Position-Based Simulation Methods in Computer Graphics

Jan Bender[1], Matthias Müller[2] and Miles Macklin[2]

[1]Graduate School CE, TU Darmstadt
[2]NVIDIA PhysX Research

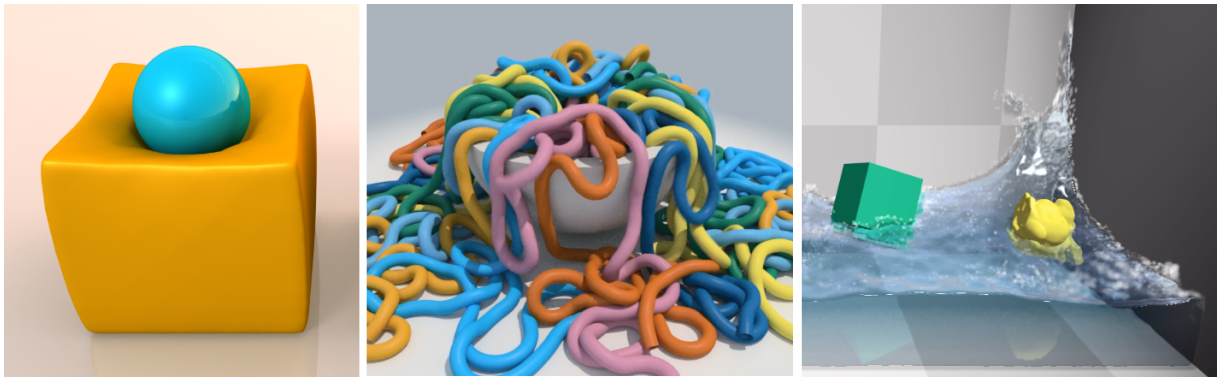

Figure 1: *A selection of scenes simulated inside the position-based dynamics framework.*

Abstract

The physically-based simulation of mechanical effects has been an important research topic in computer graphics for more than two decades. Classical methods in this field discretize Newton's second law and determine different forces to simulate various effects like stretching, shearing, and bending of deformable bodies or pressure and viscosity of fluids, to mention just a few. Given these forces, velocities and finally positions are determined by a numerical integration of the resulting accelerations.

In the last years position-based simulation methods have become popular in the graphics community. In contrast to classical simulation approaches these methods compute the position changes in each simulation step directly, based on the solution of a quasi-static problem. Therefore, position-based approaches are fast, stable and controllable which make them well-suited for use in interactive environments. However, these methods are generally not as accurate as force-based methods but still provide visual plausibility. Hence, the main application areas of position-based simulation are virtual reality, computer games and special effects in movies and commercials.

In this tutorial we first introduce the basic concept of position-based dynamics. Then we present different solvers and compare them with the classical implicit Euler method. We discuss approaches to improve the convergence of these solvers. Moreover, we show how position-based methods are applied to simulate hair, cloth, volumetric deformable bodies, rigid body systems and fluids. We also demonstrate how complex effects like anisotropy or plasticity can be simulated and introduce approaches to improve the performance. Finally, we give an outlook and discuss open problems.

Keywords: physically-based animation, position-based dynamics, deformable solids, rigid bodies, fluids

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

© The Eurographics Association 2015.

Contents

1 Tutorial Details **3**
1.1 Presenters . . . 3
1.2 Length . . . 3
1.3 Necessary Background . . . 3
1.4 Potential Target Audience . . . 3

2 Introduction **4**

3 Background **4**
3.1 Equations of Motion . . . 4
3.2 Time Integration . . . 5
3.3 Constraints . . . 5

4 The Core Of Position Based Dynamics **6**
4.1 The Algorithm . . . 6
4.2 Solver . . . 7

5 Specific Constraints **10**
5.1 Stretching . . . 10
5.2 Bending . . . 10
5.3 Isometric Bending . . . 10
5.4 Collisions . . . 11
5.5 Volume Conservation . . . 12
5.6 Long Range Attachments . . . 14
5.7 Strands . . . 14
5.8 Continuous Materials . . . 15
5.9 Rigid Body Dynamics . . . 17
5.10 Fluids . . . 18
5.11 Shape Matching . . . 19

6 Implementation **26**
6.1 Parallelization . . . 26
6.2 Unified Solver . . . 27

7 Applications **27**
7.1 Strain Limiting . . . 27
7.2 Wrinkle Meshes . . . 28
7.3 Further Applications . . . 29

8 Conclusion **29**

References **29**

© The Eurographics Association 2015.

1. Tutorial Details

1.1. Presenters

Jan Bender Until 2002 Jan Bender studied computer science at the University of Karlsruhe with the majors computer graphics and software engineering. Afterwards, he started with his PhD studies at the Institute for Computer Graphics at the University of Karlsruhe. In the beginning of 2007 he received his doctoral degree with distinction. The topic of his thesis was the dynamic simulation of multibody systems in VR applications. In the following years he worked in a research department for driver assistance systems in industry. Since 2010 he is employed as Assistant Professor at the Department of Computer Science, TU Darmstadt and works at the Graduate School of Computational Engineering in Darmstadt which has been recognized as a center for top-level research by the highly competitive 'Excellence Initiative' of the German government. His current research areas include: interactive simulation of multibody systems, deformable solids and cloth, position-based methods, collision detection and resolution, fracture, fluid simulation and real-time visualization. Since 2009 Jan Bender is program chair of the Virtual Reality Interaction and Physical Simulation conference.

- email address: bender@gsc.tu-darmstadt.de
- URL: *www.interactive-graphics.de*

Matthias Müller Matthias Müller received his PhD in atomistic simulation of dense polymer systems in 1999 from ETH Zürich. During his post-doc with the MIT Computer Graphics Group (1999-2001), he changed fields to macroscopic physically based simulations. He has published papers on particle-based water simulation and visualization, finite element-based soft bodies, cloth simulation, and fracture simulation. The main focus of his research are unconditionally stable, fast and controllable simulation techniques for the use in computer games. Most relevant to this tutorial, he is one of the founders of the field of position based simulation methods.

In 2002, he co-founded the game middleware company NovodeX (acquired in 2004 by AGEIA), where he was head of research and responsible for extension of the physics simulation library PhysX by innovative new features. He has been head of the PhysX research team of NVIDIA since that company acquired AGEIA Technologies, Inc. in early 2008.

- email address: matthiasm@nvidia.com
- URL: *www.matthias-mueller-fischer.ch*

Miles Macklin Miles Macklin is a researcher on the PhysX research team at NVIDIA. Since 2013, he has been working on a unified physics library called NVIDIA Flex, built using Position-Based Dynamics. The technology in Flex has been the subject of two papers at SIGGRAPH and has now been released to several games and visual effects studios. Prior to joining NVIDIA, Miles worked in the games industry as a visual effects engineer at Sony Computer Entertainment on early Playstation3 development, Rocksteady Studios in London on the Batman Arkham series, and LucasArts in San Francisco on the Star Wars franchise. His current research is focused on real-time methods for simulation and rendering using GPUs.

- email address: mmacklin@nvidia.com
- URL: *www.mmacklin.com*

1.2. Length

This is a half day tutorial (180 minutes).

1.3. Necessary Background

In our tutorial we will make a short introduction in the basics of physically-based animation. However, general knowledge in this area is recommended.

1.4. Potential Target Audience

This tutorial is intended for researchers and developers in the area of computer animation who are interested in interactive physically-based simulation methods. This will be an intermediate level tutorial.

© The Eurographics Association 2015.

2. Introduction

The simulation of solid objects such as rigid bodies, soft bodies or cloth has been an important and active research topic in computer graphics for more than 30 years. The field was introduced to graphics by Terzopoulos and his colleagues in the late eighties [TPBF87a]. Since then, a large body of work has been published and the list is growing rapidly. There exists a variety of survey papers [GM97, MTV05, NMK*06, MSJT08, BET14] which document this development.

In this tutorial we focus on a special class of simulation methods, namely position-based approaches [BMO*14]. These methods were originally developed for the simulation of solids. However, some recent works demonstrated that the position-based concepts can even be used to simulate fluids and articulated rigid bodies. Classical dynamics simulation methods formulate the change of momentum of a system as a function of applied forces, and evolve positions through numerical integration of accelerations and velocities. Position-based approaches, instead, compute positions directly, based on the solution to a quasi-static problem.

Physical simulation is a well studied problem in the computational sciences and therefore, many of the well established methods have been adopted in graphics such as the Finite Element Method (FEM) [OH99], the Finite Differences Method [TPBF87b], the Finite Volume Method [TBHF03], the boundary element method [JP99] or particle-based approaches [DSB99, THMG04]. The main goal of computer simulations in computational physics and chemistry is to replace real-world experiments and thus, to be as accurate as possible. In contrast, the main applications of physically based simulation methods in computer graphics are special effects in movies and commercials and more recently, computer games and other interactive systems. Here, speed and controllability are the most important factors and all that is required in terms of accuracy is visual plausibility. This is especially true for real-time applications.

Position-based methods are tailored particularly for use in interactive environments. They provide a high level of control and are stable even when simple and fast explicit time integration schemes are used. Due to their simplicity, robustness and speed these approaches have recently become very popular in computer graphics and in the game industry.

Collision detection is an important part of any simulation system. However, an adequate discussion of this topic is beyond the scope of this tutorial. Therefore, we refer the reader to the surveys of Lin and Gottschalk [LG98] and the one of Teschner et al. [TKH*05].

3. Background

The most popular approaches for the simulation of dynamic systems in computer graphics are force based. Internal and external forces are accumulated from which accelerations are computed based on Newton's second law of motion. A time integration method is then used to update the velocities and finally the positions of the object. A few simulation methods (most rigid body simulators) use impulse based dynamics and directly manipulate velocities [BFS05, Ben07]. In contrast, geometry-based methods omit the velocity layer as well and immediately work on the positions. The main advantage of a position-based approach is its controllability. Overshooting problems of explicit integration schemes in force based systems can be avoided. In addition, collision constraints can be handled easily and penetrations can be resolved completely by projecting points to valid locations.

Among the force based approaches, one of the simplest methods is to represent and simulate solids with mass-spring networks. A mass spring system consists of a set of point masses that are connected by springs. The physics of such a system is straightforward and a simulator is easy to implement. However, there are some significant drawbacks of the simple method.

- The behavior of the object depends on the way the spring network is set up.
- It can be difficult to tune the spring constants to get the desired behavior.
- Mass spring networks cannot capture volumetric effects directly such as volume conservation or prevention of volume inversions.

The Finite Element Method solves all of the above problems because it considers the entire volume of a solid instead of replacing it with a finite number of point masses. Here, the object is discretized by splitting the volume into a number of elements with finite size. This discretization yields a mesh as in the mass spring approach in which the vertices play the role of the mass points and the elements, typically tetrahedra, can be viewed as generalized springs acting on multiple points at the same time. In both cases, forces at the mass points or mesh vertices are computed due to their velocities and the actual deformation of the mesh.

In this tutorial we focus on position-based simulation methods which omit the velocity and acceleration layer and directly modify the positions. In the following we first introduce the basics of physically-based simulation, before we present the position-based concept in the next section.

3.1. Equations of Motion

Each particle i has three attributes, namely its mass m_i, its position $\mathbf{x}_i$ and its velocity $\mathbf{v}_i$. The equation of motion of a particle is derived from Newton's second law:

$$\dot{\mathbf{v}}_i = \frac{1}{m_i}\mathbf{f}_i, \tag{1}$$

where $\mathbf{f}_i$ is the sum of all forces acting on particle i. The relationship between $\dot{\mathbf{x}}$ and $\mathbf{v}$ is described by the velocity

© The Eurographics Association 2015.

kinematic relationship:

$$\dot{\mathbf{x}}_i = \mathbf{v}_i. \tag{2}$$

While particles have only three translational degrees of freedom, rigid bodies have also three rotational ones. Therefore, a rigid body requires additional attributes, namely its inertia tensor $\mathbf{I}_i \in \mathbb{R}^{3\times3}$, its orientation, which is typically represented by a unit quaternion $\mathbf{q}_i \in \mathbb{H}$, and its angular velocity $\omega_i \in \mathbb{R}^3$. For a rigid body generally a local coordinate system is chosen so that its origin is at the center of mass and its axes are oriented such that the inertia tensor is diagonal in local coordinates.

Newton's second law actually applies only to particles. By viewing rigid bodies as collections of infinite numbers of particles, Euler extended this law to the case of rigid bodies. Therefore, the equations of motion for rigid bodies are also known as the Newton-Euler equations. The equation of motion for the rotational part of a rigid body is:

$$\dot{\omega}_i = \mathbf{I}_i^{-1}\left(\tau_i - (\omega_i \times (\mathbf{I}_i\omega_i))\right), \tag{3}$$

where τ_i is the sum of all moments. A moment can be a pure moment or a byproduct of a force $\tau = (\mathbf{p}-\mathbf{x})\times\mathbf{f}$ if the force $\mathbf{f}$ acts at a point $\mathbf{p}$ and $\mathbf{x}$ is the center of mass of the body. The velocity kinematic relationship for the rotational part is defined by

$$\dot{\mathbf{q}}_i = \frac{1}{2}\tilde{\omega}_i\mathbf{q}_i, \tag{4}$$

where $\tilde{\omega}_i$ is the quaternion $[0,\omega_i^x,\omega_i^y,\omega_i^z]$.

3.2. Time Integration

A simulation step for an unconstrained particle or rigid body is performed by numerical integration of Equations (1)-(2) or Equations (1)-(4), respectively. The most popular integration method in the field of position-based dynamics is the symplectic Euler method which is introduced in the following.

In contrast to the well-known explicit Euler, the symplectic Euler uses the velocity at time $t_0+\Delta t$ instead of time t_0 for the integration of the position vector. The time integration for a particle is then performed by the following equations:

$$\begin{aligned}\mathbf{v}_i(t_0+\Delta t) &= \mathbf{v}_i(t_0)+\Delta t\,\frac{1}{m_i}\mathbf{f}_i(t_0)\\ \mathbf{x}_i(t_0+\Delta t) &= \mathbf{x}_i(t_0)+\Delta t\,\mathbf{v}_i(t_0+\Delta t).\end{aligned}$$

In the case of a rigid body also Equations (3) and (4) must be integrated. Using the symplectic Euler method this yields:

$$\begin{aligned}\omega_i(t_0+\Delta t) &= \omega_i(t_0)+\Delta t\,\mathbf{I}_i^{-1}(t_0)\cdot\\ &\quad(\tau_i(t_0)-(\omega_i(t_0)\times(\mathbf{I}_i(t_0)\omega_i(t_0))))\\ \mathbf{q}(t_0+\Delta t) &= \mathbf{q}(t_0)+\Delta t\,\frac{1}{2}\tilde{\omega}_i(t_0+\Delta t)\mathbf{q}_i(t_0).\end{aligned}$$

Note that due to numerical errors the condition $\|\mathbf{q}\|=1$, which must be satisfied by a quaternion that represents a rotations, can be violated after the integration. Therefore, the quaternion must be normalized after each time integration step.

Symplectic Euler is a first-order integrator, and is used only for the prediction step of the algorithm. In Position Based Dynamics (PBD), constraint forces are integrated implicitly as described in Section 4.2.4.

3.3. Constraints

Constraints are kinematic restrictions in the form of equations and inequalities that constrain the relative motion of bodies. Equality and inequality constraints are referred to as *bilateral* and *unilateral* constraints, respectively. Generally, constraints are functions of position and orientation variables, linear and angular velocities, and their derivatives to any order. However, position-based simulation methods only consider constraints that depend on positions and in the case of rigid bodies on orientations. Hence, a bilateral constraint is defined by a function

$$C(\mathbf{x}_{i_1},\mathbf{q}_{i_1},\ldots,\mathbf{x}_{i_{n_j}},\mathbf{q}_{i_{n_j}}) = 0$$

and a unilateral constraint by

$$C(\mathbf{x}_{i_1},\mathbf{q}_{i_1},\ldots,\mathbf{x}_{i_{n_j}},\mathbf{q}_{i_{n_j}}) \geq 0,$$

where $\{i_1,\ldots i_{n_j}\}$ is a set of body indices and n_j is the cardinality of the constraint. Typically, the constraints used in PBD only depend on positions and time but not on velocities. Such constraints are called *holonomic*.

Since constraints are kinematic restrictions, they also affect the dynamics. Classical methods determine forces to simulate a dynamic system with constraints. This is done, e.g. by defining a potential energy $E=\frac{k}{2}C^2$ and deriving the forces as $\mathbf{f}=-\nabla E$ (soft constraints) or via Lagrange multipliers derived from constrained dynamics (hard constraints) [Wit97]. In contrast to that position-based approaches modify the positions and orientations of the bodies directly in order to fulfill all constraints.

© The Eurographics Association 2015.

4. The Core Of Position Based Dynamics

In this section we present Position-Based Dynamics (PBD), an approach which omits the velocity and acceleration layer and immediately works on the positions [MHHR07]. We will first describe the basic idea and the simulation algorithm of PBD. Then we will focus specifically on how to solve the system of constraints that describe the object to be simulated.

In the following the position-based approach is introduced first for particle systems. An extension to handle rigid bodies is presented in Section 5.9.

4.1. The Algorithm

The objects to be simulated are represented by a set of N particles and a set of M constraints. For each constraint we introduce a stiffness parameter k which defines the strength of the constraint in a range from zero to one. This gives a user more control over the elasticity of a body.

4.1.1. Time Integration

Algorithm 1 Position-based dynamics

1: **for all** vertices i **do**
2: initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
3: **end for**
4: **loop**
5: **for all** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
6: **for all** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
7: **for all** vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
8: **loop** solverIteration **times**
9: projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
10: **end loop**
11: **for all** vertices i **do**
12: $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
13: $\mathbf{x}_i \leftarrow \mathbf{p}_i$
14: **end for**
15: velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
16: **end loop**

Given this data and a time step Δt, the simulation proceeds as described by Algorithm 1. Since the algorithm simulates a system which is second order in time, both the positions and the velocities of the particles need to be specified in (1)-(3) before the simulation loop starts. Lines (5)-(6) perform a simple symplectic Euler integration step on the velocities and the positions. The new locations $\mathbf{p}_i$ are not assigned to the positions directly but are only used as predictions. Non-permanent external constraints such as collision constraints are generated at the beginning of each time step from scratch in line (7). Here the original and the predicted positions are used in order to perform continuous collision detection. The solver (8)-(10) then iteratively corrects the predicted positions such that they satisfy the M_{coll} external as well as the M internal constraints. Finally, the corrected positions $\mathbf{p}_i$ are used to update the positions *and* the velocities. It is essential here to update the velocities along with the positions. If this is not done, the simulation does not produce the correct behavior of a second order system. As you can see, the integration scheme used here is very similar to the Verlet method. It is also closely related to Jos Stam's *Nucleus* solver [Sta09] which also uses a set of contraints to describe the objects to be simulated. The main difference is that *Nucleus* solves the constraints for velocities, not positions.

