating results. Furthermore, jitter and deep intersections disturb the visual impression.

7.1.1 Analysis of 1D Particle-on-Plane Contact

The analysis of a 1D particle-on-plane contact provides the possibility to describe the scenario in an analytical way, which is not feasible for more complex scenarios. The analytical description allows us to apply analytical methods to evaluate the stability. Afterwards, we will validate these findings in the simulation of the particle and the simulation of a more complex scenario.
The continuous motion of a particle heading towards a plane under gravity can be described as

$$\dot{x} = v \tag{7.1}$$

$$\dot{v} = \begin{cases} -g - \frac{kx}{m} & \text{if } x < 0 \\ -g & \text{else} \end{cases} \tag{7.2}$$

with the gravity coefficient g, the mass m, the penalty stiffness k, the position x and the velocity v. The particle only moves up and down, therefore the movement can be expressed in 1D and

all quantities are scalars. As in the simulation the equation needs to be discretized in time with a time step Δt in a form of

$$\begin{pmatrix} x(t+\Delta t) \\ v(t+\Delta t) \end{pmatrix} = \mathbf{A} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}, \tag{7.3}$$

where the matrix $\mathbf{A}$ describes the integration scheme. The integration scheme is stable if all eigenvalues λ satisfy $|\lambda(\mathbf{A})| < 1$.
Discrete penalty forces (see section 5.1) compute the collision impulse as

$$I_{dpf} = -\Delta t k x^*(t+\Delta t) \tag{7.4}$$

with the predicted position $x^*(t+\Delta t) = x(t) + \Delta t\, v^*(t+\Delta t)$ and the predicted velocity $v^*(t+\Delta t)$, which is $v^*(t+\Delta t) = v(t)$ neglecting gravity. The update rule yields

$$v(t+\Delta t) = v(t) + \frac{I_{dpf}}{m} = v(t) - \Delta t \frac{k}{m} x(t) - \Delta t^2 \frac{k}{m} v(t) \tag{7.5}$$

$$x(t+\Delta t) = x(t) + \Delta t\, v(t+\Delta t) \tag{7.6}$$

and in matrix form it is

$$\begin{pmatrix} x(t+\Delta t) \\ v(t+\Delta t) \end{pmatrix} = \begin{pmatrix} 1-\Delta t^2 \frac{k}{m} & \Delta t - \Delta t^3 \frac{k}{m} \\ -\Delta t \frac{k}{m} & 1-\Delta t^2 \frac{k}{m} \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}. \tag{7.7}$$

The eigenvalues analysis gives stable time-steps for $\Delta t < \sqrt{\frac{4}{3}\frac{m}{k}}$.
Continuous penalty forces (see section 5.2) compute the collision impulse as

$$I_{cpf} = -\Delta t k x(t) - \frac{\Delta t^2}{2} k v^*(t+\Delta t) \tag{7.8}$$

with the predicted velocity $v^*(t+\Delta t) = v(t)$. The update rule yields

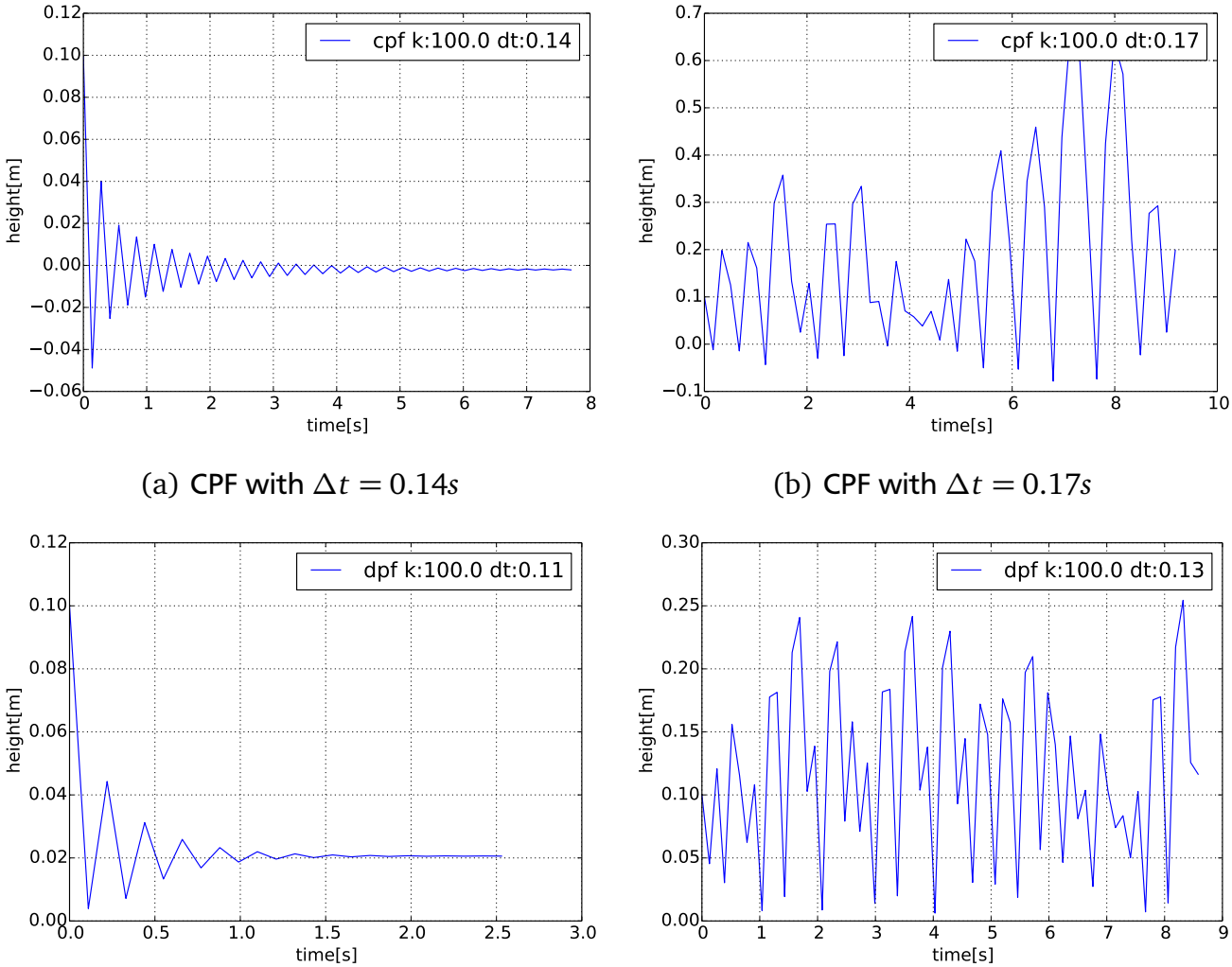
$$v(t+\Delta t) = v(t) + \frac{I_{cpf}}{m} = v(t) - \Delta t \frac{k}{2m} x(t) - \Delta t^2 \frac{k}{m} v(t) \tag{7.9}$$

$$x(t+\Delta t) = x(t) + \Delta t\, v(t+\Delta t) \tag{7.10}$$

and matrix form it is

$$\begin{pmatrix} x(t+\Delta t) \\ v(t+\Delta t) \end{pmatrix} = \begin{pmatrix} 1-\Delta t^2 \frac{k}{m} & \Delta t - \Delta t^3 \frac{k}{2m} \\ -\Delta t \frac{k}{m} & 1-\Delta t^2 \frac{k}{2m} \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \tag{7.11}$$

The eigenvalues analysis yields stable time-steps for $\Delta t < \sqrt{2\frac{m}{k}}$.
Comparing the stable time steps shows that the maximum time step size for CPF is $\frac{\sqrt{6}}{2} \approx 1.22$ times larger than for DPF.
To verify the previous theoretical consideration, we analyze a particle with a mass $m = 1.0kg$ and a starting height $h = 0.1m$ dropped on the plane with a penalty coefficient $k = 100.0\frac{N}{m}$. According to the previous formula the critical step size for CPF is $\Delta t_{cpf} = \sqrt{0.02}s \approx 0.142s$ and for DPF $\Delta t_{dpf} = \sqrt{\frac{4}{300}}s \approx 0.115s$. As shown in figure 7.1, CPF are stable for $\Delta t = 0.14 <$