4.1.2. Damping

The quality of dynamic simulations can generally be improved by the incorporation of an appropriate damping scheme. As a positive effect, damping can improve the stability by reducing temporal oscillations of the point positions of an object. This enables the use of larger time steps which increases the performance of a dynamic simulation. On the other hand, damping changes the dynamic motion of the simulated objects. The resulting effects can be either desired, e.g. reduced oscillations of a deformable solid, or disturbing, e.g. changes of the linear or angular momentum of the entire object.

Generally, a damping term $\mathbf{C}\dot{\mathbf{X}}$ can be incorporated into the motion equation of an object where $\dot{\mathbf{X}}$ denotes the vector of all first time derivatives of positions. If the user-defined matrix $\mathbf{C}$ is diagonal, absolute velocities of the points are damped, which sometimes is referred to as point damping. If appropriately computed, such point damping forces result in an improved numerical stability by reducing the acceleration of a point. Such characteristics are desired in some settings, e.g. in the context of friction. In the general case, however, the overall slow-down of an object, caused by point damping forces, is not desired. Point damping forces are, e.g., used in [TF88] or in [PB88], where point damping is used for dynamic simulations with geometric constraints such as point-to-nail.

In order to preserve linear and angular momentum of deformable objects, symmetric damping forces, usually referred to as spring damping forces, can be used. Such forces can be represented by non-diagonal entries in the matrix $\mathbf{C}$. Damping forces are, e.g., described by Baraff and Witkin [BW98] or Nealen et al. [NMK*06]. These forces can also be applied to position-based methods. However, as the approaches of Baraff and Witkin and Nealen et al. rely on topological information of the object geometry, they cannot be applied to meshless techniques such as shape matching.

Point and spring damping can be used to reduce current velocities or relative velocities. However, it is generally more appropriate to consider predicted velocities or relative velocities for the next time step.

An interesting damping alternative has been presented in [SGT09]. Here, the idea of symmetric, momentum-conserving forces is extended to meshless representations.

© The Eurographics Association 2015.

Global symmetric damping forces are computed with respect to the center of mass of an object. While such forces conserve the linear momentum, the preservation of the angular momentum is guaranteed by force projection onto relative positions or by torque elimination using Linear Programming. The approach presented in [SGT09] iteratively computes damping forces. The paper, however, also shows the convergence of the iterative process and how the solution can be computed directly without performing iterations. Therefore, the approach is an efficient alternative to compute damping forces for arbitrary position-based deformation models with or without connectivity information. The approach can be used to damp oscillations globally or locally for user-defined clusters.

4.2. Solver

4.2.1. The System to be Solved

The goal of the solver step (8)-(10) in Algorithm 1 is to correct the predicted positions of the particles such that they satisfy all constraints. In what follows and in contrast to Algorithm 1, we will use the symbol $\mathbf{x}$ for the positions of the particles the solver works on which is a more common symbol for positions. In Algorithm 1 we have a larger context and used the symbol $\mathbf{p}$ to distinguish the predicted positions from the positions of the previous time step.

The problem that needs to be solved comprises of a set of M equations for the $3N$ unknown position components, where M is now the total number of constraints. This system does not need to be symmetric. If $M > 3N$ ($M < 3N$), the system is over-determined (under-determined). In addition to the asymmetry, the equations are in general non-linear. The function of a simple distance constraint $C(\mathbf{x}_1,\mathbf{x}_2) = |\mathbf{x}_1 - \mathbf{x}_2|^2 - d^2$ yields a non-linear equation. What complicates things even further is the fact that collisions produce inequalities rather than equalities. Solving a non-symmetric, non-linear system with equalities and inequalities is a tough problem.

Let $\mathbf{x}$ be the concatenation $[\mathbf{x}_1^T,\ldots,\mathbf{x}_N^T]^T$ and let all the constraint functions C_j take the concatenated vector $\mathbf{x}$ as input while only using the subset of coordinates they are defined for. We can now write the system to be solved as

$$\begin{aligned} C_1(\mathbf{x}) &\succ 0 \\ &\ldots \\ C_M(\mathbf{x}) &\succ 0, \end{aligned}$$

where the symbol $\succ$ denotes either $=$ or $\geq$. Newton-Raphson iteration is a method to solve non-linear symmetric systems with equalities only. The process starts with a first guess of a solution. Each constraint function is then linearized in the neighborhood of the current solution using

$$C(\mathbf{x}+\Delta\mathbf{x}) = C(\mathbf{x}) + \nabla C(\mathbf{x})\cdot\Delta\mathbf{x} + O(|\Delta\mathbf{x}|^2) = 0.$$

This yields a *linear* system for the global correction vector $\Delta\mathbf{x}$

$$\begin{aligned} \nabla C_1(\mathbf{x})\cdot\Delta\mathbf{x} &= -C_1(\mathbf{x}) \\ &\ldots \\ \nabla C_M(\mathbf{x})\cdot\Delta\mathbf{x} &= -C_M(\mathbf{x}), \end{aligned}$$

where $\nabla C_j(\mathbf{x})$ is the $1\times N$ dimensional vector containing the derivatives of the function C_j w.r.t. all its parameters, i.e. the N components of $\mathbf{x}$. It is also the j-th row of the linear system. Both, the rows $\nabla C_j(\mathbf{x})$ and the right hand side scalars $-C_j(\mathbf{x})$ are constant because they are *evaluated* at the location $\mathbf{x}$ before the system is solved. When $M = 3N$ and only equalities are present, the system can be solved by any linear solver, e.g. a preconditioned conjugate gradient method. Once it is solved for $\Delta\mathbf{x}$ the current solution is updated as $\mathbf{x} \leftarrow \mathbf{x}+\Delta\mathbf{x}$. A new linear system is generated by evaluating $\nabla C_j(\mathbf{x})$ and $-C_j(\mathbf{x})$ at the new location after which the process repeats.

If $M \neq 3N$ the resulting matrix of the linear system is non-symmetric and not invertible. Goldenthal et al. [GHF*07] solve this problem by using the pseudo-inverse of the system matrix which yields the best solution in the least-squares sense. Still, handling inequalities is not possible directly.

4.2.2. The Non-Linear Gauss-Seidel Solver

In the PBD approach, non-linear Gauss-Seidel is used. It solves each constraint equation separately. Each constraint yields a single scalar equation $C(\mathbf{x}) \succ 0$ for all the particle positions associated with it. The subsystem is therefore highly under-determined. PBD solves this problem as follows. Again, given $\mathbf{x}$ we want to find a correction $\Delta\mathbf{x}$ such that $C(\mathbf{x}+\Delta\mathbf{x}) \succ 0$. It is important to notice that PBD also linearizes the constraint function but individually for each constraint. The constraint equation is approximated by

$$C(\mathbf{x}+\Delta\mathbf{x}) \approx C(\mathbf{x}) + \nabla C(\mathbf{x})\cdot\Delta\mathbf{x} \succ 0. \tag{5}$$

The problem of the system being under-determined is solved by restricting $\Delta\mathbf{x}$ to be in the direction of ∇C which is also a requirement for linear and angular momentum conservation. This means that only one scalar λ - a Lagrange multiplier - has to be found such that the correction

$$\Delta\mathbf{x} = \lambda\mathbf{M}^{-1}\nabla C(\mathbf{x}) \tag{6}$$

solves Equation (5), where $\mathbf{M} = diag(m_1,m_2,\ldots,m_N)$. This yields the following formula for the correction vector of a single particle i

$$\Delta\mathbf{x}_i = -\lambda\, w_i \nabla_{\mathbf{x}_i} C(\mathbf{x}), \tag{7}$$

where

$$\lambda = \frac{C(\mathbf{x})}{\sum_j w_j |\nabla_{\mathbf{x}_j} C(\mathbf{x})|^2} \tag{8}$$

© The Eurographics Association 2015.

and $w_i = 1/m_i$. Formulated for the concatenated vector $\mathbf{x}$ of all positions we get

$$\lambda = \frac{C(\mathbf{x})}{\nabla C(\mathbf{x})^T \mathbf{M}^{-1} \nabla C(\mathbf{x})}. \quad (9)$$

As mentioned above, the solver linearizes the constraint functions. However, in contrast to the Newton-Raphson method, the linearization happens individually *per constraint*. It is important to note that linearization does not affect the projection of an individual distance constraint. This is because despite being non-linear globally, a distance constraint is linear along the constraint gradient which happens to be the search direction. This is true for other constraints as well like the tetrahedral volume constraint we will discuss in Section 5.5.1. Constraints of this type can be solved in a single step. Because the positions are immediately updated after a constraint is processed, these updates will influence the linearization of the next constraint because the linearization depends on the actual positions. Asymmetry does not pose a problem because each constraint produces one scalar equation for one unknown Lagrange multiplier λ. Inequalities are handled trivially by first checking whether $C(\mathbf{x}) \geq 0$. If this is the case, the constraint is simply skipped.

The fact that each constraint is linearized individually before its projection makes the solver more stable than a global approach in which the linearizations are kept fixed for the entire global solve of a Newton iteration.

We have not considered the stiffness k of the constraint so far. There are several ways to incorporate it. The simplest variant is to multiply the corrections $\Delta\mathbf{x}$ by $k \in [0 \dots 1]$. However, for multiple iteration loops of the solver, the effect of k is non-linear. The remaining error for a single distance constraint after n_s solver iterations is $\Delta\mathbf{x}(1-k)^{n_s}$. To get a linear relationship we multiply the corrections not by k directly but by $k' = 1-(1-k)^{1/n_s}$. With this transformation the error becomes $\Delta\mathbf{x}(1-k')^{n_s} = \Delta\mathbf{x}(1-k)$ and, thus, becomes linearly dependent on k and independent of n_s as desired. However, the resulting material stiffness is still dependent on the time step of the simulation. Real time environments typically use fixed time steps in which case this dependency is not problematic.

4.2.3. Hierarchical Solver

The Gauss-Seidel method is stable and easy to implement but it typically converges significantly slower than global solvers. The main reason is that error corrections are propagated only locally from constraint to constraint. Therefore, the Gauss-Seidel method is called a smoother because it evens out the high frequency errors much faster than low frequency errors.

A popular method to increase the convergence rate of the Gauss-Seidel method is to create a hierarchy of meshes in which the coarse meshes make sure that error corrections propagate fast across the domain. A smoother works on all meshes of the hierarchy one by one while the error corrections are carried over across meshes of different resolutions typically in multiple cycles from fine to coarse levels and back. This technique is called the multi-grid method [GW06]. Transferring corrections from coarse to fine meshes and from fine to coarse meshes is called *prolongation* and *restriction*, respectively. Multi-grid methods differ in the way the hierarchy is created, in how the restriction and prolongation operators are defined and in what order the meshes are processed.

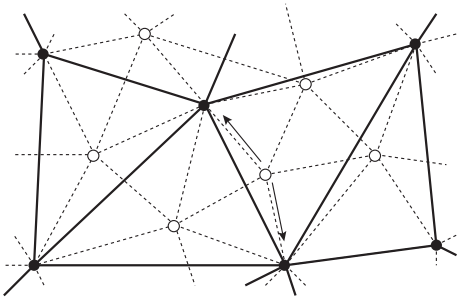

Figure 2: *The construction of a mesh hierarchy: A fine level l is composed of all the particles shown and the dashed constraints. The next coarser level $l+1$ contains the proper subset of black particles and the solid constraints. Each fine white particle needs to be connected to at least k (=2) black particles – its parents – shown by the arrows.*

In [Mül08], Müller et al. used this technique and introduced Hierarchical Position Based Dynamics (HPBD). They define the original simulation mesh to be the finest mesh of the hierarchy and create coarser meshes by only keeping a subset of the particles of the previous mesh. The hierarchy is traversed only once from the coarsest to the finest level. Therefore, they only need to define a prolongation operator. By making sure that each particle of a given level is connected to at least two particles in the next coarser level (see Figure 2), prolongation amounts to interpolating the information from adjacent particles of the coarser level. They also propose a method to create distance constraints on the coarse meshes based on the constraints of the original mesh. It is important to note that these coarse constraints must be unilateral, i.e. only act if the current distance is larger than the rest distance otherwise they would prevent bending and folding.

In Section 5.6 we describe a much simpler and effective way to speed up error propagation for the specific but quite common case of cloth that is attached to a kinematic or static object.

4.2.4. Connection to Implicit Methods

As Liu et al. [LBOK13] pointed out, PBD is closely related to implicit backward Euler integration schemes. We can see

© The Eurographics Association 2015.

this by considering backward Euler as a constrained minimization over positions. Starting with the traditional implicit Euler time discretization of the equations of motion:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \tag{10}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\left(\mathbf{F}_{ext} + k\nabla \mathbf{C}^{n+1}\right) \tag{11}$$

where $\mathbf{C}$ is the vector of constraint potentials, and k is the stiffness, we can eliminate velocity to give:

$$\mathbf{M}(\mathbf{x}^{n+1} - 2\mathbf{x}^n + \mathbf{x}^{n-1} - \Delta t^2 \mathbf{M}^{-1}\mathbf{F}_{ext}) = \Delta t^2 k \nabla \mathbf{C}^{n+1}. \tag{12}$$

Equation (12) can be seen as the first order optimality condition for the following minimization:

$$\min_{\mathbf{x}} \quad \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) - \Delta t^2 k \mathbf{C}^{n+1} \tag{13}$$

where $\tilde{\mathbf{x}}$ is the predicted position, given by:

$$\tilde{\mathbf{x}} = 2\mathbf{x}^n - \mathbf{x}^{n-1} + \Delta t^2 \mathbf{M}^{-1}\mathbf{F}_{ext} \tag{14}$$

$$= \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{M}^{-1}\mathbf{F}_{ext}. \tag{15}$$

Taking the limit as $k \to \infty$ we obtain the following constrained minimization:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}})^T \mathbf{M}(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}) \\ \text{s.t.} \quad & C_i(\mathbf{x}^{n+1}) = 0, \quad i = 1, \ldots, n. \end{aligned} \tag{16}$$

We can interpret this minimization problem as finding the closest point on the constraint manifold to the predicted position (in a mass-weighted measure). PBD approximately solves this minimization using a variant of the fast projection algorithm of Goldenthal et al. [GHF*07], which first takes a prediction step and then iteratively projects particles onto the constraint manifold. PBD follows this approach, but differs in the method used to solve the projection step. In contrast to [GHF*07] PBD does not necessarily linearize and solve the system as a whole in each Newton step. Instead, it linearizes one constraint at a time in a Gauss-Seidel fashion as discussed in Section 4.2.1. This helps to make PBD robust in the presence of large constraint non-linearities.

Projective Dynamics [BML*14] presents a modification to PBD that allows treating constraints in an implicit manner that does not depend on the constraints being infinitely stiff. This is accomplished by adding additional constraints that act to pull the solution back towards the predicted (inertial) position.

4.2.5. Second Order Methods

Now we have established the connection to backward Euler, we can apply higher order integration schemes to PBD. Following the derivation in [EB08] we will adapt BDF2, a second order accurate multistep method. First, we write the second order accurate BDF2 update equations:

$$\mathbf{x}^{n+1} = \frac{4}{3}\mathbf{x}^n - \frac{1}{3}\mathbf{x}^{n-1} + \frac{2}{3}\Delta t \mathbf{v}^{n+1} \tag{17}$$

$$\mathbf{v}^{n+1} = \frac{4}{3}\mathbf{v}^n - \frac{1}{3}\mathbf{v}^{n-1} + \frac{2}{3}\Delta t \mathbf{M}^{-1}\left(\mathbf{F}_{ext} + k\nabla \mathbf{C}^{n+1}\right). \tag{18}$$

Eliminating velocity and re-arranging gives

$$\mathbf{M}\left(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}\right) = \frac{4}{9}\Delta t^2 k \nabla \mathbf{C}^{n+1}, \tag{19}$$

where the inertial position $\tilde{\mathbf{x}}$ is given by

$$\tilde{\mathbf{x}} = \frac{4}{3}\mathbf{x}^n - \frac{1}{3}\mathbf{x}^{n-1} + \frac{8}{9}\Delta t \mathbf{v}^n - \frac{2}{9}\Delta t \mathbf{v}^{n-1} + \frac{4}{9}\Delta t^2 \mathbf{M}^{-1}\mathbf{F}_{ext}. \tag{20}$$

Equation (19) can again be considered as the optimality condition for a minimization of the same form as (16). Once the constraints have been solved, the updated velocity is obtained according to (17),

$$\mathbf{v}^{n+1} = \frac{1}{\Delta t}\left[\frac{3}{2}\mathbf{x}^{n+1} - 2\mathbf{x}^n + \frac{1}{2}\mathbf{x}^{n-1}\right]. \tag{21}$$

To evaluate this more accurate scheme we need only store the previous position and velocity, and perform some additional basic arithmetic during the prediction and velocity update steps, while the rest of the PBD algorithm is unchanged. The benefits of this simple modification are an order of magnitude less numerical damping, and faster convergence for the constraint projection. This can be understood by considering the algorithm as using previous time-step information in order to generate predicted positions that stay closer to the constraint manifold, making projection faster.

© The Eurographics Association 2015.

5. Specific Constraints

In the following we will introduce different constraints that can be used to simulate a variety of materials such as articulated rigid bodies, soft bodies, cloth or even fluids with PBD. For better readability we define $\mathbf{x}_{i,j} = \mathbf{x}_i - \mathbf{x}_j$.

5.1. Stretching

To give an example, let us consider the distance constraint function $C(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_{1,2}| - d$. The derivatives with respect to the points are $\nabla_{\mathbf{x}_1} C(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{n}$ and $\nabla_{\mathbf{x}_2} C(\mathbf{x}_1, \mathbf{x}_2) = -\mathbf{n}$ with $\mathbf{n} = \frac{\mathbf{x}_{1,2}}{|\mathbf{x}_{1,2}|}$. The scaling factor λ is, thus, $\lambda = \frac{|\mathbf{x}_{1,2}| - d}{1+1}$ and the final corrections

$$\Delta\mathbf{x}_1 = -\frac{w_1}{w_1 + w_2}(|\mathbf{x}_{1,2}| - d)\,\mathbf{n}$$
$$\Delta\mathbf{x}_2 = +\frac{w_2}{w_1 + w_2}(|\mathbf{x}_{1,2}| - d)\,\mathbf{n},$$

which are the formulas proposed in [Jak01] for the projection of distance constraints (see Figure 3). They can be derived as a special case of the general constraint projection method.

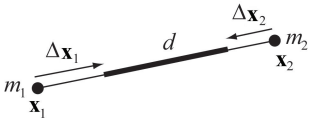

Figure 3: *Projection of the constraint $C(\mathbf{x}_1, \mathbf{x}_2) = |\mathbf{x}_{1,2}| - d$. The corrections $\Delta\mathbf{x}_i$ are weighted according to the inverse masses $w_i = 1/m_i$.*

5.2. Bending

In cloth simulation it is important to simulate bending in addition to stretching resistance. To this end, for each pair of adjacent triangles $(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2)$ and $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4)$ a bilateral bending constraint is added with constraint function

$$C_{bend}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \operatorname{acos}\left(\frac{\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}}{|\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}|} \cdot \frac{\mathbf{x}_{2,1} \times \mathbf{x}_{4,1}}{|\mathbf{x}_{2,1} \times \mathbf{x}_{4,1}|}\right) - \varphi_0$$

and stiffness k_{bend}. The scalar φ_0 is the initial dihedral angle between the two triangles and k_{bend} is a global user parameter defining the bending stiffness of the cloth (see Figure 4). The advantage of this bending term over adding a distance constraint between points $\mathbf{x}_3$ and $\mathbf{x}_4$ is that it is *independent of stretching*. This is because the term is independent of edge lengths. In Figure 9 we show how bending and stretching resistance can be tuned independently.

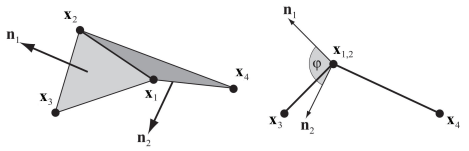

Figure 4: *For bending resistance, the constraint function $C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \varphi_0$ is used. The actual dihedral angle φ is measured as the angle between the normals of the two triangles.*

5.3. Isometric Bending

A bending constraint for inextensible surfaces was introduced in [BKCW14]. The definition of this constraint is based on the discrete isometric bending model of Bergou et al. [BWH*06], which can be applied if a surface deforms isometrically, i.e., if the edge lengths remain invariant. Since many textiles cannot be stretched significantly, this method is an appropriate choice in garment simulation.

For each interior edge e_i a stencil s is defined which consists of the two triangles adjacent to e_i. The vector $\mathbf{x}_s = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)^T$ contains the four vertices of the stencil and the vector $\mathbf{e}_s = [\mathbf{x}_0\mathbf{x}_1, \mathbf{x}_1\mathbf{x}_2, \mathbf{x}_2\mathbf{x}_0, \mathbf{x}_0\mathbf{x}_3, \mathbf{x}_3\mathbf{x}_1]$ contains the five stencil edges starting with the common edge (see Figure 5).

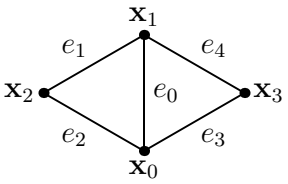

Figure 5: *The isometric bending constraint is defined using the the stencil of an interior edge e_0.*

Using the isometric bending model the local Hessian bending energy of a stencil is determined by

$$\mathbf{Q} = \frac{3}{A_0 + A_1}\mathbf{K}^T\mathbf{K},$$

where A_0 and A_1 are the areas of the adjacent triangles and $\mathbf{K}$ is the vector

$$\mathbf{K} = (c_{01} + c_{04},\ c_{02} + c_{03},\ -c_{01} - c_{02},\ -c_{03} - c_{04}),$$

where $c_{jk} = \cot\angle e_j, e_k$. The matrix $\mathbf{Q} \in \mathbb{R}^{4\times4}$ is constant and can be precomputed with the initial configuration of the stencil. The local Hessian bending energy can be used to define a bending constraint as

$$C_{bend}(\mathbf{x}_s) = \frac{1}{2}\sum_{i,j}\mathbf{Q}_{i,j}\,\mathbf{x}_i^T\mathbf{x}_j.$$

Since the Hessian bending energy is constant, the gradients

© The Eurographics Association 2015.

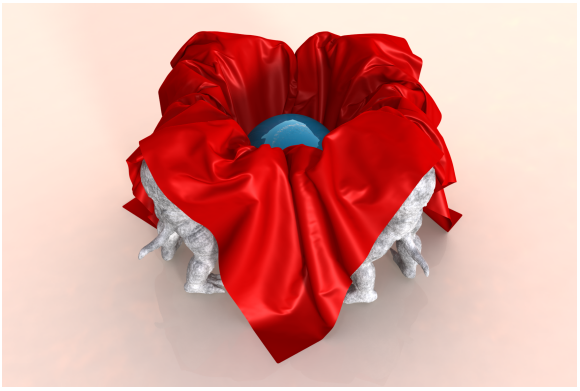

Figure 6: *A heavy sphere is pushing down a piece of cloth that is thrown over four statues. Realistic wrinkles evolve due to the isometric bending constraint.*

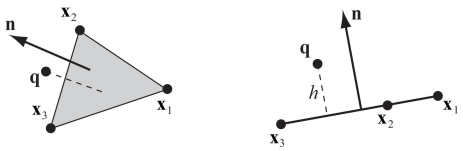

Figure 7: *Constraint function* $C(\mathbf{q},\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = (\mathbf{q}-\mathbf{x}_1)\cdot\mathbf{n}-h$ *makes sure that* $\mathbf{q}$ *stays above the triangle* $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$ *by the cloth thickness* h.

are determined by

$$\frac{\partial C_{bend}}{\partial \mathbf{x}_i} = \sum_j \mathbf{Q}_{i,j}\mathbf{x}_j.$$

Figure 6 shows a cloth simulation with the introduced bending constraint.

5.4. Collisions

5.4.1. Triangle Collisions

Self collisions within cloth can be handled by additional unilateral constraints. For vertex $\mathbf{q}$ moving through a triangle $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, the constraint function reads

$$C(\mathbf{q},\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = (\mathbf{q}-\mathbf{x}_1)\cdot\frac{\mathbf{x}_{2,1}\times\mathbf{x}_{3,1}}{|\mathbf{x}_{2,1}\times\mathbf{x}_{3,1}|}-h,$$

where h is the cloth thickness (see Figure 7). If the vertex enters from below with respect to the triangle normal, the constraint function has to be

$$C(\mathbf{q},\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = (\mathbf{q}-\mathbf{x}_1)\cdot\frac{\mathbf{x}_{3,1}\times\mathbf{x}_{2,1}}{|\mathbf{x}_{3,1}\times\mathbf{x}_{2,1}|}-h.$$

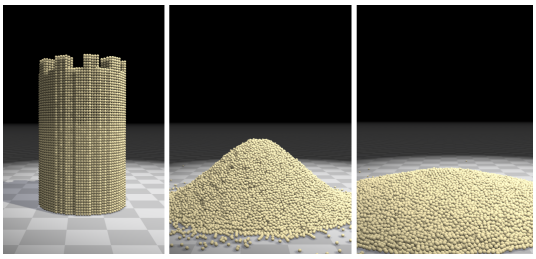

Figure 8: *A sand castle before collapse (left). After 300 frames the position-based friction model maintains a steep pile (middle), while the velocity level friction model has almost completely collapsed (right).*

5.4.2. Environment Collisions

Collisions between particles and kinematic shapes, represented as e.g.: triangle or convex meshes, can be handled by first detecting a set of candidate contact planes for each particle, then for each contact plane normal $\mathbf{n}$, a non-penetration constraint is introduced into the system of the form

$$C(\mathbf{x}) = \mathbf{n}^T\mathbf{x} - d_{rest} = 0, \tag{22}$$

where d_{rest} is the distance the particle should maintain from the geometry at rest.

5.4.3. Particle Collisions

Collisions between particles can be handled in a similar manner to the environment by linearizing and introducing a contact plane, however, it is often more robust to maintain the non-linear nature of the constraint, in the form:

$$C(\mathbf{x}_i,\mathbf{x}_j) = |\mathbf{x}_{ij}| - (r_i + r_j) \geq 0, \tag{23}$$

where r_i and r_j are the radii of the two particles. This constraint can be used to model granular-like materials as shown in [MMCK14].

5.4.4. Friction

Müller et al. [MHHR07] handled friction by introducing damping forces applied after the constraint solve. This approach is suitable for weak frictional effects, but cannot model static friction, because the positional constraints can freely violate the frictional forces. To model situations where friction is strong relative to the constraints (see Figure 8), Macklin et al. [MMCK14] include frictional effects as part of the position level constraint solve.

Once interpenetration between particles has been resolved, a frictional position delta is calculated based on the relative tangential displacement of the particles during this time-step. The relative displacement is given by

$$\Delta\mathbf{x}_\perp = \left[(\mathbf{x}_i^* - \mathbf{x}_i) - (\mathbf{x}_j^* - \mathbf{x}_j)\right] \perp \mathbf{n}, \tag{24}$$

where $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are the current candidate positions for the

© The Eurographics Association 2015.

colliding particles, including any previously applied constraint deltas, $\mathbf{x}_i$ and $\mathbf{x}_j$ are the positions of the particles at the start of the time-step, and $\mathbf{n} = \mathbf{x}^*_{ij}/|\mathbf{x}^*_{ij}|$ is the contact normal. The frictional position delta for particle i is then computed as

$$\Delta\mathbf{x}_i = \frac{w_i}{w_i + w_j} \begin{cases} \Delta\mathbf{x}_\perp, & |\Delta\mathbf{x}_\perp| < \mu_s d \\ \Delta\mathbf{x}_\perp \cdot \min(\frac{\mu_k d}{|\Delta\mathbf{x}_\perp|}, 1), & otherwise \end{cases} \quad (25)$$

where d is the penetration depth, and μ_k, μ_s are the coefficients of kinetic and static friction, respectively. The first case in Eq. (25) models static friction by removing all tangential movement when the particle's relative velocity is below the traction threshold. The second case models kinetic Coulomb friction, limiting the frictional position delta based on the penetration depth of the particle. The position change on particle j is given by

$$\Delta\mathbf{x}_j = -\frac{w_j}{w_i + w_j}\Delta\mathbf{x}_i. \quad (26)$$

Friction with kinematic shapes is handled using the same method, with the shape treated as having infinite mass and the contact plane defined by its geometry.

5.5. Volume Conservation

The conservation of volume plays an important role in the dynamic simulation of deformable bodies [HJCW06, ISF07, DBB09]. Since most soft biological tissues are incompressible, this is an essential extension in the field of medical simulation. However, it is also used in the field of shape modeling [vFTS06] since volume conserving deformations appear more realistic.

5.5.1. Tetrahedral Meshes

For tetrahedral meshes it is useful to have a constraint that conserves the volume of single tetrahedron. Such a constraint has the form

$$C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \frac{1}{6}\left(\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}\right) \cdot \mathbf{x}_{4,1} - V_0,$$

where $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$ are the four corners of the tetrahedron and V_0 is its rest volume. In a similar way, the area of a triangle can be kept constant by introducing

$$C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \frac{1}{2}\left|\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}\right| - A_0.$$

5.5.2. Cloth Balloons

For closed triangle meshes, overpressure inside the mesh as shown in Figure 10 can easily be modeled with an *equality* constraint concerning all N vertices of the mesh:

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \left(\sum_{i=1}^{n_{\text{triangles}}} (\mathbf{x}_{t_1^i} \times \mathbf{x}_{t_2^i}) \cdot \mathbf{x}_{t_3^i}\right) - k_{\text{pressure}} V_0.$$

Here t_1^i, t_2^i and t_3^i are the three indices of the vertices belonging to triangle i. The sum computes the actual volume of the closed mesh. It is compared against the original volume V_0 times the overpressure factor k_{pressure}. This constraint function yields the gradients

$$\nabla_{\mathbf{x}_i} C = \sum_{j:t_1^j = i} (\mathbf{x}_{t_2^j} \times \mathbf{x}_{t_3^j}) + \sum_{j:t_2^j = i} (\mathbf{x}_{t_3^j} \times \mathbf{x}_{t_1^j}) + \sum_{j:t_3^j = i} (\mathbf{x}_{t_1^j} \times \mathbf{x}_{t_2^j}).$$

These gradients have to be scaled by the scaling factor given in Equation (8) and weighted by the masses according to Equation (7) to get the final projection offsets $\Delta\mathbf{x}_i$.

Figure 10: *Simulation of overpressure inside a character.*

5.5.3. Surface Meshes

In the following we introduce the position-based approach for volume conservation of Diziol et al. [DBB11]. This method considers only the surface of a simulated object and does not require interior particles which reduces the computational effort. The volume V of a volumetric 3D shape $\mathcal{V}$ can be determined by using the divergence theorem as proposed in [Mir96] and [HJCW06]:

$$\iiint_{\mathcal{V}} \nabla \cdot \mathbf{x}\, d\mathbf{x} = \iint_{\partial\mathcal{V}} \mathbf{x}^{\mathrm{T}}\mathbf{n}\, d\mathbf{x} = 3V, \quad (27)$$

where $\partial\mathcal{V}$ is the boundary of the shape and $\mathbf{n}$ is the surface normal. If the boundary is given as triangle mesh, the integral can be written as sum over all triangles i:

$$V(\mathbf{X}) := \frac{1}{3}\iint_{\partial\mathcal{V}} \mathbf{x}^{\mathrm{T}}\mathbf{n}\, d\mathbf{x} = \frac{1}{9}\sum_i A_i(\mathbf{x}_{i_1} + \mathbf{x}_{i_2} + \mathbf{x}_{i_3})^{\mathrm{T}}\mathbf{n}_i, \quad (28)$$

where A_i is the area and i_1, i_2 and i_3 are the vertex indices of the i-th triangle. Now we can define a volume constraint $C := V(\mathbf{X}) - V_0 = 0$ and compute a corresponding position correction (see Section 4):

$$\Delta\mathbf{x}_i^V = -\frac{w_i C(\mathbf{X})}{\sum_j w_j \|\nabla_{\mathbf{x}_j} C(\mathbf{X})\|^2} \nabla_{\mathbf{x}_i} C(\mathbf{X}). \quad (29)$$

The weights w_i are used to realize a local volume conservation (see below). The gradient can be approximated by

$$\nabla C(\mathbf{X}) \approx \frac{1}{3}[\bar{\mathbf{n}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{n}}_n^{\mathrm{T}}]^{\mathrm{T}},$$

where $\bar{\mathbf{n}}_i = \sum A_j \mathbf{n}_j$ is the sum of the area weighted normals of all triangles which contain particle i.

The weights in Equation (29) are chosen as follows:

$$w_i = (1-\alpha)w_i^l + \alpha w_i^g, \quad w_i^l = \frac{\|\Delta\mathbf{x}_i\|}{\sum_j \|\Delta\mathbf{x}_j\|}, \quad w_i^g = \frac{1}{n},$$

© The Eurographics Association 2015.

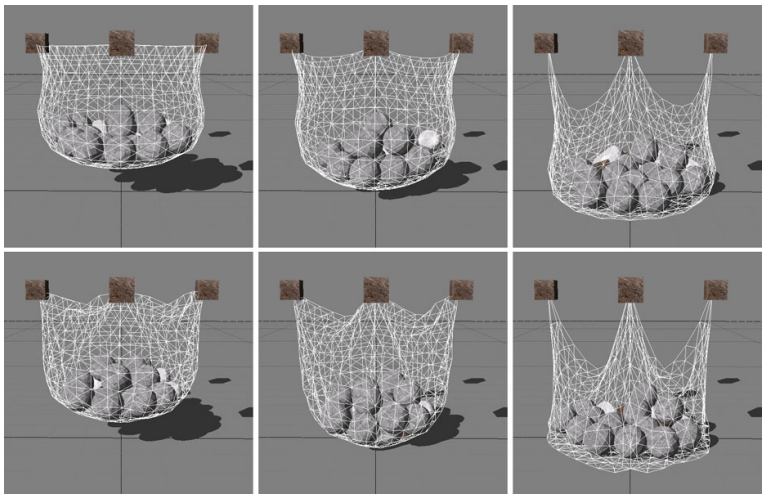

Figure 9: *The image shows a mesh that is simulated using stretching and bending constraints. The top row shows* $(k_{stretching}, k_{bending}) = (1,1)$, $(\frac{1}{2},1)$ *and* $(\frac{1}{100},1)$. *The bottom row shows* $(k_{stretching}, k_{bending}) = (1,0)$, $(\frac{1}{2},0)$ *and* $(\frac{1}{100},0)$.

where w_i^l and w_i^g are the weights for local and global volume conservation, respectively, and the user-defined value $\alpha \in [0,1]$ is used to blend between both. The vector $\Delta \mathbf{x}_i$ contains the total position change of the i-th. Hence, strongly deformed particles participate more in volume correction. The weight of a colliding particle is set to zero in order to ensure that a collision constraint is not violated during the position correction for the volume conservation. Finally, the weights are smoothed by a Laplacian filter.

Diziol et al. also propose another definition for the local weights w_i^l. To propagate volume changes through the object, they first determine pairs of opposing particles in a preprocessing step by intersecting the geometry with multiple rays. For each particle i one particle k on the opposite side of the volumetric body is stored. Then they choose a local weight which does not only depend on the position change $\Delta \mathbf{x}_i$ of a particle but also on the distance changes Δd_i of the corresponding particle pairs:

$$w_i^l = \frac{\beta s_i \Delta d_i + (1-\beta)\|\Delta \mathbf{x}_i\|}{\sum_j \left(\beta s_j \Delta d_j + (1-\beta)\|\Delta \mathbf{x}_j\|\right)},$$

where s_i is a user-defined stiffness parameter and $\beta \in [0,1]$ is used to define the influence of the distance changes.