(a) CPF with $\Delta t = 0.14s$

(b) CPF with $\Delta t = 0.17s$

(c) DPF with $\Delta t = 0.11s$

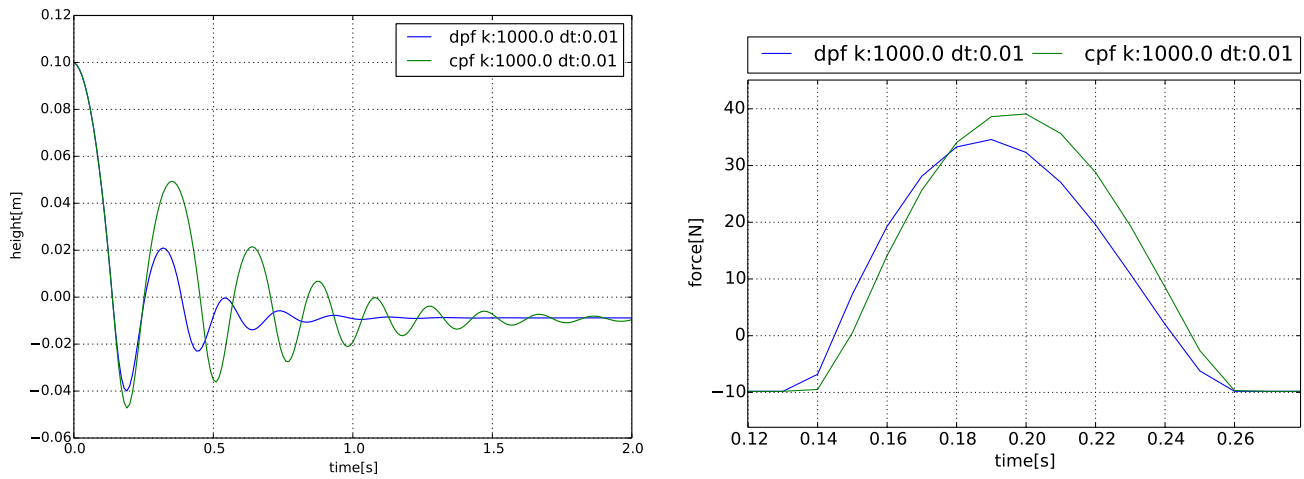
(d) DPF with $\Delta t = 0.13s$

Figure 7.1: The position over time of a particle m=1.0kg dropped from h=0.1m on a plane with continuous (a, b) and discrete (c, d) penalty forces, showing stable (a, c) and unstable (b, d) conditions. It is noticeable that CPF are stable for $\Delta t = 0.14$ whereas the discrete penalty forces are already unstable for $\Delta t = 0.13$.

Δt_{cpf} damping the initial deflection and unstable for $\Delta t = 0.17 > \Delta t_{cpf}$ amplifying the initial deflection. Similarly, the DPF are stable for $\Delta t = 0.11 < \Delta t_{dpf}$ and unstable for $\Delta t = 0.13 > \Delta t_{dpf}$.

As predicted by the stability analysis, the CPF are, for the particle-plane case, stable for larger time steps than DPF. We show that CPF are stable for $\Delta t = 0.14$ whereas the discrete penalty forces are already unstable for $\Delta t = 0.13$.

Stability is an essential property for interactive environments. With the relation for stable time steps $\Delta t < \sqrt{2\frac{m}{k}}$, stability can be ensured by choosing lower penalty coefficients $k \ll 2\frac{m}{\Delta t^2}$. Since this is a common case, we again analyze the movement of a particle $m = 1.0kg$ and a starting height $h = 0.1m$ with a time step $\Delta t = 0.01s$ and penalty coefficient $k = 1000.0\frac{N}{m} \ll 2\frac{1.0}{0.01^2}\frac{N}{m} = 20000\frac{N}{m}$. The result is shown in figure 7.2. Figure 7.2 a) shows the movement of the particle with DPF and CPF. Both methods show likewise results, damping the initial deflection,



(a) Movement of the particle

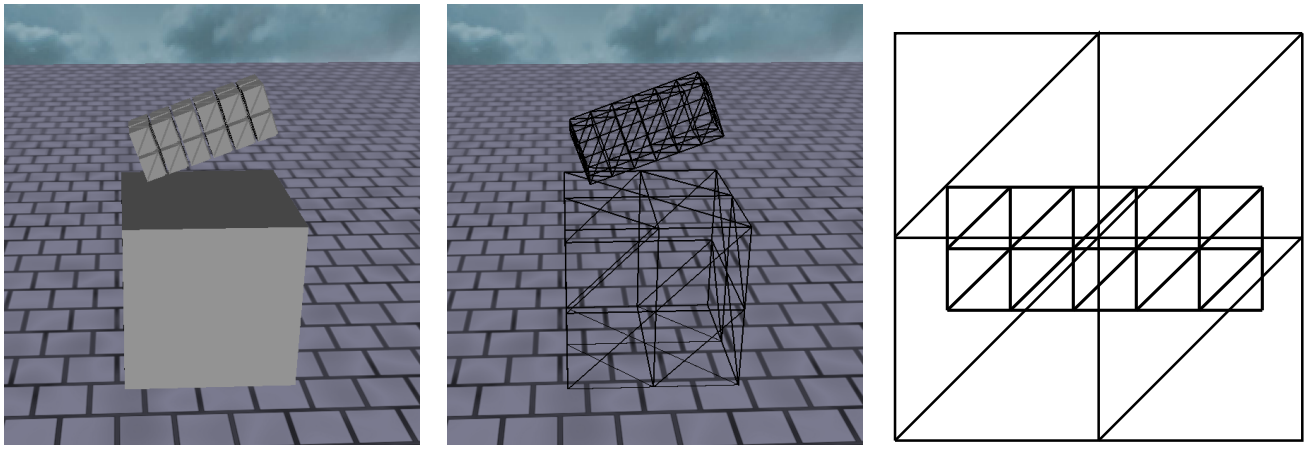
(b) Forces during the first collision

Figure 7.2: The position over time of a particle dropped on a plane with continuous and discrete penalty forces with the same time step $\Delta t = 0.01$ and penalty stiffness $k = 1000$

but the DPF damping is considerably larger than the CPF damping. As a result the CPF particle keeps bouncing for a longer time. The damping is induced by the reason that both methods overestimate the penetration depth as long as the penetration depth is increasing, resulting in an increased deceleration. Similarly, both methods underestimate the penetration depth when the penetration depth is decreasing, resulting in a reduced acceleration of the particle, as described in section 5.2.2.
CPF take the particle trajectory into account, therefore the estimation error is smaller and subsequently the damping is reduced.
The analytical evaluation of the 1D-particle-plane contact has shown that for CPF the maximum stable timestep is $\frac{\sqrt{6}}{2} \approx 1.22$ times larger than the maximum stable timestep for DPF. The result could be validated in the simulation. Furthermore, the simulation has shown less damping for the CPF. Overall, we have seen two benefits of CPF in comparison to DPF, increased stability and reduced damping. The according computational cost is analyzed in section 7.4.

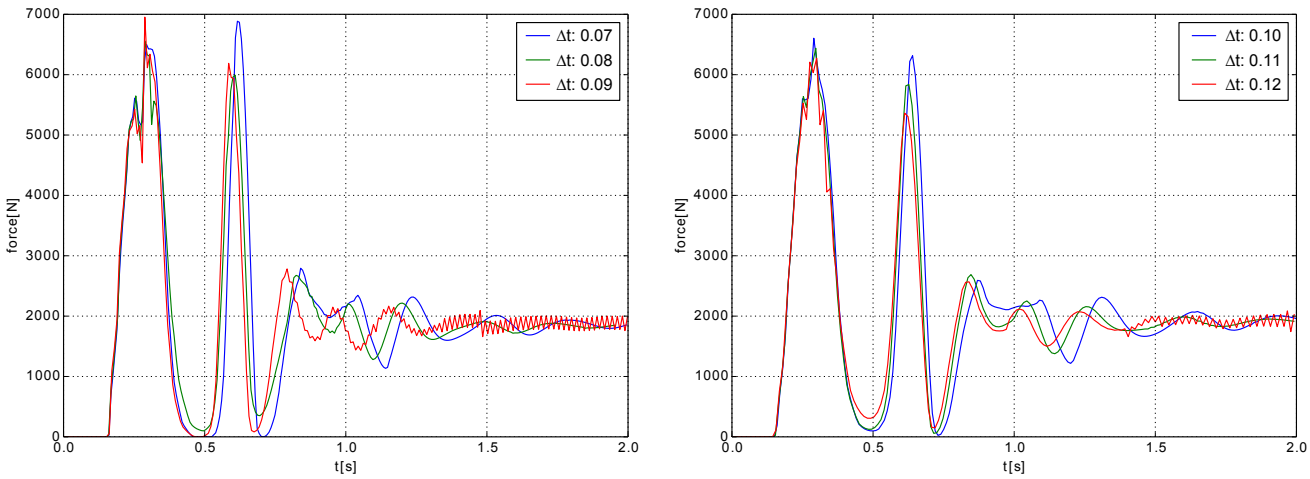
7.1.2 Deformable Bar vs Rigid Cube

The particle-on-plane scenario provides an analytical description of the algorithms, what is not feasible for more complex scenes. However, due to the static plane the overall CPF computation reduces from a degree 6 polynomial to a quadratic polynomial. To investigate the validity of the previous findings for more complex scenarios, we analyze a scenario with a rotated deformable bar dropped on a rigid cube which again is dropped on the ground, as shown in figure 7.3. Furthermore, the scenario is set up in a fashion to avoid the artifacts presented in section 7.1.
We set the penalty coefficient to $k = 5000$ and vary the time step sizes. DPF resolve the collisions without notable jitter for time steps up to $\Delta t_d = 0.08$, whereas CPF are smooth for time steps up to $\Delta t_c = 0.11$. This matches roughly the ratio of 1.22 predicted in the particle-on-plane analysis. The jitter observed in simulation is also notable in the plot of the collision forces between the cube and the bar as shown in figure 7.4. Figure 7.4 a) shows the DPF repulsion forces for time steps sizes $\Delta t = 0.007s$, $\Delta t = 0.008s = \Delta t_d$, $\Delta t = 0.009s$ and figure 7.4 b) shows the CPF



(a) 3D flat shaded (b) 3D wireframe (c) 2D from top

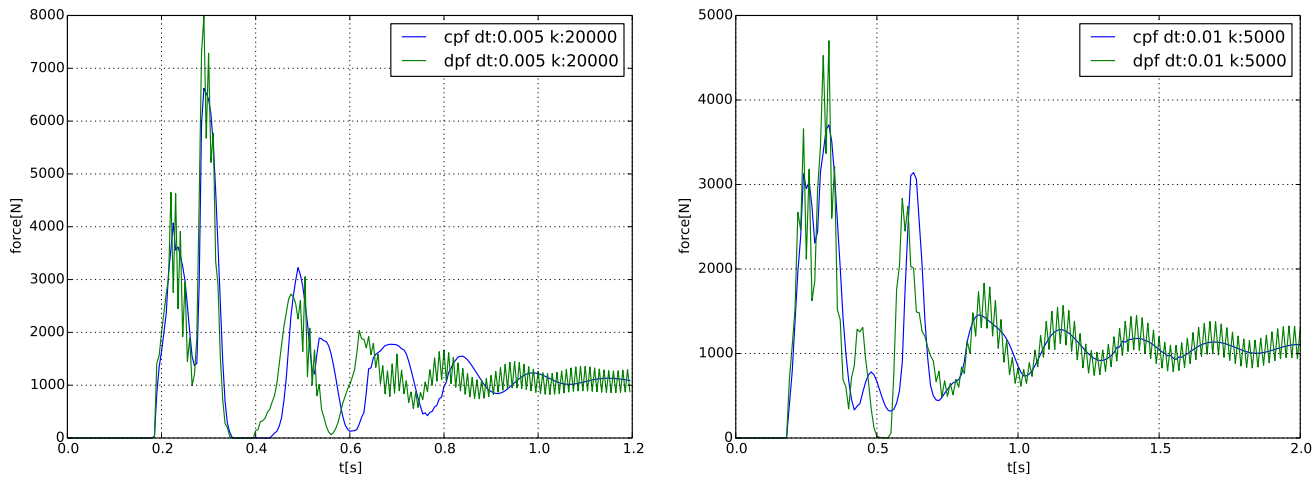
Figure 7.3: Illustration of discretizations of the deformable bar on the rigid cube. In the center of the bar more features are overlapping than at the edging and thus there are more feature collisions.



(a) Repulsion force with DPF (b) Repulsion force with CPF

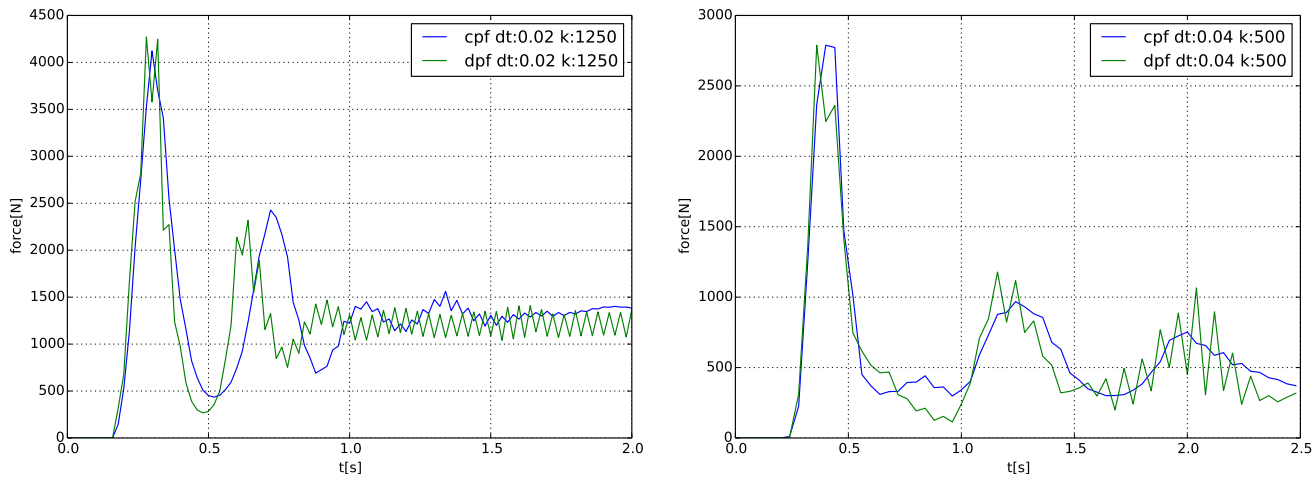
Figure 7.4: Comparison of the repulsion force between a deformable bar and a rigid cube with varying time steps Δt and and constant penalty coefficients $k = 5000$. The left plots show the force with DPF and jitter can be noted for $\Delta t = 0.009$, whereas for the plots on right side with CPF jitter can be initially noted for a larger time step $\Delta t = 0.012$

repulsion forces for time steps sizes $\Delta t = 0.010s$, $\Delta t = 0.011s = \Delta t_c$, $\Delta t = 0.012s$. The overall plots are similar, there are two large peaks in the beginning at $t = 0.3s$ and $t = 0.6s$ with high repulsion forces when the deformable bar collides with the rigid cube and is pushed up again. From $t \approx 0.8s$ on the bar tumbles on the block, arising as a damped oscillation. There is a jitter notable in the force process for the higher time steps of each method. For DPF, especially for $\Delta t = 0.009$, it is notable from $t = 0.7$ on and for CPF it is notable for $\Delta t = 0.012$ from $t = 1.4$ on. This jitter in the force plot is also optically notable as jitter in the movement of vertices.
The jitter mainly takes place at the vertices in the middle of the bottom side of the bar, since there are more feature collisions than at the physical edges due to the discretization of the cube



(a) Δt=0.005, k=20000

(b) Δt=0.01, k=5000



(c) Δt=0.02, k=1250

(d) Δt=0.04, k=500

Figure 7.5: The repulsion force between a deformable bar and a rigid cube with varying time steps dt and penalty coefficients k.

as shown in figure 7.3 c). In the center of the bar, more features overlap than at the physical edge and thus there are more feature collisions.
In sum we have seen that also for a more complex scenario CPF provide stability for larger time steps and validated the results from the particle-plane scenario.
Up to now the penalty coefficient was constant and we varied the time step. In the following we vary both, the time step and the penalty coefficient to reach a comprehensive comparison of the CPF and DPF.
We chose multiple time steps Δt and in order he provide stable simulations compute k according to the relation $\Delta t \sim \sqrt{\frac{m}{k}}$ from the 1D particle analysis. The resulting forces are shown in figure 7.5. For all four cases, CPF provide a much smoother force curve than the DPF. Again, it is also notable in the simulation since there is a notable jitter for DPF.
However, the summed penetration depth per time step, as shown in figure 7.6, is for CPF between 1.4 and 2.0 times larger than for DPF. The DPF algorithm yields larger collison forces

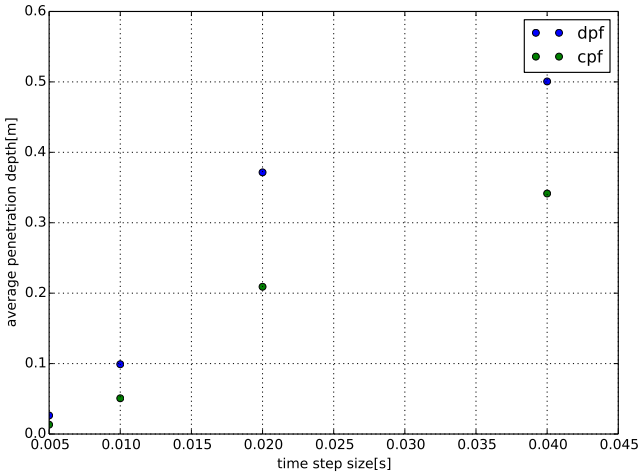
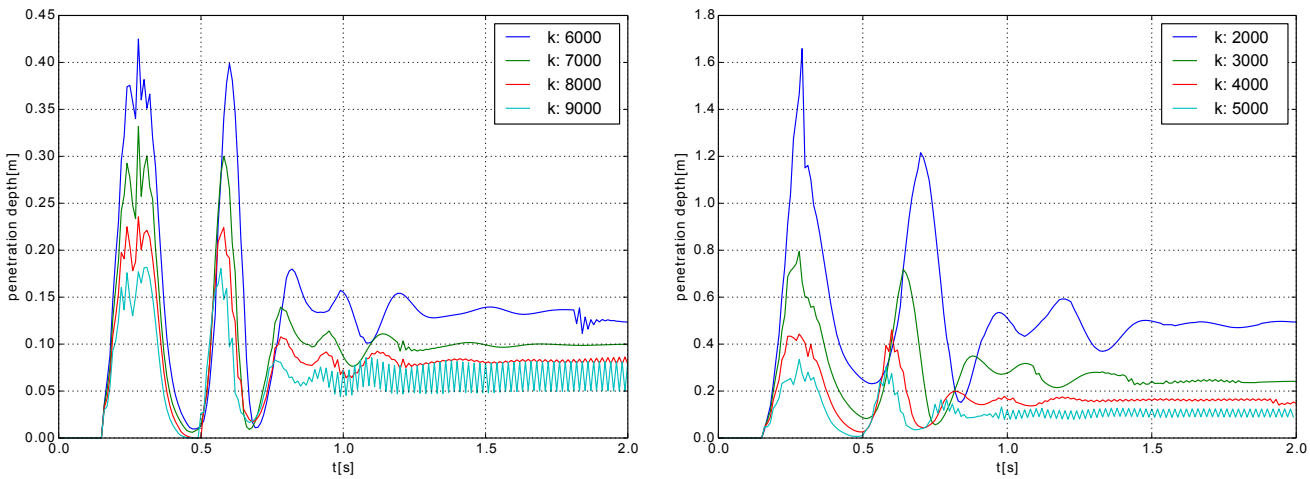