Analogous to the positions correction we perform a velocity correction to fulfill the constraint $\partial C/\partial t = 0$. This leads to a divergence free velocity field.

In Figure 11 different configurations for the presented volume conservation method are compared with each other.

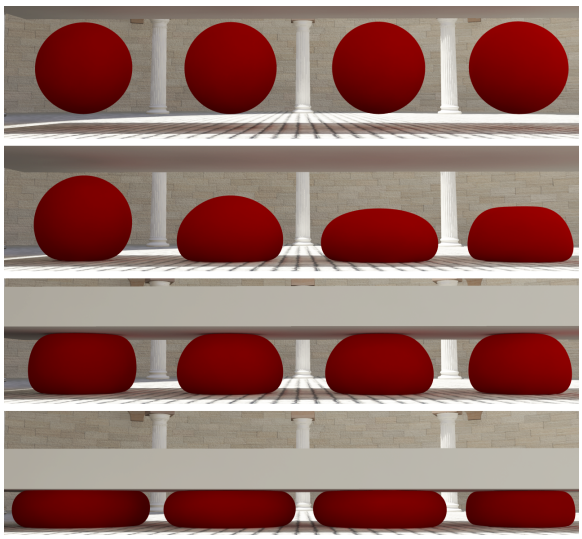

Figure 11: *Four spheres with different volume conservation squeezed by a plate. Left to right: global conservation, local conservation with distance constraints, local conservation without distance constraints and no volume conservation. The maximum volume loss was* 0.6%, 0.7%, 0.7% *and* 40% *respectively.*

5.5.4. Robust Collision Handling with Air Meshes

As Müller et al. show in [MCKM15], per-element volume constraints can also be used to robustly handle collisions. To this end, they tessellate the air between objects. Collisions can then be prevented by making sure that the air elements

© The Eurographics Association 2015.

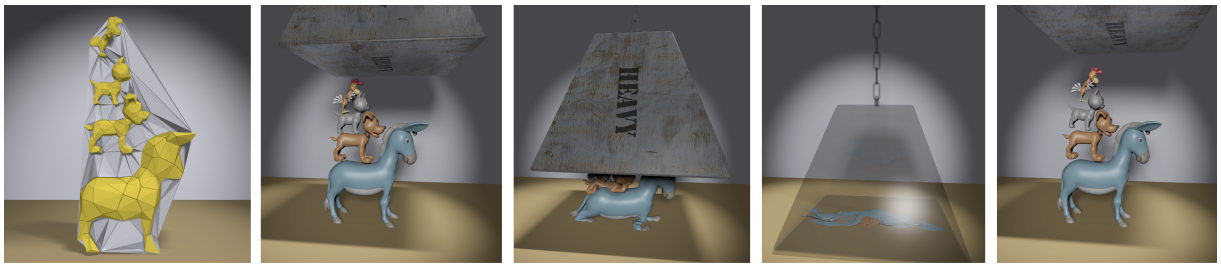

Figure 12: *With air mesh based collision handling, both the characters themselves as well as their spacial order is recovered from a completely flat state.*

Figure 13: *Smooth recovery from a severely entangled cloth state using an air mesh for collision handling.*

do not invert with the unilateral constraints

$$C_{\text{air element}}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4) = \left(\mathbf{x}_{2,1}\times\mathbf{x}_{3,1}\right)\cdot\mathbf{x}_{4,1} \geq 0 \text{ and}$$
$$C_{\text{air element}}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = \left|\mathbf{x}_{2,1}\times\mathbf{x}_{3,1}\right| \geq 0$$

in 3D and 2D, respectively. When the volume of an air element is positive, the element is passive, does not disturb the simulation and causes no computational cost. The main advantage of air meshes over existing collision handling methods is that air meshes have a memory. Even if a scene is completely flattened as shown in Figure 12, the objects pop up in the correct order when released. This is particularly useful in the simulation on complex clothing as shown in Figure 13. Air meshes not only detect entangled states easily, they also allow the smooth recovery from arbitrary entangled states which is a hard problem as the literature on this topic shows.

Müller et al. note that when large relative translations and rotations between objects occur, the air elements can lock and report collisions in a collision free state. The authors solve this problem by running a mesh optimization step. They perform edge flips in 2D and generalized edge flips in 3D whenever they improve the mesh quality. This step prevents locking - not provably but in all practical examples. In 2D, the optimization step is fast and allows the simulation of arbitrary scenarios. In 3D, mesh optimization is significantly more expensive. Fortunately, in the case of complex clothing, locking does not cause disturbing visual artifacts.

5.6. Long Range Attachments

Recently, Kim et al. [KCM12] found a surprisingly simple and robust technique they call *Long Range Attachments* (LRA) to prevent cloth from getting stretched globally with low iteration counts. Their method exploits the fact that stretching artifacts almost always appear when cloth is attached. In this case, instead of only applying attachment constraints to the subset of the vertices near the region where the cloth is attached and relying on error propagation of the solver for all other vertices, they apply unilateral attachment constraints to *all* the vertices by attaching each vertex to one or more attachment point directly. The rest lengths of these long range attachments can either be set to the Euclidean distance in the rest state or via measuring geodesic lengths along the cloth. Figure 14 demonstrates the method on a single rope attached at one end. The method allows the simulation of a piece of cloth with 90K vertices at interactive rates as shown in Figure 15.

5.7. Strands

A similar approach was recently proposed by Müller et al. [MKC12] to guarantee zero stretch in a single pass for the case of attached ropes. This approach allows the simulation of thousands of hair strands in real time (see Figure 16). Figure 17 visualizes the basic idea. Particle $\mathbf{x}_1$ is attached. To satisfy the first distance constraint, particle $\mathbf{x}_2$ is moved towards $\mathbf{x}_1$ such that their mutual distance is l_0. Particle $\mathbf{x}_3$ is then moved towards the *new* position of $\mathbf{x}_2$ and similarly along the chain until the last particle is reached. After this single pass, all the distance constraints are satisfied. This method is called *Follow The Leader (FTL)*. While LRA guarantees zero stretch of all the particles w.r.t. the attachment points, the constraint between consecutive particles can still remain overstretched. On the other hand, in contrast to LRA which is momentum conserving, FTL introduces unphysical behavior. Not projecting distance constraints symmetrically

© The Eurographics Association 2015.

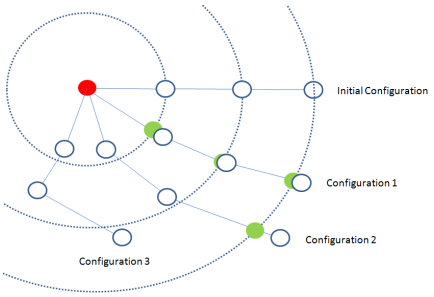

Figure 14: *The Long Range Attachments (LRA) method used to simulate an inextensible rope attached at one end. Each particle is constrained or remain inside a sphere centered at the attachment point (red) whose radius is the initial distance from the particle to the attachment. For each configuration, target positions are shown in green when particles need to be projected. Particles inside the constraint spheres are allowed to move freely.*

Figure 15: *Simulation of a piece of cloth with 90K vertices at 20fps on a GPU using LRA.*

means that a system is simulated for which each particle has infinitely more mass than its successor. To compensate for this behavior, the authors replace the PBD velocity update $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$ in Algorithm 1 by

$$\mathbf{v}_i \leftarrow \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t} + s_{\text{damping}} \frac{-\mathbf{d}_{i+1}}{\Delta t},$$

where $\mathbf{d}_{i+1}$ is the position correction applied to particle $i+1$ and $s_{\text{damping}} \in [0,1]$ a scaling factor do influence damping. While this modification of DFTL (dynamic FTL) hides the unphysical behavior of FTL, it introduces a certain amount of damping which is acceptable for the simulation of hair and fur as the author's results show.

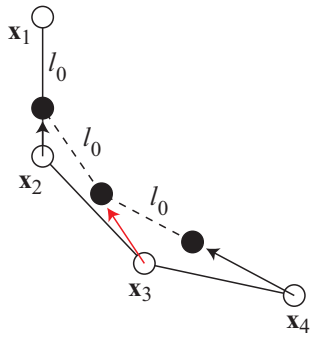

Figure 17: *Follow The Leader (FTL) projection. Starting from the attachment down, each particle is moved directly towards its predecessor such that their mutual distance constraint is satisfied.*

5.8. Continuous Materials

Recently, position-based methods based on a continuum-based formulation were presented. In the following we introduce two methods which use this formulation. The first method defines a constraint for the strain energy of a deformable solid [BKCW14] while the second one directly constrains the strain tensor [MCKM14].

5.8.1. Strain Energy Constraint

In continuum mechanics the deformation of a body is defined by the function

$$\phi(\mathbf{X}) = \mathbf{X} + \mathbf{u} = \mathbf{x},$$

which maps a point $\mathbf{X}$ in material space to its corresponding deformed location $\mathbf{x}$ in world space using a continuous displacement field $\mathbf{u}$. The Jacobian of this function $\mathbf{F} = \frac{\partial \phi(\mathbf{X})}{\partial \mathbf{X}}$, also known as deformation gradient, is used to determine the non-linear Green strain tensor

$$\varepsilon = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right), \tag{30}$$

where $\mathbf{I}$ denotes the identity matrix. Hooke's generalized law gives us the relation between stress and strain

$$\mathbf{S} = \mathbf{C}\varepsilon,$$

where $\mathbf{C}$ is the elasticity tensor which defines the elastic behavior of the material. For isotropic materials this relationship is called Saint-Venant Kirchhoff model, where $\mathbf{C}$ is defined by two independent variables, often expressed by the engineering constants Young's modulus k and Poisson ratio ν. The energy of a deformed solid is defined by integrating the scalar strain energy density field

$$\Psi_s = \frac{1}{2}\varepsilon : \mathbf{S} = \frac{1}{2}\text{tr}(\varepsilon^T\mathbf{S})$$

over the entire body Ω:

$$E_s = \int_\Omega \Psi_s d\mathbf{X}, \tag{31}$$

© The Eurographics Association 2015.

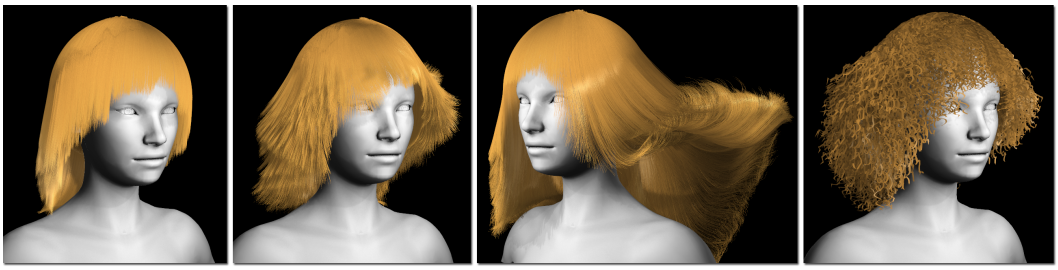

Figure 16: *Dynamic FTL allows the simulation of every hair strand in real time. From left to right: 47k hair strands simulated at 25 fps including rendering and hair-hair repulsion. Long hair composed of 1.9m particles at 8 fps. Curly hair using visualization post-processing.*

where $\mathrm{tr}(\cdot)$ is the trace of a matrix.

In order to simulate deformable solids with the position-based approach an energy constraint $C(\mathbf{x}) = E_s(\mathbf{x}) = 0$ is defined. A discretization of the solid is required to compute the energy. We use tetrahedral meshes for volumetric bodies and triangle meshes for surface models in combination with linear Lagrangian shape functions to discretize the body. For linear shape functions the deformation gradient of a tetrahedral element is determined by

$$\mathbf{F}^{\text{tet}} = \mathbf{D}_s \mathbf{D}_m^{-1},$$

where $\mathbf{D}_s$ is the deformed shape matrix and $\mathbf{D}_m$ the constant reference shape matrix defined by the vertices of the tetrahedral element

$$\mathbf{D}_s = \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_4 & \mathbf{x}_2 - \mathbf{x}_4 & \mathbf{x}_3 - \mathbf{x}_4 \end{pmatrix}$$
$$\mathbf{D}_m = \begin{pmatrix} \mathbf{X}_1 - \mathbf{X}_4 & \mathbf{X}_2 - \mathbf{X}_4 & \mathbf{X}_3 - \mathbf{X}_4 \end{pmatrix}.$$

The deformation gradient $\mathbf{F}^{\text{tri}} \in \mathbb{R}^{2\times 2}$ for a triangular element is defined analogously in the two-dimensional space of the triangle plane.

The constraint of a tetrahedral element can now be defined as

$$C(\mathbf{x}) = E_s(\mathbf{x}) = V\Psi_s(\mathbf{F}^{\text{tet}}),$$

where V is the undeformed volume of the element. Additionally, the position-based solver requires the gradients of the constraint $\nabla C_{\mathbf{x}_i} = \partial E_s / \partial \mathbf{x}_i$ which are determined by

$$\begin{bmatrix} \frac{\partial E_s}{\partial \mathbf{x}_1} & \frac{\partial E_s}{\partial \mathbf{x}_2} & \frac{\partial E_s}{\partial \mathbf{x}_3} \end{bmatrix} = V\mathbf{P}(\mathbf{F}^{\text{tet}})\mathbf{D}_m^{-T}, \quad \frac{\partial E_s}{\partial \mathbf{x}_4} = -\sum_{i=1}^{3} \frac{\partial E_s}{\partial \mathbf{x}_i},$$

where $\mathbf{P}(\mathbf{F}) = \mathbf{F}\mathbf{C}\varepsilon$ is the first Piola-Kirchhoff stress tensor.

Note that common constitutive models are not designed to handle degenerate or inverted tetrahedral elements. However, this problem can be solved by using the inversion handling of Irving et al. [ITF04].

The constraint of a triangular element is defined analogously

$$C(\mathbf{x}) = E_s(\mathbf{x}) = A\Psi(\mathbf{F}^{\text{tri}}),$$

where A is the area of the undeformed triangle. The constraint gradients of the three vertices are determined by

$$\begin{bmatrix} \frac{\partial E_s}{\partial \mathbf{x}_1} & \frac{\partial E_s}{\partial \mathbf{x}_2} \end{bmatrix} = A\mathbf{P}(\mathbf{F}^{\text{tri}})\mathbf{D}_m^{-T}, \quad \frac{\partial E_s}{\partial \mathbf{x}_3} = -\sum_{i=1}^{2} \frac{\partial E_s}{\partial \mathbf{x}_i}.$$

The proposed energy constraint formulation [BKCW14] has the advantage that it can handle complex physical effects like lateral contraction, anisotropy or elastoplasticity (see Figure 18, right). Moreover, it is not limited to the introduced Saint-Venant Kirchhoff model, also other material models like e.g. the Neo-Hookean model are supported. Finally, Bender et al. [BKCW14] demonstrated that this approach is very efficient and even faster than shape matching. Therefore, the method allows to simulate complex scenes with a high number of elements (see Figure 18, left).

5.8.2. Strain Based Dynamics

In [MCKM14] the authors propose another position-based method based on continuum mechanics which allows the control of stretch and shear deformations independent of the tessellation of the mesh. The basic idea is to force the components of Green's strain tensor ε defined in Equation (30) to zero by introducing one constraint per independent component

$$C_{\text{stretch}}(\mathbf{x}) = \mathbf{S}_{ii} - 1 \tag{32}$$
$$C_{\text{shear}}(\mathbf{x}) = \mathbf{S}_{ij} \qquad i < j, \tag{33}$$

where $\mathbf{S} = \mathbf{F}^T\mathbf{F}$ and $\mathbf{x}$ defines the positions of the four particles adjacent to a tetrahedral element or the three particles adjacent to a triangle. In the soft body case there are three stretch and three shear constraints where as there are two stretch and one shear constraint in the cloth case. The paper above gives the explicit update formulas derived from these constraints.

The stretch constraints formulated as in Equation (32) are quadratic along the gradient and can therefore not be solved in a single step. This problem can be fixed by defining the

© The Eurographics Association 2015.

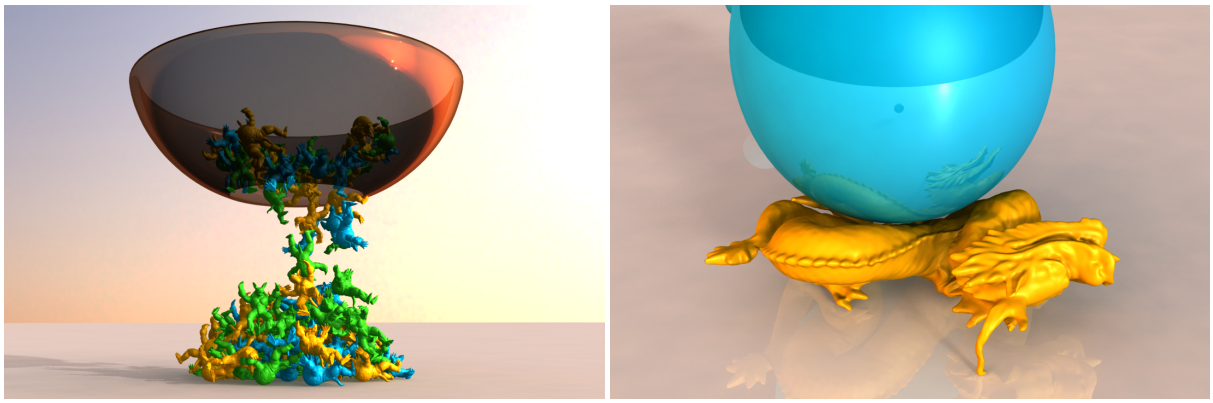

Figure 18: *Position-based simulation using the strain energy constraint. Left: 100 Stanford Armadillos with 371700 tetrahedral elements falling through a funnel. Right: Elastoplastic Stanford Dragon is deformed persistently due to the weight of a heavy sphere.*

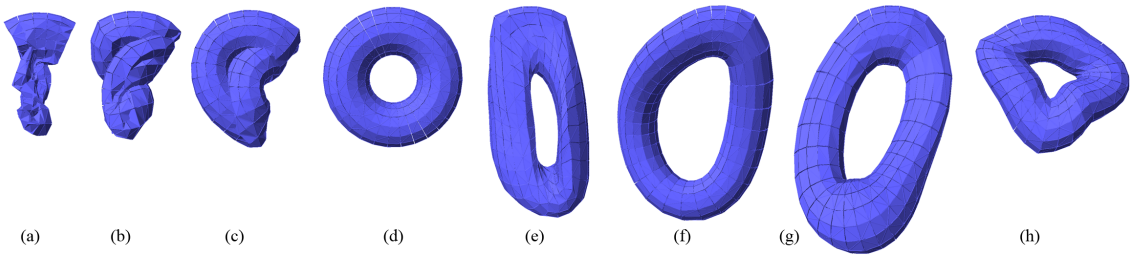

Figure 20: *Varying soft body stiffness parameters. Figures (a) - (d) show the recovery of a torus from a heavily entangled state by increasing the volume stiffness. For (e) we reduced all but the volume conservation stiffness values. As a result, the torus heavily deforms but its volume is conserved. Figure (f) shows the result of only softening the volume stiffness and the stiffness along the main axis of the torus. The result of high shear and low stretch resistance is shown in Figure (g) where angle distortion is small while the shape is stretched. Figure (h) shows the opposite configuration. Here, stretching is small while the torus bends heavily.*

stretch constraints as

$$C_{\text{stretch}}(\mathbf{x}) = \sqrt{\mathbf{S}_{ii}} - 1,$$

which is linear along the constraint gradient.