Figure 7.6: The average of the summed penetration depth per time step for a deformable bar dropped on a rigid cube with varying time steps and penalty coefficients. For increasing time steps the penalty coefficients decrease and accordingly the penetration depth increases. It is noticeable that the penetration depth with CPF is between 1.4 and 2.0 times higher than the penetration depth with DPF.

due to the larger overestimation of the penetration depth while the intersection is increasing, as explained in section 5.2.2.
We have shown that the penetration depth for CPF is larger than for DPF with equal penalty coefficients, but for these cases the DPF were already jittering whereas CPF behaved smooth. In order to make a valid statement, we have to investigate the penetration depth and the jitter for varying penalty coefficients.
Again, we drop the deformable bar on the rigid cube, this time with a constant time step $\Delta t = 0.01$ and varying penalty coefficients k. The penetration depths are shown in figure 7.7. For the equilibrium CPF, provide smooth behavior up to k=7000 with a penetration depth of 0.1m, whereas DPF only provide an almost smooth behavior up to $k = 3000$, with a penetration depth of 0.28m.
In this section, we compared the CPF algorithm by Tang et al. [18] to DPF and showed that for a constant penalty coefficient CPF can handle larger time steps without jitter. However, the DPF provide smaller penetration depths with an equal penalty coefficient. Nevertheless, with an increased penalty coefficient for the CPF, CPF are still stable and provide smaller intersections. In comparison, the DPF are with a smaller penalty coefficient already jittering and yield larger penetration depths.
In conclusion the CPF algorithm provides increased stability, less jitter and smaller penetration depths.

7.2 Contact Redefinition

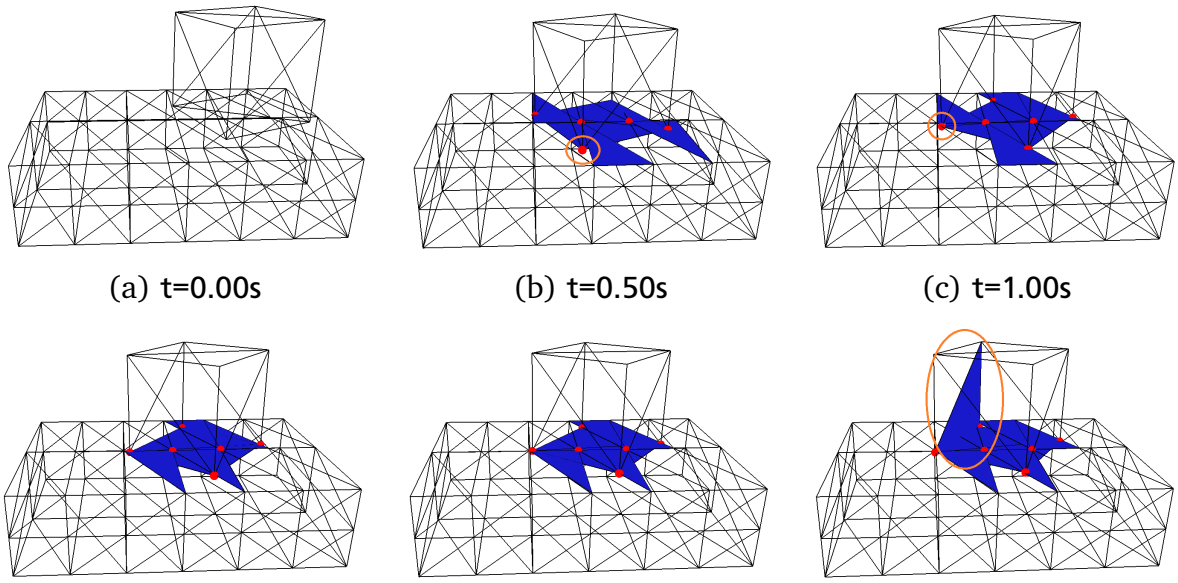
To analyze the new approach of contact redefinition (described in section 6.2) we simulate a rigid cube sliding on a deformable bar with and without the collision redefinition algorithm (see figures 7.8- 7.11). There is no friction applied, therefore the tangential motion should not change.



(a) CPF (b) DPF

Figure 7.7: The penetration depth for CPF and DPF with varying penalty coefficients and constant time step $\Delta t = 0.01s$. (Note the different scaling of the y-axis and the different penalty coefficients). For the equilibrium, CPF provide smooth behavior up to k=8000 with a penetration depth of 0.1m, whereas DPF only provide an almost smooth behavior up to k=3000, with a penetration depth of 0.28m.

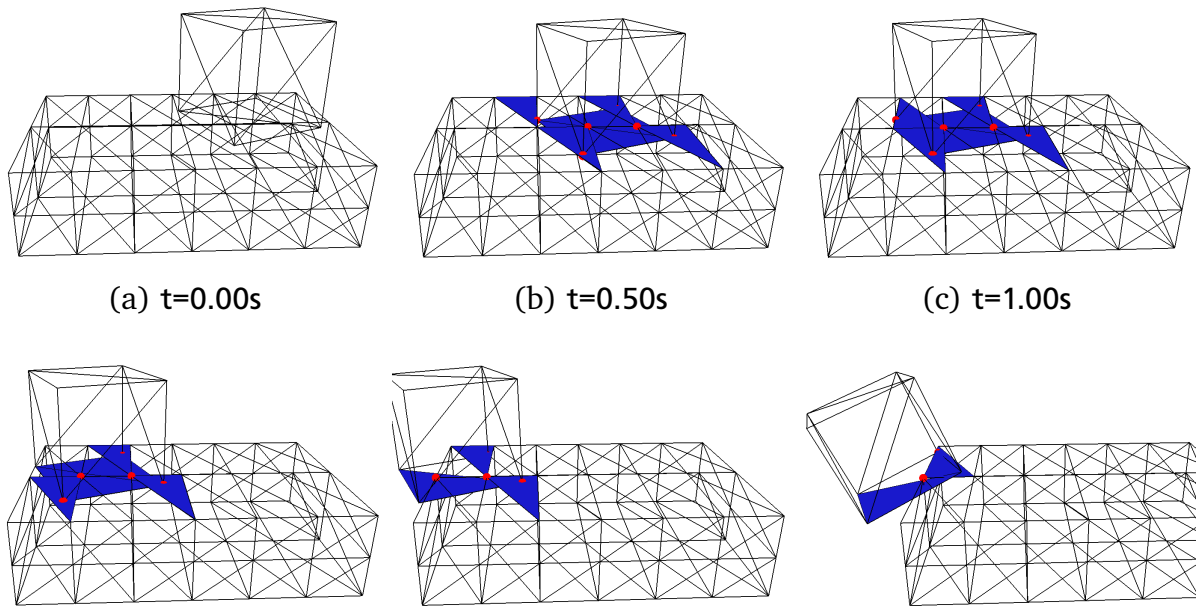
Without the redefinition algorithm the rigid cube changes its velocity in an implausible fashion tangential to the bar and bounces between the surface edges and vertices of the bar. It is hooked between the surface elements. The tangential forces are caused by the highlighted collisions in figure 7.10 (b)-(f) and 7.8 (f). Furthermore, there are implausible collisions in 7.8 (b)and (c). The highlighted vertices 7.8 (b)and (c) would be expected to be colliding with the vertex of the face directly below and not with the neighboring face. In contrast, with the redefinition algorithm (see figure 7.9 and 7.9) the cube's tangential velocity does not change notable, it slides on the bar and tips over the edge. There are no wrong collisions inducing strong tangential forces and no implausible collision pairs.
The repulsion forces are shown in figure 7.12. The plot shows a smooth with several leaps. The leaps are induced by the contact redefinitions. When a sliding contact is redefined, there is a sudden additional geometrically deep collision providing a sudden increase of the contact force. Similarly, if an EE collision is added or removed by the redefinition algorithm, force changes are induced. Overall, these leaps are small and do not induce a notable jitter.
Up to now we analyzed rather simple geometries. Therefore, we will now investigate a more complex scene, with a bunny dropped in a rigid bowl with CPF and the redefinition algorithm. Screenshots of the simulation are shown in figure 7.13.
The bunny drops in the bowl, bounces around for approximately two seconds and then comes to rest. The overall result is visually satisfying. However, a closer look at the bunny shows that when it is colliding some tetrahedrons are inverted , see figure 7.14. Inverted tetrahedrons disturb the visual impression of the scene. They are induced due to large collision impulses by the CPF algorithm and can be reduced by choosing lower penalty coefficients. Nevertheless, reducing the penalty coefficients yields larger penetration depths.