The shear constraint function $\mathbf{S}_{ij}$ can also be written as $\mathbf{S}_{ij} = \mathbf{f}_i \cdot \mathbf{f}_j$, where $\mathbf{f}_i$ and $\mathbf{f}_j$ are the i^{th} and j^{th} column vectors of $\mathbf{F}$. However, this function not only penalizes the angle between the axes of the deformed coordinate system, i.e. the dot product of the column vectors, but also the principal stretches, i.e. the magnitudes of the column vectors. The following modification of Equation (33) decouples strain from stretch

$$C_{\text{shear}}(\mathbf{x}) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{|\mathbf{f}_i||\mathbf{f}_j|}.$$

Figure 19 shows Strain Based Dynamics on cloth in action. Even though the tessellation of the mesh is not aligned with the principal directions, stretch and shear w.r.t. to those directions can be controlled. In Figure 20, the deformation of a torus is controlled by varying the stiffnesses of the volume, stretch and shear constraints.

5.9. Rigid Body Dynamics

The position-based simulation method is not limited to particle-based models. It can also be used to simulate articulated rigid body systems with joint and contact constraints [DCB14].

A particle has three translational degrees of freedom (DOF). In addition a rigid body has three rotational ones. We parameterize the rotation by a vector ϑ which represents a rotation of $|\vartheta|$ about the axis $\vartheta/|\vartheta|$ in order to define constraint functions $\mathbf{C}(\mathbf{x}, \vartheta)$ for positions and orientations. The vector ϑ is also known as the exponential map [Gra98]. Analogous to Equation (5) each constraint for rigid bodies is

© The Eurographics Association 2015.

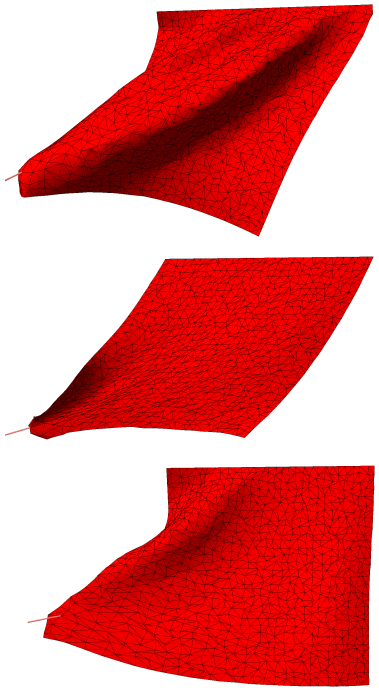

Figure 19: *Varying the cloth stiffness parameters of different strain components. From top to bottom the resistance to x-stretch, y-stretch and shear are: (high,high,high), (hight,high,low) and (low,high,high). Our method allows the control of these modes independently on triangle meshes with highly non-regular tessellations as the one used here.*

approximated by a linearization of the constraint equation:

$$\mathbf{C}(\mathbf{x}+\Delta\mathbf{x},\vartheta+\Delta\vartheta) \approx \mathbf{C}(\mathbf{x},\vartheta)+\mathbf{J}(\mathbf{x},\vartheta)\left(\Delta\mathbf{x}^T,\Delta\vartheta^T\right)^T.$$

However, instead of formulating constraints with respect to $\mathbf{x}$ and ϑ it is easier and more intuitive to use the concept of connectors which was introduced by Witkin et al. [WGW90]. A connector can be a point or vector in local coordinates of a rigid body which is used to define a constraint. The definition of connectors allows to formulate generic constraints without knowledge about the body itself. For example a ball joint which removes all translational DOFs between two linked bodies is defined by the constraint

$$\mathbf{C}(\mathbf{P}_1,\mathbf{P}_2) = \mathbf{P}_1 - \mathbf{P}_2 = \mathbf{0},$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are connector points in the first and second body, respectively.

The world space position of a connector point $\mathbf{P}_i$ of a body j with position $\mathbf{x}_j$ and orientation ϑ_j is defined by

$$\mathbf{P}_i(\mathbf{x}_j,\vartheta_j) = \mathbf{x}_j + \mathbf{R}(\vartheta_j)\mathbf{r}_i, \tag{34}$$

where $\mathbf{r}_i$ denotes the position of the connector in the local coordinate system of the body. The Jacobian of a constraint function $\mathbf{C}(\mathbf{P})$ which depends on a set of connector points $\mathbf{P}$ is determined by

$$\mathbf{J} = \underbrace{\frac{\partial \mathbf{C}(\mathbf{P})}{\partial \mathbf{P}}}_{\substack{\text{constraint}\\ \text{specific part}}} \cdot \underbrace{\begin{pmatrix} \frac{\partial \mathbf{P}}{\partial \mathbf{x}} & \frac{\partial \mathbf{P}}{\partial \vartheta} \end{pmatrix}^T}_{\substack{\text{connector}\\ \text{specific part}}},$$

where the first term is constraint specific and can be computed without knowledge of the body while the second term only depends on the connector type.

For our ball joint example the constraint specific part of the Jacobian is determined by $\partial\mathbf{C}(\mathbf{P})/\partial\mathbf{P}_1 = -\partial\mathbf{C}(\mathbf{P})/\partial\mathbf{P}_2 = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The connector specific part for a point connector is obtained by deriving Equation (34) with respect to $\mathbf{x}$ and ϑ. The first term is determined by $\partial\mathbf{P}/\partial\mathbf{x} = \mathbf{I}$ while the second term $\partial\mathbf{P}/\partial\vartheta$ requires the computation of $\partial\mathbf{R}(\vartheta)/\partial\vartheta$ which is explained in detail by Grassia [Gra98].

In the following we describe how the position and orientation corrections are computed for rigid bodies. Analogous to Equations (5)-(8) first a Lagrange multiplier λ is determined by solving

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = -\mathbf{C}(\mathbf{x},\vartheta). \tag{35}$$

Then the Lagrange multiplier is used to compute the position and orientation change of the linked rigid bodies

$$\left[\Delta\mathbf{x}^T,\Delta\vartheta^T\right] = \mathbf{M}^{-1}\mathbf{J}^T\lambda. \tag{36}$$

The position-based solver for rigid bodies works analogously to the one for particles. Each constraint is linearized individually and position and orientation corrections are determined in a Gauss-Seidel fashion.

Collisions can be simulated by defining inequality constraints for colliding rigid bodies (see Figure 21, right). These constraints can be handled similar to unilateral particle constraints [DCB14]. Moreover, servo motors can be simulated by combining hinge or slider joints with additional constraints that define the goal positions and orientations for the linked bodies (see Figure 21, left).

5.10. Fluids

It is also possible to simulate fluids in the PBD framework even though it has been used almost exclusively for the simulation of deformable objects. We mention fluids simply as an item in the list of possible constraints because all that is needed to simulate liquids and gases is a specialized constraint.

A straightforward approach would be to model the fluid as a system of particles constrained to maintain a minimum distance from each other, however this leads to granular-like behavior and will typically fail to reach hydrostatic equilibrium when coming to rest. An alternative method is presented

© The Eurographics Association 2015.

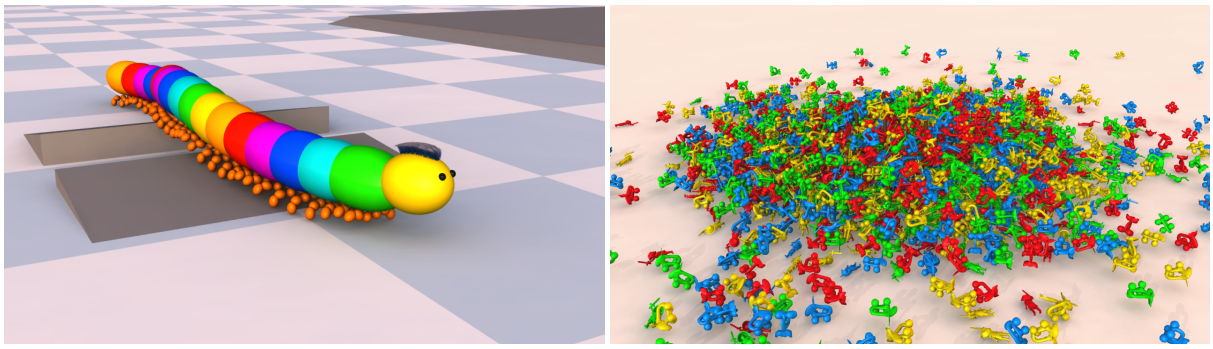

Figure 21: *Left: Millipede with walking over several obstacles. The simulation model consists of 261 rigid bodies, 340 constraints and 240 motors. Right: 2000 rigid bodies collide with each other.*

by Macklin and Müller [MM13] where fluid incompressibility is enforced using density constraints. Borrowing the concept of a density estimator from Smoothed Particle Hydrodynamics (SPH) [Mon94, Mon92], a density constraint is constructed for each particle i in the system as follows

$$C_i(\mathbf{x}_1,...,\mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1, \tag{37}$$

where ρ_0 is the fluid rest density and ρ_i is the density at a particle, defined as the sum of smooth kernels [MCG03] centered at the particle's neighbor positions

$$\rho_i = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h).$$

Note that here each particle's mass is assumed to be one, and the rest density adjusted accordingly. In order to solve these density constraints using position-based dynamics, the derivative of the constraint function (37) with respect to each particle's position is required. This can be calculated using the gradient of SPH kernels

$$\nabla_{\mathbf{x}_k} C_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{\mathbf{x}_k} W(\mathbf{x}_i - \mathbf{x}_j, h) & \text{if } k = i \\ -\nabla_{\mathbf{x}_k} W(\mathbf{x}_i - \mathbf{x}_j, h) & \text{if } k = j. \end{cases}$$

Then, by taking advantage of symmetry in the SPH smoothing kernel W, the corrective change in position due to the particle's own density constraint, and the density constraints of its neighbors is given by

$$\Delta\mathbf{x}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h),$$

where λ is the per-constraint scaling factor (see Equation (6)). Figure 22 shows a real-time water simulation using this method.

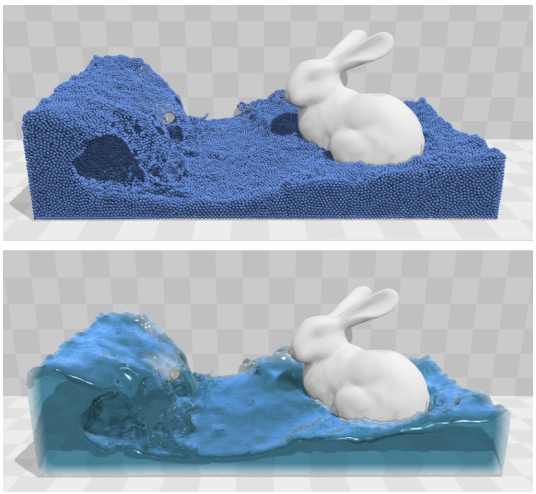

Figure 22: *A wave pool scene consisting of 128k fluid particles simulated in 10ms/frame on the GPU. Incompressibility is enforced using density constraints solved using position-based dynamics.*

5.11. Shape Matching

The geometrically motivated concept of shape matching to simulate deformable objects was introduced by Müller et al. [MHTG05]. Shape matching is a meshless approach which is able to simulate visually plausible elastic and plastic deformations (see Figure 23). This approach is easy to implement, very efficient and unconditionally stable.

Shape matching can be seen as a form of constraint projection which can directly be integrated in the position-based dynamics algorithm. By performing shape matching in line (9) of Algorithm 1 it can be easily combined with other position-based constraints.

The basic idea of simulating elastic behavior with shape matching is shown in Figure 24. For the simulation the initial configuration of the deformable object must be stored. Since no connectivity information is needed, this configuration is defined by the initial positions $\bar{\mathbf{x}}_i$. In each time step the

© The Eurographics Association 2015.

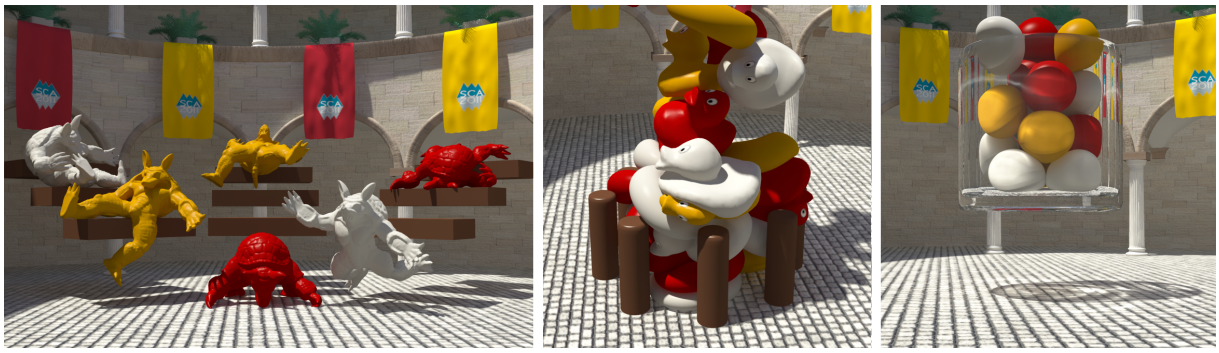

Figure 23: *Robust and volume-conserving deformations using shape matching. Armadillos (32442 particles total), 20 ducks and 20 tori (21280 particles total) and 20 balls (7640 particles total) were simulated in real-time on a GPU.*

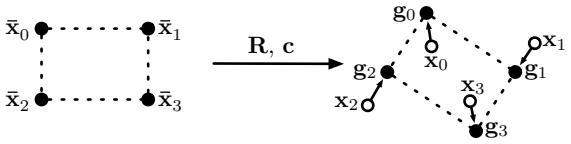

Figure 24: *The initial shape with the vertex positions $\bar{\mathbf{x}}_i$ is matched to the deformed configuration $\mathbf{x}_i$ to obtain goal positions $\mathbf{g}_i$. The deformed shape is pulled towards these goal positions to simulate elastic behavior.*

positions and velocities of the particles are updated without considering any internal constraints between the particles. Only external forces and collision response are taken into account. Instead of using internal constraints, goal positions are determined by matching the initial shape with the deformed configuration. Then, each particle is pulled towards its goal position.

In the following we first describe how the goal positions are determined. Then we show how large deformations can be simulated using region-based shape matching and introduce fast summation techniques for this approach. In the end the concept of oriented particles and different extensions of the shape matching method are presented.

5.11.1. Goal Positions

In order to obtain goal positions for the deformed shape the best rigid transformation is determined which matches the set of initial positions $\bar{\mathbf{x}}$ and the set of deformed positions $\mathbf{x}$. The corresponding rotation matrix $\mathbf{R}$ and the translational vectors $\mathbf{c}$ and $\bar{\mathbf{c}}$ are determined by minimizing

$$\sum_i w_i \left(\mathbf{R}\left(\bar{\mathbf{x}}_i - \bar{\mathbf{c}}\right) + \mathbf{c} - \mathbf{x}_i\right)^2,$$

where w_i are the weights of the individual points. The optimal translation vectors are given by the center of mass of the initial shape and the center of mass of the deformed shape:

$$\bar{\mathbf{c}} = \frac{1}{M}\sum_i m_i \bar{\mathbf{x}}_i, \quad \mathbf{c} = \frac{1}{M}\sum_i m_i \mathbf{x}_i, \quad M = \sum_i m_i. \tag{38}$$

If we minimize the term $\sum_i (\mathbf{A}\bar{\mathbf{r}}_i - \mathbf{r}_i)^2$ with $\mathbf{r}_i = \mathbf{x}_i - \mathbf{c}$ and $\bar{\mathbf{r}}_i = \bar{\mathbf{x}}_i - \bar{\mathbf{c}}$, we get the optimal linear transformation $\mathbf{A}$ of the initial and the deformed shape. This transformation is determined by:

$$\mathbf{A} = \left(\sum_i m_i \mathbf{r}_i \bar{\mathbf{r}}_i^T\right)\left(\sum_i m_i \bar{\mathbf{r}}_i \bar{\mathbf{r}}_i^T\right)^{-1} = \mathbf{A}_r \mathbf{A}_s. \tag{39}$$

In our case we are only interested in the rotational part of this transformation. Since $\mathbf{A}_s$ is symmetric, it contains no rotation. Therefore, we only need to extract the rotational part of $\mathbf{A}_r$ to get the optimal rotation $\mathbf{R}$ for shape matching. This can be done by a polar decomposition $\mathbf{A}_r = \mathbf{RS}$ of the transformation matrix where $\mathbf{S}$ is a symmetric matrix.

Finally, the goal positions are determined by

$$\mathbf{g}_i = \mathbf{T}\begin{bmatrix}\bar{\mathbf{x}}_i \\ 1\end{bmatrix},$$

where $\mathbf{T} = \begin{bmatrix}\mathbf{R} & (\mathbf{c} - \mathbf{R}\bar{\mathbf{c}})\end{bmatrix}$. These goal positions are used to compute position corrections:

$$\Delta\mathbf{x}_i = \alpha\left(\mathbf{g}_i(t) - \mathbf{x}_i(t)\right),$$

where $\alpha \in [0,1]$ is a user-defined stiffness parameter which defines how far the particles are pulled to their goal positions.