(a) t=0.00s (b) t=0.50s (c) t=1.00s

(d) t=1.50s (e) t=2.00s (f) t=2.50s

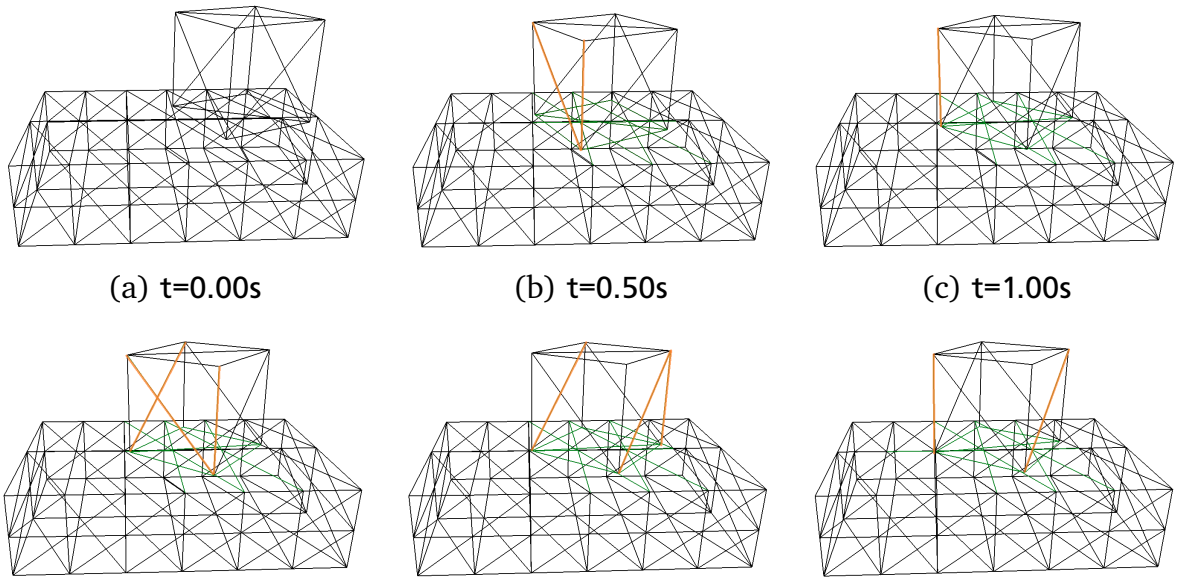
Figure 7.8: A rigid cube sliding on a deformable bar without the collision redefinition algorithm and without friction. The sideway collision pairs induce implausible tangential forces affecting the movement of the rigid cube in an implausible way. The subfigures show active VF collisions, with the vertices in red and the faces in blue, the features inducing artifact forces are highlighted orange.



(a) t=0.00s (b) t=0.50s (c) t=1.00s

(d) t=1.50s (e) t=2.00s (f) t=2.50s

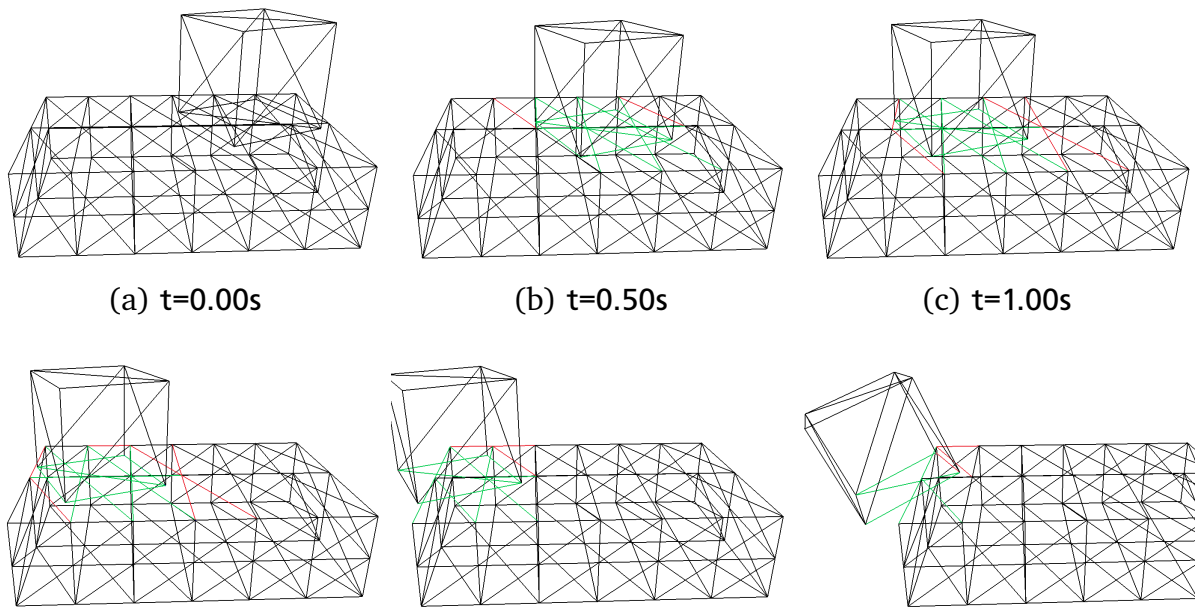
Figure 7.9: A rigid cube sliding on a deformable ba with the collision redefinition algorithm and without friction. With the redefinition algorithm no sideways collision appear and the cube slides plausible on the bar without notable changes in its tangential velocity. The subfigures show active VF collisions, with the vertices in red and the faces in blue.



(a) t=0.00s (b) t=0.50s (c) t=1.00s

(d) t=1.50s (e) t=2.00s (f) t=2.50s

Figure 7.10: A rigid cube sliding on a deformable bar without the collision redefinition algorithm and without friction. The sideway collision pairs induce implausible tangential forces affecting the movement of the rigid cube in an implausible way. The subfigures show EE collisions in green and the collisions inducing implausible tangential forces are highlighted orange.



(a) t=0.00s (b) t=0.50s (c) t=1.00s

(d) t=1.50s (e) t=2.00s (f) t=2.50s

Figure 7.11: A rigid cube sliding on a deformable bar with the collision redefinition algorithm and without friction. With the redefinition algorithm no sideways collision appear and the cube slides plausible on the bar without notable changes in its tangential velocity. The subfigures show active EE collisions in green and candidate collisions in red.

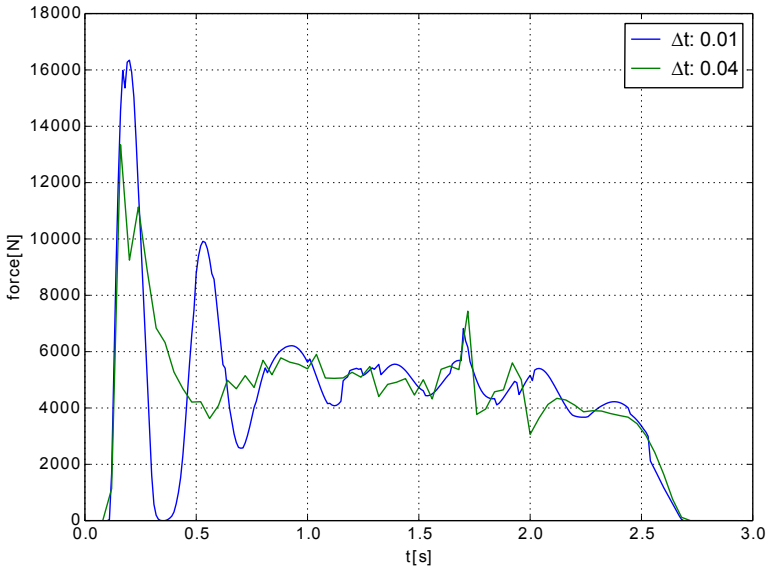
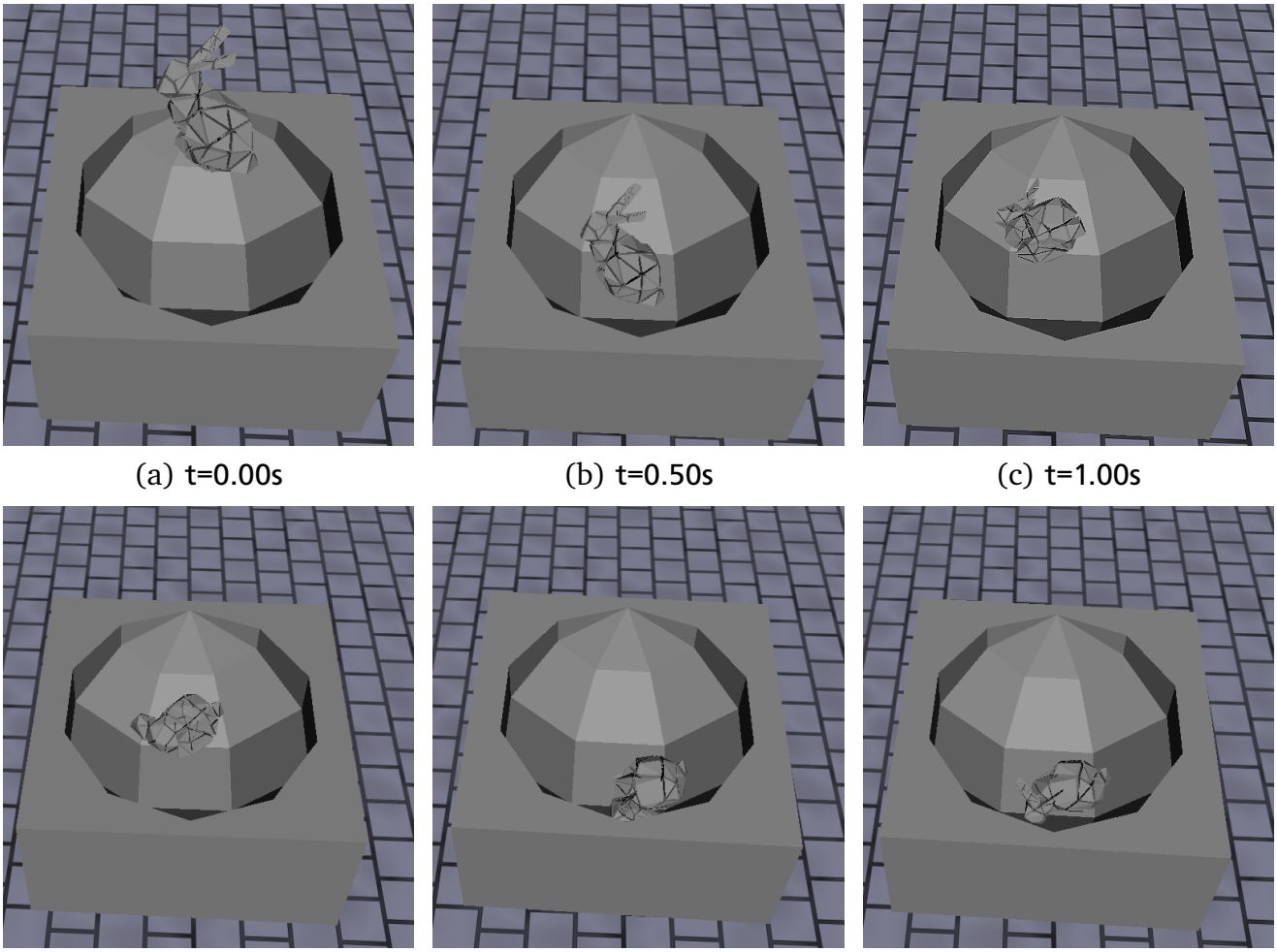


Figure 7.12: Contact force for a rigid cube sliding on a deformable bar



(a) t=0.00s (b) t=0.50s (c) t=1.00s

(d) t=1.50s (e) t=2.00s (f) t=2.50s

Figure 7.13: A deformable bunny is dropped in a rigid bowl, which is dropped on a static plane.

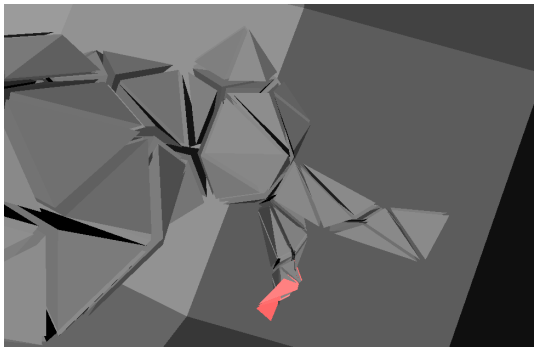


Figure 7.14: Inverted tetrahedrons (marked red) occur when the deformable bunny collides with the rigid bowl.

To sum it up, we have shown in the cube-bar benchmark that the redefinition algorithm provides plausible collision pairs and preserves the tangential motion, whereas with the original CPF algorithm we obtained an implausible stop of the tangential motion. Furthermore, we proved that the redefinition algorithm can handle complex geometries, as shown in the bunny-bowl scene.

7.3 Friction

To achieve a plausible friction behavior, the friction model should induce forces according to Coulombs law. Dynamic friction should slow down the tangential movement. Static friction should stop any tangential movement if the tangential velocity is close to zero and the tangential force $\mathbf{f}_t$ is smaller than the product of the static friction coefficient μ_S and the normal force $\mathbf{f}_n$.

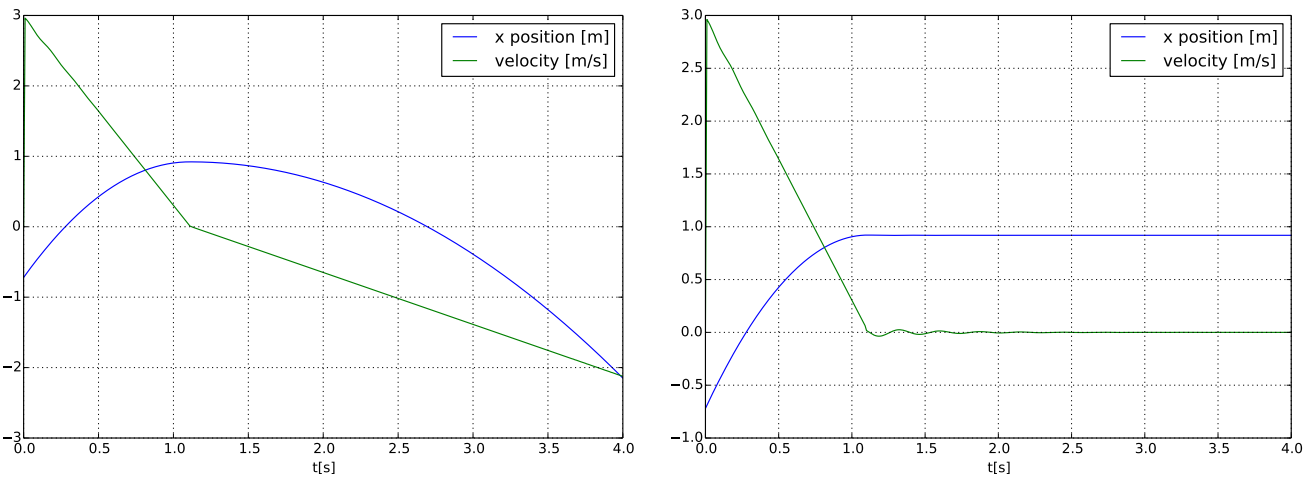
To analyze the friction model, we investigate a rigid cube with mass $m = 100kg$ sliding on a $10°$ slope. To test the static friction, the velocity of the rigid cube is initialized with $v_0 = 3\frac{m}{s}$ upwards the plane. We used continuous penalty forces with a penalty coefficient $k = 20000$. The dynamic friction coefficient is set to $\mu_d = 0.1$ and the static friction is set to $\mu_s = 0.3$ and $\mu_s = 0.4$. The friction penalty coefficients are set to $k_{fp} = 100$, $k_{fd} = 25$ and $k_f = 10000$.

The movement and the velocity of the cube are shown in figure 7.15. For the lower coefficient $\mu_s = 0.3$ the cube slides up the plane with decreasing velocity, decelerated by gravity and dynamic friction. At $t \approx 1.2s$ it reaches the maximum height and starts sliding down again with increasing velocity, this time accelerated by gravity and slightly decelerated by dynamic friction.

For the higher coefficient $\mu_s = 0.4$ the cube starts exactly equal to the previous one. However, when it reaches the top it remains at the position, due to the increased static friction coefficient. The static friction is now larger than the downhill-slope force. The velocity shows a small damped oscillation at $t \approx 1.2s$, it is induced since we model the static friction by applying a damped spring at the point where the cube is to be hold.

Comparing the results with the analytical solution of Coulomb's Friction law, we would expect the block to stop for $\mu_s > 0.176$, whereas in our simulation the block stopped for $\mu_s \gtrsim 0.4$. Nevertheless the friction model provides results analogously to Coulomb's Friction law, providing satisfying results for an interactive environment.

To analyze the friction between multiple bodies we place a rigid cube with an initial velocity on a deformable bar and simulate two different friction coefficients. Furthermore, to test for



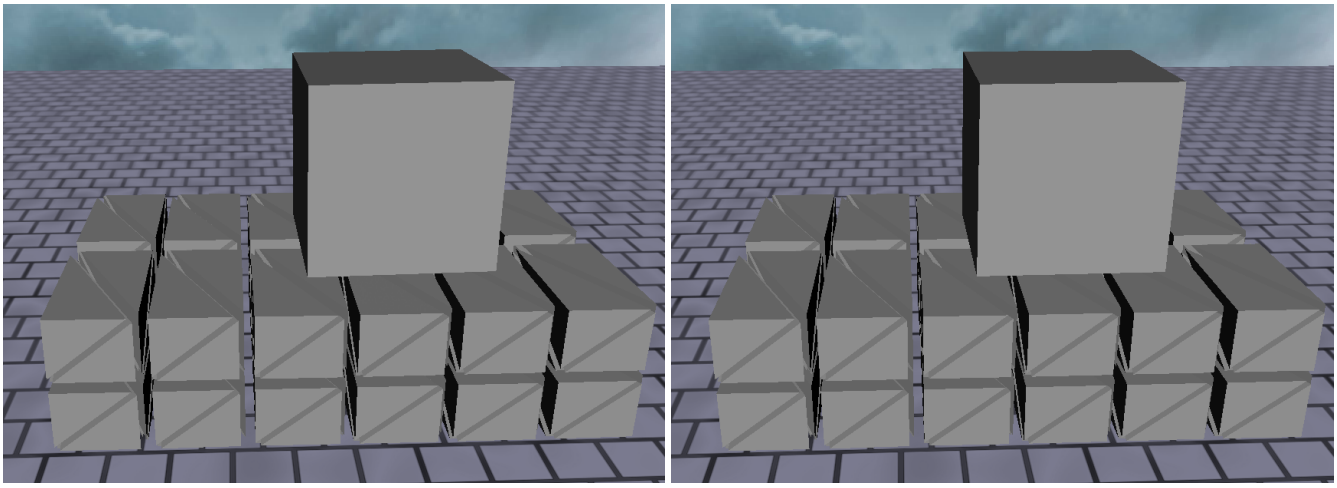
(a) static friction coefficients $\mu_s = 0.3$ (b) static friction coefficients $\mu_s = 0.4$

Figure 7.15: Position and velocity over time of a rigid cube sliding on a slope with different static friction coefficients μ_s. With the lower coefficient $\mu_s = 0.3$ the cube slides up the slope until its velocity reaches zero, then it starts sliding down. Whereas, with the higher coefficient μ_s the cube stops after its velocity reached zero and stays at its position.