5.11.2. Region-Based Shape Matching

The shape matching algorithm described above allows only for small deviations from the initial shape. For the simulation of large deformations the concept of region-based shape matching became popular, see e.g. [MHTG05, RJ07,

© The Eurographics Association 2015.

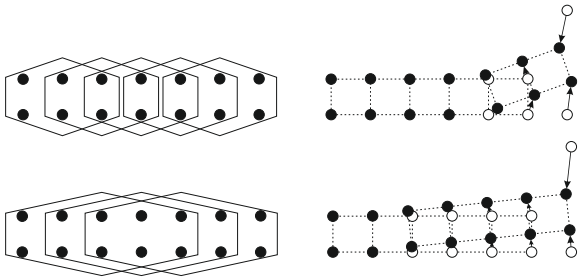

Figure 25: *The stiffness of the model depends on the region size. Smaller regions (top) allow larger deformations than larger regions (bottom). The hexagons in the left images represent the overlapping regions of the model. The right images show the goal positions after one particle is moved away.*

DBB11]. The idea is to perform shape matching on several overlapping regions of the original shape. In each region we can have a small deviation from the corresponding part of the initial shape which results in a large deformation over all regions.

Diziol et al. [DBB11] propose to define a region for each particle of the model where the i-th region contains all particles in the ω-ring of the i-th particle in the original mesh of the model. Shape matching is a meshless method but Diziol et al. require a mesh to define the shape matching regions. Rivers and James [RJ07] use a regular lattice instead to define their regions. No matter which kind of regions are used, the stiffness of the model depends on the size of the overlapping regions (see Figure 25). Enlarging the regions results in a more global shape matching and therefore the stiffness of the simulated model is increased.

In region-based shape matching a particle is part of multiple regions. In the following we denote the set of regions to which a particle i belongs by $\Re_i$. Since particles can belong to more than one region, Rivers and James [RJ07] proposed to use modified particle masses $\tilde{m}_i = m_i/|\Re_i|$ for shape matching. This ensures that a particle which is part of many regions has not more influence than others. The optimal translation vectors for a region i are determined by

$$\bar{\mathbf{c}}_i = \frac{1}{\tilde{M}_i} \sum_{j \in \Re_i} \tilde{m}_j \bar{\mathbf{x}}_j, \quad \mathbf{c}_i = \frac{1}{\tilde{M}_i} \sum_{j \in \Re_i} \tilde{m}_j \mathbf{x}_j, \qquad (40)$$

where $\tilde{M}_i = \sum_{j \in \Re_i} \tilde{m}_j$ is the effective region mass which can be precomputed. The optimal rotation matrix $\mathbf{R}$ is computed by extracting the rotational part of the following matrix:

$$\mathbf{A}_{r,i} = \sum_{j \in \Re_i} \tilde{m}_j \mathbf{x}_j \bar{\mathbf{x}}_j^T - \tilde{M}_i \mathbf{c}_i \bar{\mathbf{c}}_i^T. \qquad (41)$$

In this form the first term depends on the particles j of the region while the second term depends on the region i. This isolation of the dependencies is required for fast summation techniques (see below).

After performing shape matching for all regions, we get multiple goal positions for each particle. The final goal position for a particle is determined by blending the goal positions of the corresponding regions:

$$\mathbf{g}_i = \frac{1}{|\Re_i|} \sum_{j \in \Re_i} \mathbf{T}_j \begin{bmatrix} \bar{\mathbf{x}}_i \\ 1 \end{bmatrix}.$$

5.11.3. Fast Summation Techniques

In the case of region-based shape matching the stiffness increases with growing region size ω. However, at the same time the computation of the optimal translation $\mathbf{c}$ and the transformation matrix $\mathbf{A}_r$ becomes a bottleneck since large sums have to be computed for each region. For a mesh with the dimension d and n regions, $O(\omega^d n)$ operations are required with the naive approach.

5.11.3.1. Regular Lattices Rivers and James demonstrated in [RJ07] how the number of operations for computing the sums can be reduced to $O(n)$ for regular lattices ($d = 3$). Their optimization is closely related to the concept of summed-area tables [Cro84]. In their approach they compute the summation for a set of particles just once and reuse it for all regions that contain this set. This reduces redundant computations significantly for a system with large overlapping regions. The fast summation of Rivers and James is based on the usage of cubical regions. These cubical regions can be subdivided in two-dimensional plate regions which can again be subdivided in one-dimensional bar regions. The region summation is performed in three passes. In the first pass the sum for each bar is determined. The results are used to compute the sums for the plates which are again used to obtain the final region sum. Each pass requires $O(\omega)$ operations. However, the region sum can even be determined in constant time if we take into account that the sum of two neighboring bars, plates or cubes only differs by one element. Lattice shape matching can be performed in linear time if the sums in Equations (40) and (41) are evaluated using the fast summation technique described above.

The FastLSM method of Rivers and James has several limitations. To handle regions where the lattice is not regular, e.g. on the boundary, several sums are defined in a preprocessing step for the corresponding node. In the case of fracturing the definition of these sums must be performed at run-time which is expensive to compute. Small features need a fine sampling to obtain realistic results. Since a regular lattice is used, a fine sampling yields an explosion of the computational costs. FastLSM does not support a varying region size to simulate inhomogeneous material.

5.11.3.2. Adaptive Lattices Steinemann et al. [SOG08] introduce an adaptive shape matching method which is based on lattice shape matching to overcome these limitations. A

© The Eurographics Association 2015.

fast summation is realized by an octree-based sampling and an interval-based definition of the shape matching regions. The hierarchical simulation model is created by starting with a coarse cubic lattice and then performing an octree subdivision. The subdivision process can be controlled by a user-defined criterion. At the end of the process a simulation node is placed at the center of each leaf cell and a virtual node at the center of each non-leaf cell. A virtual node stores the sum of all its descendant simulation nodes.

The fast summation for the hierarchical model is performed by an interval-based method which requires $O(1)$ operations per region. For each simulation node n_i a shape matching region is defined by a region width ω_i. To perform a fast summation, all summation nodes of the region i are determined in a pre-processing step. First, for each node n_j of the octree the interval of minimum and maximum distances of all descendant leaves of n_j to n_i are determined. Then, during a top-down traversal each node n_j where the maximum distance is smaller than the region width is added to region i. If the descendant leaf nodes are contained only partially in region i, the current node must be refined. Only in this case the traversal continues.

The top-down traversal assigns $O(1)$ summation nodes to each region. A fast summation can now be performed in two steps. In the first step the sums of all nodes in the hierarchy are determined. This is done by first computing the sums for the simulation nodes which are the leaf nodes of the hierarchy, and then updating the sums of the virtual nodes in a bottom-up fashion. The second step sums up the values of the summation nodes for each region. For a roughly balanced octree the computation of the sums takes $O(n)$ time where n is the number of simulation nodes. Hence, the adaptive shape matching method requires linear time when using the described fast summation technique to evaluate Equations (40) and (41).

5.11.3.3. Triangle Meshes In contrast to Rivers and James, Diziol et al. [DBB11] only use the surface mesh of a volumetric model to simulate its deformation. Therefore, no interior elements are required for the simulation which reduces the computational costs. Diziol et al. introduce a fast summation technique for arbitrary triangle meshes ($d = 2$) to compute the large sums of the region-based approach efficiently. This technique only requires $O(\omega n)$ operations instead of $O(\omega^2 n)$ and can be performed very efficiently in parallel.

The fast summation technique of Diziol et al. is based on a subdivision of all particles of the mesh in disjoint paths. A path i is a set of vertices $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_n}$ which are connected by edges. The paths are determined in a precomputation step. The goal of the path construction algorithm is that each region is intersected by a minimum number of paths. To determine the optimal path layout is computationally expensive. Therefore, a heuristic is used to find a good path layout. Starting with a single vertex, adjacent vertices are added to a path until the path length exceeds a maximum size or cannot be extended any further. The heuristic tries to avoid gaps by choosing vertices which have neighbors that are already part of a path. To obtain paths which are as parallel as possible we add the vertex which is closest to a plane passing through the starting vertex of the current path, e.g. the xy-plane.

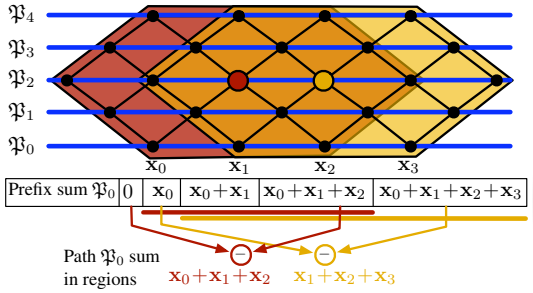

Figure 26: *Fast summation technique for arbitrary triangle meshes [DBB11]. First the prefix sums for the disjoint paths are determined. Then the region sum is computed by adding the difference of the intersection interval for each path.*

The fast summation is split in two phases (see Figure 26). In the first phase the prefix sum for each path i is computed with $j \in [1, n_i]$:

$$\mathbf{c}^p_{i_j} = \sum_{k=1}^{j} \tilde{m}_{i_k} \mathbf{x}_{i_k}, \qquad \mathbf{A}^p_{i_j} = \sum_{k=1}^{j} \tilde{m}_{i_k} \mathbf{x}_{i_k} \bar{\mathbf{x}}^T_{i_k}.$$

Since the prefix sums for all paths are independent of each other, they can be computed in parallel. The sums for a region r are computed by first setting $\mathbf{c}_r := \mathbf{0}$ and $\mathbf{A}_r := \mathbf{0}$. Then for each path i which intersects the region in the interval $[i_k, \ldots, i_l]$, the following terms are added:

$$\mathbf{c}_r := \mathbf{c}_r + \mathbf{c}^p_{i_l} - \mathbf{c}^p_{i_{k-1}}, \qquad \mathbf{A}_r := \mathbf{A}_r + \mathbf{A}^p_{i_l} - \mathbf{A}^p_{i_{k-1}}. \tag{42}$$

The final translational vector and the affine matrix are determined by $\mathbf{c}_r := (1/\tilde{M}_r)\mathbf{c}_r$ and $\mathbf{A}_r := \mathbf{A}_r - \tilde{M}_r \mathbf{c}_r \bar{\mathbf{c}}^T_r$ respectively.

5.11.4. Oriented Particles

For a small number of particles or particles that are close to co-linear or co-planar (as in Figure 27), the matrix $\mathbf{A}_r$ in Equation (39) becomes ill-conditioned and the polar decomposition needed to obtain the optimal rotation tends to be numerically unstable.

To solve this problem, Müller et al. [MC11] proposed to use oriented particles. By adding orientation information to particles, the polar decomposition becomes stable even for single particles. The moment matrix of a single spherical particle with orientation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and finite radius r at the origin is well defined and can be computed via an integral

© The Eurographics Association 2015.

over its volume as

$$\begin{aligned}\mathbf{A}_{\text{sphere}} &= \int_{V_r} \rho(\mathbf{R}\mathbf{x})\mathbf{x}^T dV = \rho\mathbf{R}\int_{V_r} \mathbf{x}\mathbf{x}^T dV \\ &= \frac{4}{15}\pi r^5 \rho\mathbf{R} = \frac{4}{15}\pi r^5 \frac{m}{V_r}\mathbf{R} \\ &= \frac{1}{5}mr^2\mathbf{R},\end{aligned}$$

where V_r is the volume of a sphere of radius r. Since $\mathbf{R}$ is an orthonormal matrix, $\mathbf{A}_i$ always has full rank and an optimal condition number of 1. For an ellipsoid with radii a, b and c we get

$$\mathbf{A}_{\text{ellipsoid}} = \frac{1}{5}m \begin{bmatrix} a^2 & 0 & 0 \\ 0 & b^2 & 0 \\ 0 & 0 & c^2 \end{bmatrix} \mathbf{R}.$$

However, the moment matrices of the individual particles cannot simply be added because each one is computed relative to the origin. We need the moment matrix of particle i relative to the position $\mathbf{x}_i - \mathbf{c}$.

Fortunately, this problem can be fixed easily. As we saw above, the equation for computing the moment matrix

$$\mathbf{A} = \sum_i m_i(\mathbf{x}_i - \mathbf{c})(\bar{\mathbf{x}}_i - \bar{\mathbf{c}})^T \tag{43}$$

can be re-written as

$$\mathbf{A} = \sum_i m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T - M\mathbf{c}\bar{\mathbf{c}}^T,$$

where $\bar{\mathbf{c}}$ and $\mathbf{c}$ are the centers of mass of the initial and the deformed shape, respectively (see Equation (38)).

Therefore, shifting the evaluation from the origin to the position $\mathbf{x}_i - \mathbf{c}$ yields

$$\mathbf{A}_i^{\text{global}} = \mathbf{A}_i + m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T - m_i \mathbf{c}\bar{\mathbf{c}}^T.$$

Equation (43) now generalizes to

$$\begin{aligned}\mathbf{A} &= \sum_i \left(\mathbf{A}_i + m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T\right) - M\mathbf{c}\bar{\mathbf{c}}^T \\ &= \sum_i \left(\mathbf{A}_i + m_i(\mathbf{x}_i - \mathbf{c})(\bar{\mathbf{x}}_i - \bar{\mathbf{c}})^T\right).\end{aligned}$$

As you can see, the last form looks like Equation (43) but with all the individual particle moment matrices added in the sum.

In addition to position $\mathbf{x}$ and velocity $\mathbf{v}$, oriented particles carry a rotation which can be defined as an orthonormal matrix $\mathbf{R}$ as above or a unit quaternion $\mathbf{q}$. They also carry the angular velocity ω. In the prediction step of position-based dynamics, these two quantities have to be integrated as well:

$$\begin{aligned}\mathbf{x}_p &\leftarrow \mathbf{x} + \mathbf{v}\Delta t \\ \mathbf{q}_p &\leftarrow \left[\frac{\omega}{|\omega|}\sin(\frac{|\omega|\Delta t}{2}), \cos(\frac{|\omega|\Delta t}{2})\right] \mathbf{q}.\end{aligned}$$

For stability reasons, $\mathbf{q}_p$ should directly be set to $\mathbf{q}$ if $|\omega| < \varepsilon$.

After the prediction step, the solver iterates multiple times through all shape match constraints in a Gauss-Seidel type fashion as before. To simulate objects represented by a mesh of linked particles, Müller and Chentanez [MC11] define one shape matching group per particle. A group contains the corresponding particle and all the particles connected to it via a single edge. The positions of the particles in a group are updated as in regular shape matching by pulling them towards the goal positions while the orientation of the center particle only is replaced by the optimal rotation of shape matching.

After the solver has modified the predicted state $(\mathbf{x}_p, \mathbf{q}_p)$, the current state is updated using the integration scheme

$$\begin{aligned}\mathbf{v} &\leftarrow (\mathbf{x}_p - \mathbf{x})/\Delta t \\ \mathbf{x} &\leftarrow \mathbf{x}_p \\ \omega &\leftarrow \text{axis}(\mathbf{q}_p\mathbf{q}^{-1}) \cdot \text{angle}(\mathbf{q}_p\mathbf{q}^{-1})/\Delta t \\ \mathbf{q} &\leftarrow \mathbf{q}_p,\end{aligned}$$

where axis() returns the normalized direction of a quaternion and angle() its angle. Again, for stability reasons, ω should be set to zero directly if $|\text{angle}(\mathbf{q}_p\mathbf{q}^{-1})| < \varepsilon$. There are two rotations, $\mathbf{r} = \mathbf{q}_p\mathbf{q}^{-1}$ and $-\mathbf{r}$ transforming $\mathbf{q}$ into $\mathbf{q}_p$. It is important to always choose the shorter one, i.e. if $\mathbf{r}_w < 0$ use $-\mathbf{r}$, where $\mathbf{r}_w$ is the real part of the quaternion. As in traditional PBD for translation, changing the rotational quantity $\mathbf{q}_p$ in the solver also affects its time derivate ω through the integration step creating the required second order effect.

The orientation information of particles cannot only be used to stabilize shape matching but also to move a visual mesh along with the physical mesh. With position and orientation, each particle defines a full rigid transformation at every point in time. This allows the use of traditional linear blend skinning with particles replacing skeletal bones.

An additional advantage of having orientation information is that ellipsoids can be used as collision volumes for particles. This allows a more accurate approximation of the object geometry than with the same number of spherical primitives (see Figure 28).

5.11.5. Plastic Deformation

Shape matching can be extended in order to simulate plastic deformations [MHTG05]. If we perform a polar decomposition $\mathbf{A}_r = \mathbf{RS}$ for the linear transformation matrix $\mathbf{A}_r$ (see Equation (39)), we get a rotational part $\mathbf{R}$ and a symmetric part $\mathbf{S} = \mathbf{R}^T\mathbf{A}_r$. The matrix $\mathbf{S}$ represents a deformation in the unrotated reference frame. Hence, for each region we can store the plastic deformation state in a matrix $\mathbf{S}^p$ which is initialized with the identity matrix $\mathbf{I}$. As proposed by Goktekin et al. [GBO04], we use two parameters c_{yield} and c_{creep} to control the plastic behavior of the material. If the condition $\|\mathbf{S} - \mathbf{I}\|_2 > c_{\text{yield}}$ is fulfilled for the deformation matrix $\mathbf{S}$ of the current time step, the plastic deformation state is updated as follows:

$$\mathbf{S}^p \leftarrow [\mathbf{I} + \Delta t c_{\text{creep}}(\mathbf{S} - \mathbf{I})]\,\mathbf{S}^p.$$

© The Eurographics Association 2015.