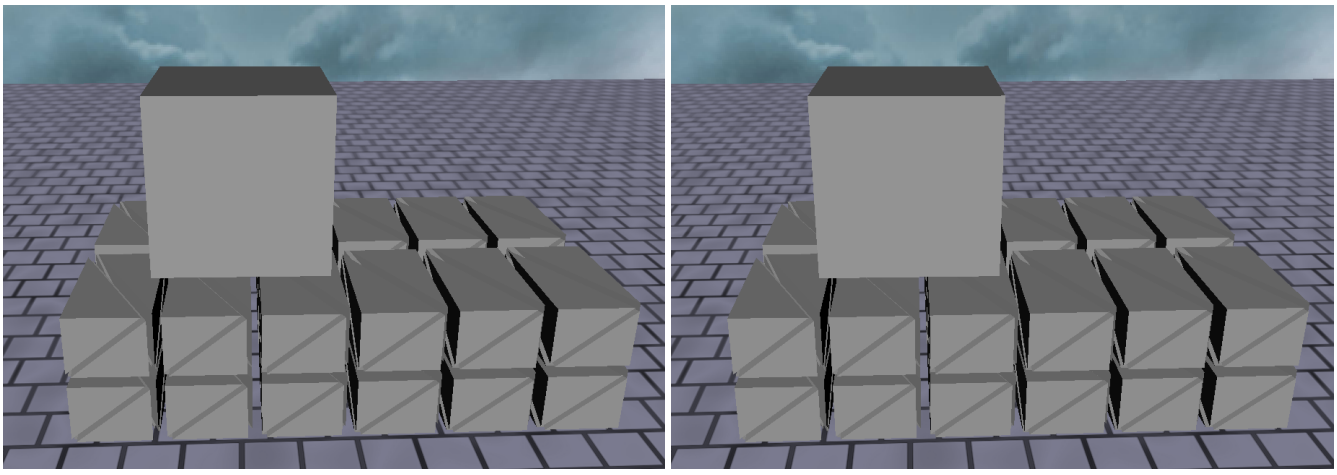
numerical drift in the position due to the static friction we will analyze the simulation for a large number of time steps (20000).
Screenshots of the scene are shown in figure 7.16. The cube starts at the right side with its initial velocity in the left direction. With the higher friction coefficients in a) and b) the cube is stronger decelerated as with the lower coefficients in c) and d). After the cube stopped it movements (a) and c), it rests exactly at its position. Even after 20000 steps there is no numerical drift in the position notable. This a result of the static friction modeled with a fixed spring. Therefore the bodies can make small movements around the fixed position, but no optically notable movement or position drift is possible.
In conclusion the penalty-based friction model [23] applies a plausible modeling of dynamic and static friction with no notable numerical drift in the position for the static friction. The model does not grant the analytical description of Coulomb's friction law. However it provides analogue results, which are satisfying for the use in an interactive environment.

7.4 Computational Cost

The computational cost is a crucial property for interactive applications. Therefore, we investigate the computational cost of the CPF in comparison to the DPF with regard to the results from the stability analysis.
We analyzed the scene with a deformable bunny dropped into a rigid bowl from section 7.2 (see figure 7.13) using an Intel Core i7-2630QM 2.0Ghz processor with 6.0GB RAM and a Nvidia Geforce 540M graphics card. To provide an accurate comparison of the CPF and DPF, we simulate the scene with CPF and compute for each feature pair the impulse with both algorithms, though DPF computation is only for time measurements and the impulses are not applied. We



(a) $\mu_s = 0.8$, $\mu_d = 0.1$, $t = 0.2s$ (b) $\mu_s = 0.8$, $\mu_d = 0.1$, $t = 20.0s$



(c) $\mu_s = 0.3$, $\mu_d = 0.05$, $t = 0.35s$ (d) $\mu_s = 0.3$, $\mu_d = 0.05$, $t = 20.0s$

Figure 7.16: A rigid cube sliding from the right to the left on a deformable bar with two friction coefficient combinations. (a) and (c) show the cube right after the static friction set in. (b) and (d) show the cube after two seconds. With the lower valued coefficient combination the cube keeps sliding further than with higher combination due to dynamic friction. After the blocks stopped they rest at their position due to static friction.

measured five runs with 200 simulation steps each and detecting the time for the handling of each feature and its components. The averaged times are shown in figure 7.17.

The handling for a single EE pair takes average $6.8\mu s$, whereas a VF pair is slightly slower and takes average $7.8\mu s$. The difference is mainly caused by the overall most expensive part of the collision handling, the categorization of the collision. For the EE pair, the collision categorization takes average $3.11\mu s$ and thereby 45.7% of the overall time for one feature pair. In the VF case, the categorization takes average $4.07\mu s$ and thereby 53.1% of the overall time for one feature pair. The difference in the computation time of the VF and EE categorization can be explained by examining the number of tested normals. A vertex is typically adjacent to six faces and accordingly six normals need to be tested. Whereas, in the EE categorization each edge has only two directly adjacent faces, yielding only four normals to be tested. In sum, there are to more

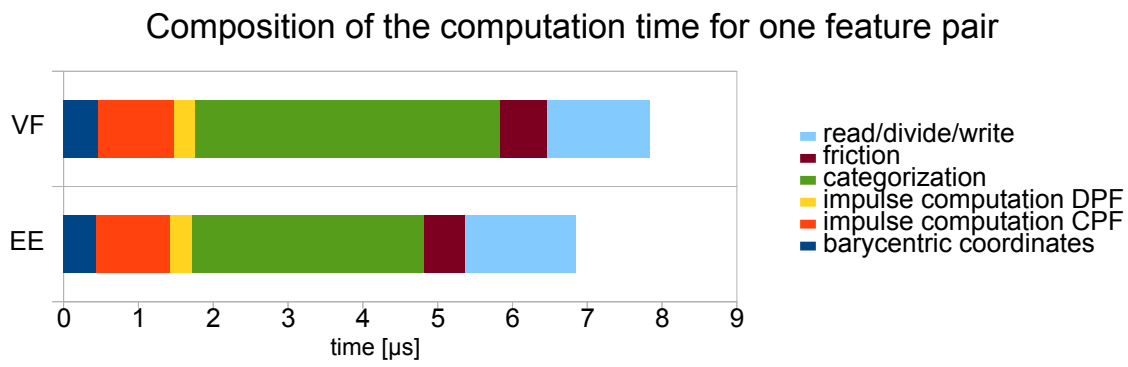


Figure 7.17: Composition of the computation time for one feature pair

tests for the VF case and consequently the VF categorization is more expensive. All other parts take almost the same time for VF and EE. It is remarkable that the actual impulse computation in the VF case with CPF takes only 12.9% (EE 14.6%) of the overall computation time and for DPF only 3.7% (EE 4.1%).

A comparison of the computation time for one impulse with CPF and DPF is shown in figure 7.18. The computation of the impulse with CPF takes in the VF case $1,01\mu s$ (EE $1.00\mu s$) and for DPF in the VF case it takes $0,29\mu s$ (EE $0.28\mu s$). Hence, the DPF need only a third of the time the CPF need for the computation of one impulse. The computational overhead of the CPF seems reasonable, since the CPF require the computation of degree-six polynomial, whereas the DPF correspond to a linear polynomial. However, the impulse computation is only one component in the handling of a collision. In order gain a valid statement, we need to compare the overall computation time for a feature with DPF and CPF. With DPF the handling of one feature pair takes $6.83\mu s$ (EE $5.84\mu s$), whereas it takes $7.55\mu s$ (EE $6.56\mu s$) with CPF. The handling with CPF takes 11.8% longer than the handling with DPF. In section 7.1 we have shown that the maximum time step for CPF is 1.22 times larger than for DPF. Combining the computational overhead with increased step size yields a theoretical speed-up of 9.12% for CPF. However, this result provides only limited validity, since DPF require only a DCD, whereas CPF require a CCD. In general, CCD provides a higher computational cost than DCD.

To compare the computational cost of DPF with a state of the art DCD to the computational cost of CPF with a state of the art CCD would yield an interesting and meaningful result in order to evaluate the actual difference in computational cost. However, the CCD in our framework is in a prototype state and far from optimal speed. Therefore, we cannot provide this analysis. However, Tang et al. [18] compared DPF+DCD to CPF+CCD and showed that CPF+CCD are about 3 - 4 times slower than DPF+DCD.

In conclusion, the redefinition algorithm provides a significant additional cost by almost doubling the time needed to handle one collision pair. However, the additional cost is necessary in order to reduce the CPF artifacts, since otherwise no simulation of sliding contacts is feasible. The additional cost induced by the CPF in comparison to the DPF is small ($+11,8\%$). However, CPF require CCD. Taking the collision detection into account and comparing the CCD+CPF to DCD+DPF shows that CPF+CCD are about 3 - 4 times slower than DPF+DCD [18].

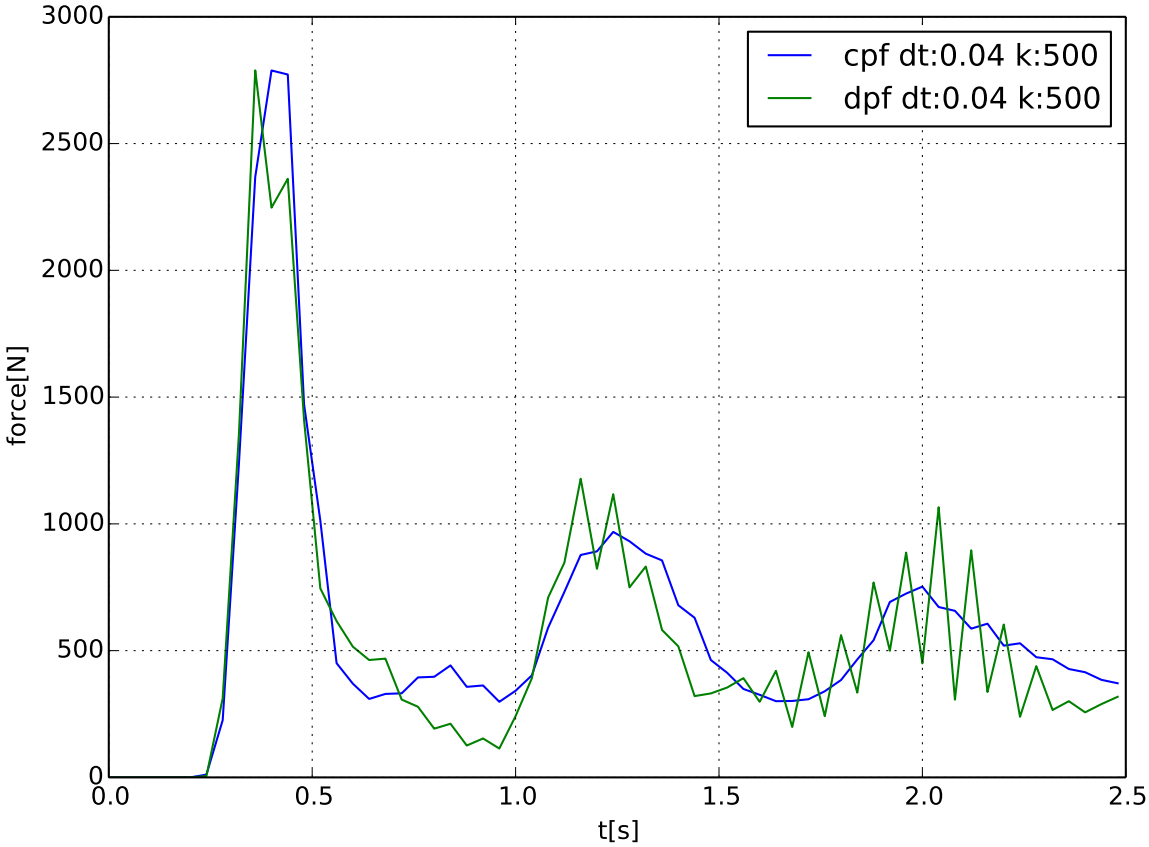


Figure 7.18: Comparison of the computation time for one impulse with CPF and DPF

8 Discussion

In this thesis, we present a unified system to handle collisions between rigid and deformable bodies with friction. We modified the integration scheme by Bridson et al. [4] to handle rigid and deformable bodies, applied a CCD [16] and handled the detected collisions with CPF [18]. Friction is applied with a penalty based approach [23] providing static and dynamic friction and a straightforward integration in the integration scheme.
We compared the CPF algorithm by Tang et al. [18] to a DPF algorithm and showed that the CPF provide more stability, less jitter and smaller penetration depths with a small computational overhead if a CCD algorithm is applied for both. DPF in comparison to CPF provide only a notable computational cost advantage if DCD is used, since CCD is more expensive than DCD.
We develop an advanced collision resolution condition and an algorithm based on the redefinition of feature pairs for non-deep intersections to handle the observed artifacts of separated features still assumed colliding, additional artificial collisions through discretization and inconsistent feature pair mapping. Our new algorithm detects artifact collisions, by defining normal cone alike spans for the colliding features and checking their position. The detected artifact collisions are redefined in order to provide a consistent collision handling.
As shown in our benchmarks the new algorithm solves the issue of collisions remaining, reduces the artifact tangential forces induced by the CPF algorithm and provides a more plausible behavior. The algorithm is limited since we keep the CPF normal definition, and thereby in many cases the tangential component can only be reduced.
In the future we want to solve the current drawback of the redefinition algorithm that we can only handle non-deep contacts, since we cannot differentiate between deep contacts and sliding contacts. A possible solution for deep collisions could be to check the neighbors of the new colliding vertex for collisions and to take their contact normal as a reference. Furthermore it may be interesting to investigate alternative contact normal definitions, taking the adjacent elements of the vertices and edges into account and integrating them into the CPF algorithm. Moreover, alternative integration of the impulses in the deformable bodies could reduce instability and the inverting of tetrahedrons. A possible approach would be to divide the impulse on the neighboring vertices. Furthermore, the stability could be increased by the application of an implicit integration method. Additionally, an automatic computation of the penalty coefficient would increase the user-friendliness. A possible basis could be the stability ratio of section 7.1: $\Delta t \sim \sqrt{\frac{m}{k}}$. In particular, for scenes with large amounts of colliding features a GPU-based implementation would provide a speedup, since the CPF impulses and the contact categorizations could be computed parallel.

Bibliography

[1] T. Akenine-Moller, T. Moller, and E. Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2002. ISBN 1568811829.

[2] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 862–870, New York, NY, USA, 2003. ACM. ISBN 1-58113-709-5. doi: 10.1145/1201775.882357. URL `http://doi.acm.org/10.1145/1201775.882357`.

[3] J. Bender. *Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität*. Univ.-Verlag Karlsruhe, 2007.

[4] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 594–603, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. doi: 10.1145/566570.566623. URL `http://doi.acm.org/10.1145/566570.566623`.

[5] R. D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471107743.

[6] E. Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 4(6), 2006.

[7] D. Gross, W. Hauger, J. Schröder, and W. A. Wall. *Technische Mechanik 1*. Springer-Lehrbuch. Springer, 11 edition, 2011. ISBN 9783642138058.

[8] M. Hutter and A. Fuhrmann. Optimized continuous collision detection for deformable triangle meshes. 2007.

[9] B. Keiser, M. Heidelberger, R. Teschner, M. Gross, and M. Mueller. Consistent penetration depth estimation for deformable collision response. In *Vision, Modeling, and Visualization 2004: Proceedings, November 16-18, 2004, Standford, USA*, page 339. IOS Press, 2004.

[10] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, pages 602–608, 1998.

[11] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan. Fast simulation of mass-spring systems. *ACM Transactions on Graphics*, 32(6):209:1–7, Nov. 2013. URL `http://graphics.berkeley.edu/papers/Liu-FSM-2013-11/`. Proceedings of ACM SIGGRAPH Asia 2013, Hong Kong.

[12] T. Möller. A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30, 1997.

[13] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *SIGGRAPH Comput. Graph.*, 22(4):289–298, June 1988. ISSN 0097-8930. doi: 10.1145/378456.378528. URL `http://doi.acm.org/10.1145/378456.378528`.

[14] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society. ISBN 1-56881-227-2. URL http://dl.acm.org/citation.cfm?id=1006058.1006087.

[15] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 471–478, New York, NY, USA, 2005. ACM. doi: 10.1145/1186822.1073216. URL http://doi.acm.org/10.1145/1186822.1073216.

[16] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In D. Thalmann and M. van de Panne, editors, *Computer Animation and Simulation '97*, Eurographics, pages 177–189. Springer Vienna, 1997. ISBN 978-3-211-83048-2. doi: 10.1007/978-3-7091-6874-5_13. URL http://dx.doi.org/10.1007/978-3-7091-6874-5_13.

[17] S. P. Serna, J. G. B. R. da Silva, A. Stork, and A. F. Marcos. Neighboring-based linear system for dynamic meshes. In *Workshop in Virtual Reality Interactions and Physical Simulation*, pages 95–103. The Eurographics Association, 2009.

[18] M. Tang, D. Manocha, M. A. Otaduy, and R. Tong. Continuous penalty forces. *ACM Trans. Graph.*, 31(4):107:1–107:9, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185603. URL http://doi.acm.org/10.1145/2185520.2185603.

[19] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, et al. Collision detection for deformable objects. In *Computer Graphics Forum*, volume 24, pages 61–81. Wiley Online Library, 2005.

[20] D. Weber, T. Kalbe, A. Stork, D. Fellner, and M. Goesele. Interactive deformable models with quadratic bases in Bernstein-Bézier-form. *TVC*, 27:473–483, 2011. ISSN 0178-2789.

[21] D. Weber, J. Bender, M. Schnoes, A. Stork, and D. Fellner. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum*, 32(1):16–26, 2013. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2012.03227.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.03227.x.

[22] R. Weber. *Physik: Teil I: Mechanik und Thermodynamik ; experimentelle und theoretische Grundlagen*. Physik. Books on Demand, 2012. ISBN 9783844829501. URL http://books.google.de/books?id=A95dewBQ7v8C.

[23] K. Yamane and Y. Nakamura. Stable penalty-based model of frictional contacts. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1904–1909. IEEE, 2006.

[24] X. Zhang, Y. J. Kim, and D. Manocha. Continuous penetration depth. *Comput. Aided Des.*, 46:3–13, Jan. 2014. ISSN 0010-4485. doi: 10.1016/j.cad.2013.08.013. URL http://dx.doi.org/10.1016/j.cad.2013.08.013.