Figure 27: *This underwater scene demonstrates the ability of the oriented particle approach to handle sparse meshes such as the one-dimensional branches of the plants or the fins of the lion fish.*

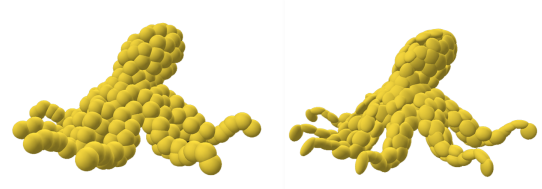

Figure 28: *The rotation information of oriented particles cannot only be used to stabilize shape matching, it also allows the use of ellipsoids as collision primitives. The figure shows how the same mesh is approximated much more accurately with ellipsoids (right) than with the same number of spheres (left).*

After this update, $\mathbf{S}^p$ is divided by $\sqrt[3]{\det(\mathbf{S}^p)}$ in order to conserve the volume. The plastic state $\mathbf{S}^p$ is integrated in the shape matching process by deforming the reference shape in Equation (39). This is done by replacing the definition of $\bar{\mathbf{r}}_i$ (see Section 5.11.1) with

$$\bar{\mathbf{r}}_i = \mathbf{S}^p (\bar{\mathbf{x}}_i - \bar{\mathbf{c}}) .$$

Note that the plasticity can be bound by the condition $\|\mathbf{S}^p - \mathbf{I}\|_2 > c_{\max}$ where $c_{\max}$ is the threshold for the maximum plastic deformation. If this condition is fulfilled, we use $\mathbf{S}^p \leftarrow \mathbf{I} + c_{\max}(\mathbf{S}^p - \mathbf{I})/\|\mathbf{S}^p - \mathbf{I}\|_2$.

Figure 29: *A stiff cloth model with 32467 triangles is simulated using multi-resolution shape matching with five hierarchy levels.*

5.11.6. Cloth Simulation

Stumpp et al. [SSBT08] present a region-based shape matching approach for the simulation of cloth. In their work they define a region for each triangle in the model. But instead of using the triangles directly as regions for shape matching, overlapping regions are defined. The region of a triangle is defined by the outer corners of its adjacent triangles. These overlapping regions enable the bending resistance of the cloth model. Since the model of Stumpp et al. uses regions with only three vertices, the stiffness of high resolution models is too low for realistic results. Therefore, they introduce so-called fiber clusters to increase the stretching stiffness. These one-dimensional regions are determined in a pre-processing step by subdividing the mesh into multiple edge strips. During the simulation each strip is traversed in both directions to obtain additional goal positions. The resulting displacements are translated so that they sum up to 0 to preserve the momentum of the model. The final goal positions are blended with the goal positions of the triangular regions.

The usage of fiber clusters increases the stiffness of the cloth model. However, this effect is limited and for high-resolution models the stiffness is still too low to achieve a realistic cloth behavior. Bender et al. [BWD13] solve this problem by the introduction of multi-resolution shape matching (see Figure 29) which is based on the idea of multi-grid solvers [Hac85]. A shape matching region is defined for each edge and each triangle in a cloth model. To increase the influence of these simple regions and therefore the stretching and shearing stiffness of the model, shape matching is performed on different resolution levels. Multi-resolution shape matching enables the robust simulation of stiff cloth models in linear time.

In the following we first describe 2D shape matching for

© The Eurographics Association 2015.

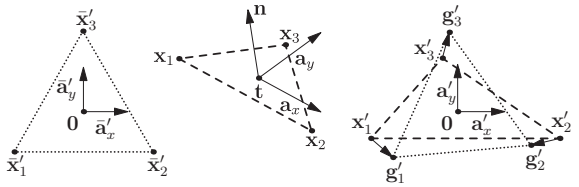

Figure 30: *2D shape matching. The initial configuration of a triangle in 2D (left) is matched to the deformed configuration (middle) by projecting the deformed triangle into 2D and computing the optimal translation and rotation to get goal positions (right).*

triangular regions and then introduce multi-resolution shape matching.

For a cloth simulation with triangular regions, shape matching is performed per triangle in the two-dimensional space of the triangle plane. First the optimal translation vectors of the regions are computed by evaluating Equation (40). Then, for each triangle with the vertices $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ and the normal $\mathbf{n}$ a projection matrix is determined:

$$\mathbf{P} = \begin{pmatrix} \mathbf{a}_x^T \\ \mathbf{a}_y^T \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

with

$$\mathbf{a}_x = \frac{\mathbf{x}_2 - \mathbf{x}_1}{\|\mathbf{x}_2 - \mathbf{x}_1\|}, \quad \mathbf{a}_y = \frac{\mathbf{n} \times \mathbf{a}_x}{\|\mathbf{n} \times \mathbf{a}_x\|}.$$

The matrix $\mathbf{P}$ is used to project the vectors $\mathbf{r}$ and $\bar{\mathbf{r}}$ in Equation (39) to get a 2D version of the matrix $\mathbf{A}_r$:

$$\bar{\mathbf{r}}'_i = \bar{\mathbf{P}}(\bar{\mathbf{x}}_i - \bar{\mathbf{c}}), \quad \mathbf{r}'_i = \mathbf{P}(\mathbf{x}_i - \mathbf{c}),$$

where $\bar{\mathbf{r}}'_i \in \mathbb{R}^2$ can be precomputed. The optimal rotation for shape matching is obtained by performing a 2D polar decomposition [SD92] for the resulting matrix $\mathbf{A}'_r \in \mathbb{R}^{2\times 2}$. This rotation matrix is used to compute 2D goal positions $\mathbf{g}'_i$ for the particles and the corresponding 2D position changes $\Delta\mathbf{x}'_i$:

$$\mathbf{g}'_i = \mathbf{R}'\bar{\mathbf{r}}'_i, \quad \Delta\mathbf{x}'_i = \alpha\frac{1}{|\Re_i|}(\mathbf{g}'_i - \mathbf{x}'_i).$$

Finally, the vectors $\Delta\mathbf{x}'_i$ are transformed to world space by $\Delta\mathbf{x}_i = \mathbf{P}^T\Delta\mathbf{x}'_i$ and the particle positions are updated. This process is shown in Figure 30.

In a simulation with multi-resolution shape matching [BWD13] two intergrid transfer operators are required to couple the different meshes in the multi-resolution hierarchy. The restriction operator $\mathbf{I}_{l+1}^l$ transfers values from level $l+1$ to the next coarser level l and the prolongation operator $\mathbf{I}_l^{l+1}$ transfers values in the opposite direction. These operators can be defined by barycentric coordinates [GW06]. In each simulation step first the positions of the finest mesh are updated by time integration. For non-nested models the positions of the coarser meshes are interpolated using the restriction operator. Then multi-resolution shape matching is performed in a V-cycle as described by Algorithm 2.

Algorithm 2 Multi-resolution shape matching

for $l = l_{max}$ **to** 1 **do**
  Store current positions: $\hat{\mathbf{x}}^l \leftarrow \mathbf{x}^l$
  Perform shape matching
  $\mathbf{x}^{l-1} := \mathbf{x}^{l-1} + \mathbf{I}_l^{l-1}(\mathbf{x}^l - \hat{\mathbf{x}}^l)$
end for
for $l = 0$ **to** l_{max} **do**
  Store current positions: $\hat{\mathbf{x}}^l \leftarrow \mathbf{x}^l$
  Perform shape matching
  if $l \neq l_{max}$ **then**
    $\mathbf{x}^{l+1} := \mathbf{x}^{l+1} + \mathbf{I}_l^{l+1}(\mathbf{x}^l - \hat{\mathbf{x}}^l)$
  end if
end for

In the restriction phase the hierarchy is traversed from the finest to the coarsest mesh performing a shape matching step on each level and projecting the resulting position differences $\mathbf{x}^l - \hat{\mathbf{x}}^l$ to the next coarser level with the restriction operator. In the prolongation phase the hierarchy is traversed in the opposite direction. On each level a shape matching step is performed and the position differences are interpolated and added to the next finer level. Since only position differences are propagated between the levels, fine details are conserved on finer levels. However, fine details could get lost if the original shape matching method is used on the coarse levels of the hierarchy. Wrinkles on a fine resolution cause a compression of elements on a coarser level. Shape matching reduces this compression and thus eliminates fine details. Therefore, Bender et al. [BWD13] propose a modified computation of the goal positions on the coarse levels of the hierarchy so that shape matching only prevents stretching on these levels but not a compression.

© The Eurographics Association 2015.

6. Implementation

6.1. Parallelization

The parallelization of the position-based approach is an important topic since multi-core systems and massively parallel GPUs are ubiquitous today.

6.1.1. Graph-Coloring Methods

In a single CPU implementation, the solver processes the constraints one by one in a Gauss-Seidel-type fashion. Thereby, after each constraint projection, the positions of affected particles are immediately updated. In a parallel implementation, the constraints are processed in parallel by multiple threads. If two constraints affecting the same particle are handled by two different threads simultaneously, they are not allowed to immediately update the particle's position because writing to the same position simultaneously leads to race conditions making the process unpredictable. A solution to circumvent this problem is to use atomic operations. Such operations are guaranteed not to be interrupted. However, atomics can slow down parallel execution significantly.

To avoid these issues, a parallel implementation of PBD needs to split the constraints into groups or phases. In each phase, none of the constraints are allowed to share a common particle. With this restriction, the constraints in the first phase can be processed in parallel without conflicts. Then, after a global synchronization, the next phase can be processed. This cycle is repeated until all constraints are processed.

As an example, if N particles are connected in a serial chain, the constraints $1-2, 3-4, 5-6, 7-8, ..$ can be processed in phase 1 and the constraints $2-3, 4-5, 6-7, ..$ in phase 2. This specific example corresponds to the Red-Black Gauss Seidel scheme, where there are two sets (colors) of constraints. For more general types of constraints such as the stretch, shear and bending constraints of cloth, more phases are needed. In this general case, splitting constraints into phases corresponds to the graph coloring problem, where each constraint corresponds to a node of the graph and two constraints are connected by an edge if they affect one or more common particles. The minimum number of colors determines how many phases are needed in the parallel execution of PBD. Keeping the number of phases small is not the only optimization criterion. The sets also need to have similar sizes for good load balancing.

6.1.2. Jacobi Methods

For some models with high valence, graph-coloring methods may generate poor work load distributions, where initial sets of constraints may be large, but tailing sets are very small. This imbalance leads to resource under-utilization and potentially high synchronization costs when many colors are required. An alternative method for parallelizing PBD is to use a Jacobi-style constraint solver. In a Jacobi solve, each constraint may be processed in parallel, and the position delta for each particle obtained by summing the delta from each constraint at the end of an iteration.

Jacobi methods often converge significantly slower than Gauss-Seidel iteration, and may not converge at all, for example if the system matrix is not positive definite. To address this problem, under-relaxation based on the concept of constraint averaging [BFA02], or mass-splitting [TBV12] can be applied. At the end of the iteration, once all constraints are processed, the particle's total constraint delta is divided by n_i, the number of constraints affecting the particle, to obtain the averaged position update $\Delta\tilde{\mathbf{x}}$:

$$\Delta\tilde{\mathbf{x}}_i = \frac{1}{n_i}\Delta\mathbf{x}_i. \tag{44}$$

This form of local relaxation is not guaranteed to conserve momentum when neighboring particles have differing number of constraints, however, visual errors are typically not noticeable. Averaging constraint forces as described above ensures convergence, but in some cases this averaging is too aggressive and the number of iterations required to reach a solution increases. To address this a global user-parameter ω can be introduced to control the rate of successive over-relaxation (SOR),

$$\Delta\tilde{\mathbf{x}}_i = \frac{\omega}{n_i}\Delta\mathbf{x}_i. \tag{45}$$

We recommend using $1 \leq \omega \leq 2$, although higher values may be used depending on the scene being simulated. Additional under-relaxation ($\omega < 1$) is not typically required as the constraint averaging is sufficient to avoid divergence.

6.1.3. Hybrid Methods

To take advantage of both Gauss-Seidel and Jacobi solvers, Fratarcangeli et al. [FP15] proposed a hybrid approach. They use graph coloring and modify the graph such that it produces a desired number of k colors by splitting high valence particles, i.e. solving them Jacobi style. An even simpler hybrid approach is to solve the first $k-1$ colors using $k-1$ Gauss-Seidel passes and then solve the remaining constraints with one Jacobi pass.

6.1.4. Shape Matching

In Section 5.11.3 we presented different fast summation techniques for shape matching. The one of Diziol et al. [DBB11] is best suited for a parallel implementation on the GPU. In the following the GPU implementation of this technique with CUDA is described in detail. For such an implementation memory access and memory layouts play an important role as well as the number of kernel calls.

Since each kernel call introduces a computational overhead, the particles of all objects in a simulation are packed into one single array. This array is ordered according to the

© The Eurographics Association 2015.

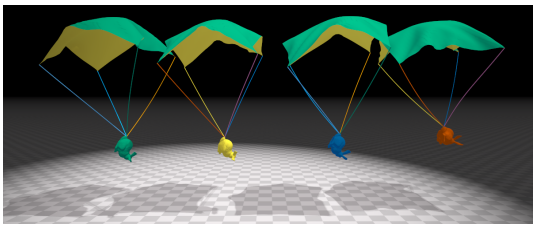

Figure 31: *Rigid body bunnies, attached to cloth by deformable ropes parachute to the ground. A drag model on the clothing particles slows the descent of the bunnies.*

path layout which is used for the fast summation (see Section 5.11.3). Since the array contains the paths one after another, a segmented prefix sum [SHZO07] can be used to determine the prefix sums of all paths at once. To avoid numerical problems due to the 32 bit floating-point arithmetics on the GPU, the path length is limited to 512. The resulting prefix sums are stored in texture memory to benefit from the texture cache when the translational vectors and the affine matrices are determined (see Equation (42)).

The multi-resolution approach described in Section 5.11.6 can be implemented on the GPU as follows. Shape matching on each level of the hierarchy is performed by computing the goal positions per element in parallel in a first step. The results are stored for each element. In a second step shape matching is completed by summing up the contributions of all elements containing a vertex to get a final goal position for the vertex. The restriction and the prolongation of the results can be performed efficiently using the sparse matrix data structure of Weber et al. [WBS*13]. This implementation allows to simulate the deformation of a cloth model with more than 200k triangles on the finest level in 22 ms/step on a GeForce GTX 470.

6.2. Unified Solver

Macklin et al. [MMCK14] present a method that brings together many of the PBD applications in a unified framework. The core idea is to represent everything in the system by particles, and leverage fast GPU particle-particle collision techniques [Gre08] to generate complex interactions efficiently.

In this framework, rigid bodies are represented by voxelizing closed triangle meshes, and adding particles in interior cells. A shape-matching constraint is then added to the system to enforce the rigid particle configuration. Interaction between objects is accomplished by simply connecting particles by constraints, e.g.: tethering a rigid object to the corners of a piece of cloth generates a basic parachute (see Figure 31).

Particles are extended with an integer *phase* attribute, which is used to control the generation of constraints. One possible interpretation of the phase attribute is that particles of the same phase do not generate collision constraints. For example, when modeling rigid bodies, particles belonging to the same body are given the same phase to avoid generating internal collisions.

Fluids are modeled using the density constraint of Section 5.10, because the fluid is also modeled by particles, full two-way coupling with clothing, rigid bodies, and granular materials is possible. Constraints may also be combined to achieve new effects, e.g.: a rigid body constraint combined with the fluid density constraint can be animated to model phase changes such as melting.

7. Applications

In this section we introduce different application areas of position-based methods. These methods are mainly used in interactive applications where performance, controllability and stability are more important than accuracy, like e.g. in [SGdA*10, DB13]. But there exist also other works which use a position-based approach for stabilization.

7.1. Strain Limiting

Strain limiting is an important topic in the field of cloth simulation. The reason is that the low solver iteration counts used in real-time applications yield stretchy cloth. Since most cloth types are perceived by the human eye as completely inextensible, it is important to make simulated cloth inextensible in order to avoid disturbing visual artifacts [GHF*07, BB08].

A strain limiting method makes sure that the overall stretch of the cloth stays below a certain threshold. In force based simulations, strain limiting is a separate pass which is executed before or after the regular cloth solver. In most cases, this pass moves the positions of vertices directly, even in force based simulations. Therefore, most strain limiting methods fall under the category of position-based methods.

A straightforward way of limiting strain is to iterate through all edges of a cloth mesh and project the adjacent particles of overstretched edges as shown in Figure 3 so that the stretch of the edge does not exceed the stretch limit. Provot [Pro95] was among the first to use this method in the context of cloth simulation. He performs a single iteration through all cloth edges after a force based solver. Desbrun et al. [DSB99] and Bridson et al. [BMF03] later used the same post solver strain limiter but with multiple iterations through all edges. Due to its simplicity, this method is still one of the most popular strain limiting methods used in cloth simulations.

The method is very similar to position-based cloth simulation. The main difference is that the strain limiting pass described above does not influence the velocities. These are updated by the force-based solver. In contrast, position-based

© The Eurographics Association 2015.

cloth simulation derives the new velocities from the projections, making an additional solver pass obsolete. Therefore, every position-based strain limiting method used in force based simulations can directly be used in a PBD solver.

The result of projecting along edges depends on the structure of the mesh. To reduce this artifact, Wang et al. [WOR10] propose to limit the principal strains of the 2D deformation field within each triangle. The 2D deformation field can be determined by considering the 2D coordinates of the vertices of a triangle within the planes of the rest and current triangle configurations. Wang et al. compute the principal strains of the 2D deformation gradient, clamp them and construct a new 2D transformation using the clamped strains. With this new transformation they correct the current positions of the triangle vertices. As before, to limit strain globally, they iterate through all triangles multiple times in a Gauss-Seidel fashion.

Due to the relatively slow convergence rate of a Gauss-Seidel solver, high iteration counts are necessary to limit the strain globally which slows down the simulation. The two main methods to improve the convergence rate are the use of a global Newton-Raphson solver as proposed by Goldenthal et al. [GHF*07] or to perform Gauss-Seidel iterations on a hierarchy of meshes as proposed in [Mül08], [WOR10] and [SKBK13]. However, these methods complicate the implementation and even though their convergence rate is higher, a single iteration can be significantly more expensive than a simple Gauss-Seidel iteration.

7.2. Wrinkle Meshes

In cloth simulations, reducing the mesh resolution not only reduces the cost of a single solver iteration but also the number of iterations required to get visually pleasing results. In [MC10] the authors proposed a way to reduce the resolution of the dynamic mesh without losing too much visual detail. The most significant detail in cloth simulations are small wrinkles. The method is based on the observation that global dynamic behavior of the cloth and wrinkle formation can be separated. Therefore, expensive dynamic simulation including collision handling is performed on a low-resolution mesh. The wrinkle formation is handled on a high resolution mesh that is attached to the dynamic mesh (see Figures 32 and 33). Since wrinkles do not oscillate, it is sufficient to use a static solver with a low iteration count on the high-resolution mesh.

Figure 34 shows the constraints defined on the high-resolution mesh to make it form wrinkles and follow the dynamic mesh. The attachment constraints makes sure that the vertices of the wrinkle mesh stay close to their attachment points on the dynamic mesh. If the dynamic mesh has outside/inside information, a one-sided constraint can be used which makes sure that the wrinkle vertices stay on the outside of the dynamic mesh, thus avoiding penetrations with other objects. The stretching and bending constraints are responsible for wrinkle formation.

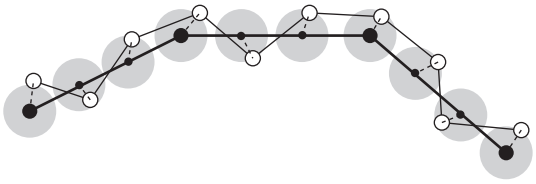

Figure 32: *Basic idea of wrinkle meshes. The high resolution wrinkle mesh (white vertices) follows the low-resolution dynamic mesh (black vertices) by restricting the white vertices to remain within a certain distance (gray discs) to the dynamic mesh.*

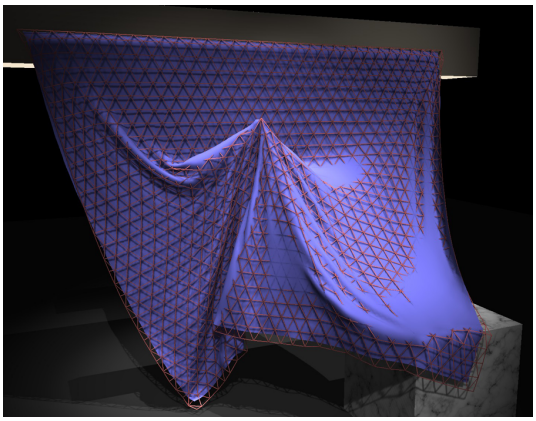

Figure 33: *Visualization of the wrinkle mesh (solid) and the underlying dynamic mesh (wireframe).*

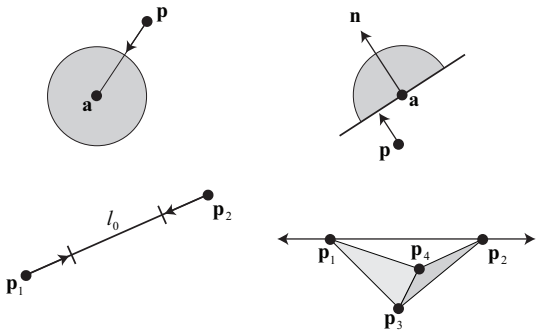

Figure 34: *Static constraints on a wrinkle mesh: Attachment constraint (top left), one sided attachment constraint (top right), stretch constraint (bottom left) and bending constraint (bottom right).*

© The Eurographics Association 2015.

7.3. Further Applications

Another application area for position-based methods is interactive surgical simulation. In this area Wang et al. [WXX*06] introduce a mass-spring model based on a surface mesh to simulate deformable bodies in real-time. Since such a model can neither preserve its volume nor resume its rest shape in the absence of external forces, the authors propose to couple the surface model with a rigid core by using spring forces. This rigid core is simulated using shape matching [MHTG05] which results in a fast and stable simulation. Kubiak et al. [KPGF07] present a simulation method for surgical threads which is based on the position-based dynamics approach of Müller et al. [MHHR07]. Their method simulates the stiffness, bending and torsion of a thread and also provides feedback for a haptic device. For the simulation Kubiak et al. define distance constraints for stiffness and bending, torsion constraints, contact constraints and friction constraints. The presented method allows for an interactive and robust simulation of knots.

The simulation of complex hairstyles using a shape matching approach is presented by Rungjiratananon et al. [RKN10]. Their approach is based on Lattice Shape Matching which was originally introduced by Rivers and James [RJ07]. For the simulation each hair strand is represented by a chain of particles which is subdivided in overlapping chain regions. After shape matching an additional position-based strain limiting is applied to each strand which moves the particles in the direction of their root. Different hair styles are realized by using appropriate initial configurations and by modifying the region sizes of a chain.

Umetani et al. [USS14] use a position-based rod model which is derived from the Cosserat theory in order to simulate complex bending and twisting of elastic rods. The authors define material frames on the centerline of each edge to represent the orientations along the rod. These material frames are represented by ghost points which are coupled with the edges by position-based constraints.

O'Brien et al. [ODC11] use position-based dynamics for the physically plausible adaptation of motion-captured animations. In their work they use a vertex-based character skeleton and different constraints to preserve the skeleton structure, to define joint limits and to implement a center of mass control. In addition to the kinematic constraints, they define a couple of dynamics constraints which consider vertices in multiple frames. Dynamics constraints are used to enforce smooth acceleration and dynamical correctness.

Fierz et al. [FSAH12] introduce a position-based approach to stabilize a finite element simulation. When using an explicit time integration for a finite element simulation, the time step size is typically limited by the stiffness of the model and its spatial discretization. In each simulation step Fierz et al. use the Courant-Friedrichs-Lewy (CFL) condition to determine the maximum allowed time step size for each tetrahedral element in their volumetric simulation model. However, instead of using the time step size given by the CFL condition to perform a stable simulation step with an explicit integration scheme, they use a fixed size and mark all elements where the condition is not met. The marked elements are then simulated using a shape matching approach while for all other elements a linear finite element method is used for the simulation.

8. Conclusion

In this tutorial, we focused on position-based approaches. Such geometrically motivated techniques are not force-driven and are particularly appropriate in interactive applications due to their versatility, robustness, controllability and efficiency. We explained general ideas of position-based methods and introduced several specific constraints. Various aspects and efficient solution strategies were discussed with a particular focus on the benefits of position-based approaches compared to force-driven techniques.

Position-based dynamics is fast, easy to implement and controllable. Furthermore, it avoids the overshooting problems of force-based simulation models when using an explicit time integration scheme. The method can handle arbitrary bilateral and unilateral constraints as long as the gradient of the constraint function can be determined. Therefore, this method is very flexible and has already been used to simulate cloth, deformable solids and fluids.

However, position-based dynamics also has some disadvantages. The stiffness of the model does not only depend on the user-defined stiffness parameter but also on the time step size and the number of solver iterations. Although the dependency can be reduced as described in Section 4.2.2, it cannot be completely removed. Therefore, it is difficult to adjust parameters independently. Decoupling these parameters as well as adaptive time stepping are open problems and important topics for future work. Another drawback is that position-based dynamics is not convergent, i.e. the simulation does not converge to a certain solution with mesh refinement. Hence, the usage of adaptive meshes is another open problem.

Acknowledgments The work of Jan Bender was supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School CE at TU Darmstadt.

References

[BB08] BENDER J., BAYER D.: Parallel simulation of inextensible cloth. In *VRIPHYS 08: Fifth Workshop in Virtual Reality Interactions and Physical Simulations* (2008), pp. 47–56. 27

[Ben07] BENDER J.: *Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität*. PhD thesis, University of Karlsruhe, Germany, 2007. 4

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive simulation of rigid body dynamics in computer graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 4

© The Eurographics Association 2015.

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July 2002), 594–603. 26

[BFS05] BENDER J., FINKENZELLER D., SCHMITT A.: An impulse-based dynamic simulation system for VR applications. In *Proceedings of Virtual Concept* (2005), Springer. 4

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-based simulation of continuous materials. *Computers & Graphics 44*, 0 (2014), 1 – 10. 10, 15, 16

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles,. In *Proc. ACM/Eurographics Symposium on Computer Animation* (2003), pp. 28–36. 27

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. 9

[BMO*14] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M., MACKLIN M.: A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum 33*, 6 (2014), 228–251. 4

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of Computer graphics and interactive techniques* (1998), SIGGRAPH '98, ACM, pp. 43–54. 6

[BWD13] BENDER J., WEBER D., DIZIOL R.: Fast and stable cloth simulation based on multi-resolution shape matching. *Computers & Graphics* (2013). 24, 25

[BWH*06] BERGOU M., WARDETZKY M., HARMON D., ZORIN D., GRINSPUN E.: A quadratic bending model for inextensible surfaces. In *Proc. Symposium on Geometry processing* (2006), Eurographics Association, pp. 227 – 230. 10

[Cro84] CROW F. C.: Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 207–212. 21

[DB13] DEUL C., BENDER J.: Physically-Based Character Skinning. In *VRIPHYS 13: 10th Workshop on Virtual Reality Interactions and Physical Simulations* (Lille, France, 2013), Eurographics Association, pp. 25–34. 27

[DBB09] DIZIOL R., BENDER J., BAYER D.: Volume conserving simulation of deformable bodies. In *Short Paper Proceedings of Eurographics* (Mar. 2009). 12

[DBB11] DIZIOL R., BENDER J., BAYER D.: Robust real-time deformation of incompressible surface meshes. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), SCA '11, Eurographics Association. 12, 20, 21, 22, 26

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid body dynamics. *Computer Animation and Virtual Worlds* (2014). 17, 18

[DSB99] DESBRUN M., SCHRÖDER P., BARR A.: Interactive animation of structured deformable objects. In *Proc. of SIGGRAPH 99* (1999), ACM, pp. 1–8. 4, 27

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. 27*, 3 (2008), 66. 9

[FP15] FRATARCANGELI M., PELLACINI F.: Scalable partitioning for parallel position based dynamics. *EUROGRAPHICS 2015 34*, 2 (2015). 26

[FSAH12] FIERZ B., SPILLMANN J., AGUINAGA I., HARDERS M.: Maintaining large time steps in explicit finite element simulations using shape matching. *Visualization and Computer Graphics, IEEE Transactions on 18*, 5 (may 2012), 717 –728. 29

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 463–468. 23

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3 (2007), 49. 7, 9, 27, 28

[GM97] GIBSON S. F., MIRTICH B.: *A survey of deformable modeling in computer graphics*. Tech. Rep. TR-97-19, Mitsubishi Electric Research Lab., 1997. 4

[Gra98] GRASSIA F. S.: Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools 3* (1998), 29–48. 17, 18

[Gre08] GREEN S.: Cuda particles. *nVidia Whitepaper 2*, 3.2 (2008), 1. 27

[GW06] GEORGII J., WESTERMANN R.: A multigrid framework for real-time simulation of deformable bodies. *Computer & Graphics 30* (2006), 408–415. 8, 25

[Hac85] HACKBUSCH W.: *Multi-Grid methods and applications*, vol. 4 of *Springer Series in Computational Mathematics*. Springer, 1985. 24

[HJCW06] HONG M., JUNG S., CHOI M., WELCH S.: Fast volume preservation for a mass-spring system. *IEEE Comput. Graph. Appl. 26* (2006), 83–91. 12

[ISF07] IRVING G., SCHROEDER C., FEDKIW R.: Volume conserving finite element simulations of deformable models. *ACM Trans. on Graphics 26*, 3 (July 2007), 13:1–13:6. 12

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2004), Eurographics Association, pp. 131–140. 16

[Jak01] JAKOBSEN T.: Advanced character physics. In *Proceedings, Game Developer's Conference 2001* (2001). 10

[JP99] JAMES D. L., PAI D. K.: Artdefo: accurate real time deformable objects. In *Proc. of SIGGRAPH 99* (1999), ACM, pp. 65–72. 4

[KCM12] KIM T.-Y., CHENTANEZ N., MÜLLER M.: Long Range Attachments - A Method to Simulate Inextensible Clothing in Computer Games. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2012), Lee J., Kry P., (Eds.), Eurographics Association, pp. 305–310. 14

[KPGF07] KUBIAK B., PIETRONI N., GANOVELLI F., FRATARCANGELI M.: A robust method for real-time thread simulation. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology* (2007), VRST '07, ACM, pp. 85–88. 29

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 209:1–7. Proceedings of ACM SIGGRAPH Asia 2013, Hong Kong. 8

[LG98] LIN M. C., GOTTSCHALK S.: Collision detection between geometric models: A survey. In *In Proc. of IMA Conference on Mathematics of Surfaces* (1998), pp. 37–56. 4

[MC10] MÜLLER M., CHENTANEZ N.: Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), SCA '10, Eurographics Association, pp. 85–92. 28

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (July 2011), 92:1–92:10. 22, 23

© The Eurographics Association 2015.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), SCA '03, Eurographics Association, pp. 154–159. 19

[MCKM14] MÜLLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Strain based dynamics. In *Proceedings of the 2014 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2014), Eurographics Association. 15, 16

[MCKM15] MÜLLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Air meshes for robust collision handling. *to appear in ACM Trans. Graph.* (2015). 13

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 6, 11, 29

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478. 19, 20, 23, 29

[Mir96] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools 1*, 2 (Feb. 1996), 31–50. 12

[MKC12] MÜLLER M., KIM T.-Y., CHENTANEZ N.: Fast Simulation of Inextensible Hair and Fur. In *VRIPHYS 12: 9th Workshop on Virtual Reality Interactions and Physical Simulations* (2012), Eurographics Association. 14

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. 19

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 11, 27

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 19

[Mon94] MONAGHAN J. J.: Simulating free surface flows with sph. *J. Comput. Phys. 110*, 2 (Feb. 1994), 399–406. 19

[MSJT08] MÜLLER M., STAM J., JAMES D., THÜREY N.: Real time physics: class notes. In *ACM SIGGRAPH 2008 classes* (2008), SIGGRAPH '08, ACM, pp. 88:1–88:90. 4

[MTV05] MAGNENAT-THALMANN N., VOLINO P.: From early draping to haute couture models: 20 years of research. *The Visual Computer 21* (2005), 506–519. 4

[Mül08] MÜLLER M.: Hierarchical Position Based Dynamics. In *VRIPHYS 08: Fifth Workshop in Virtual Reality Interactions and Physical Simulations* (2008), Faure F., Teschner M., (Eds.), Eurographics Association, pp. 1–10. 8, 28

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (December 2006), 809–836. 4, 6

[ODC11] O'BRIEN C., DINGLIANA J., COLLINS S.: Space-time vertex constraints for dynamically-based adaptation of motion-captured animation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), SCA '11, ACM, pp. 277–286. 29

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 137–146. 4

[PB88] PLATT J. C., BARR A. H.: Constraints methods for flexible objects. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), SIGGRAPH '88, ACM, pp. 279–288. 6

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1995), Davis W. A., Prusinkiewicz P., (Eds.), Canadian Human-Computer Communications Society, pp. 147–154. 27

[RJ07] RIVERS A. R., JAMES D. L.: FastLSM: fast lattice shape matching for robust real-time deformation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), ACM, p. 82. 20, 21, 29

[RKN10] RUNGJIRATANANON W., KANAMORI Y., NISHITA T.: Chain shape matching for simulating complex hairstyles. *Computer Graphics Forum 29*, 8 (2010), 2438–2446. 29

[SD92] SHOEMAKE K., DUFF T.: Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92* (1992), Morgan Kaufmann Publishers Inc., pp. 258–264. 25

[SGdA*10] STOLL C., GALL J., DE AGUIAR E., THRUN S., THEOBALT C.: Video-based reconstruction of animatable human characters. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 139:1–139:10. 27

[SGT09] SCHMEDDING R., GISSLER M., TESCHNER M.: Optimized damping for dynamic simulations. In *Spring Conference on Computer Graphics* (2009), pp. 205–212. 6, 7

[SHZO07] SENGUPTA S., HARRIS M., ZHANG Y., OWENS J. D.: Scan primitives for GPU computing. In *Proc. of the 22nd ACM SIGGRAPH/Eurographics Symp. on Grap. Hardware* (2007), Eurographics Association, pp. 97–106. 27

[SKBK13] SCHMITT N., KNUTH M., BENDER J., KUIJPER A.: Multilevel Cloth Simulation using GPU Surface Sampling. In *VRIPHYS 13: 10th Workshop on Virtual Reality Interactions and Physical Simulations* (Lille, France, 2013), Eurographics Association, pp. 1–10. 28

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), SCA '08, Eurographics Association, pp. 87–94. 21

[SSBT08] STUMPP T., SPILLMANN J., BECKER M., TESCHNER M.: A Geometric Deformation Model for Stable Cloth Simulation. In *VRIPHYS 08: Fifth Workshop in Virtual Reality Interactions and Physical Simulations* (2008), Faure F., Teschner M., (Eds.), Eurographics Association, pp. 39–46. 24

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. *IEEE International Conference on Computer-Aided Design and Computer Graphics* (2009), 1–11. 6

[TBHF03] TERAN J., BLEMKER S., HING V. N. T., FEDKIW R.: Finite volume methods for the simulation of skeletal muscle. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* (2003), Eurographics Association, pp. 68–74. 4

[TBV12] TONGE R., BENEVOLENSKI F., VOROSHILOV A.: Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph. 31*, 4 (July 2012), 105:1–105:8. 26

[TF88] TERZOPOULOS D., FLEISCHER K.: Deformable models. *The Visual Computer 4* (1988), 306–331. 6

[THMG04] TESCHNER M., HEIDELBERGER B., MULLER M., GROSS M.: A versatile and robust model for geometrically complex deformable solids. In *Proceedings of the Computer Graphics International* (Washington, DC, USA, 2004), CGI '04, IEEE Computer Society, pp. 312–319. 4

© The Eurographics Association 2015.

[TKH*05] TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNENAT-THALMANN N., STRASSER W., VOLINO P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (Mar. 2005), 61–81. 4

[TPBF87a] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), SIGGRAPH '87, ACM, pp. 205–214. 4

[TPBF87b] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (1987), vol. 21, ACM, pp. 205–214. 4

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based Elastic Rods. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2014), The Eurographics Association. 29

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. on Graphics 25*, 3 (July 2006), 1118–1125. 12

[WBS*13] WEBER D., BENDER J., SCHNOES M., STORK A., FELLNER D.: Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum 32*, 1 (2013), 16–26. 27

[WGW90] WITKIN A., GLEICHER M., WELCH W.: Interactive dynamics. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics* (New York, NY, USA, 1990), ACM Press, pp. 11–21. 18

[Wit97] WITKIN A.: An introduction to physically based modeling: Constrained dynamics, 1997. 5

[WOR10] WANG H., O'BRIEN J., RAMAMOORTHI R.: Multi-resolution isotropic strain limiting. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 156:1–156:10. 28

[WXX*06] WANG Y., XIONG Y., XU K., TAN K., GUO G.: A mass-spring model for surface mesh deformation based on shape matching. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (2006), GRAPHITE '06, ACM, pp. 375–380. 29

© The Eurographics Association 2015.