Position-Based Rigid Body Dynamics

Conference Paper *in* Computer Animation and Virtual Worlds · September 2014

DOI: 10.1002/cav.1614

3 authors, including:

Patrick Charrier
Technische Universität Darmstadt
6 PUBLICATIONS **38** CITATIONS
SEE PROFILE

Jan Bender
RWTH Aachen University
54 PUBLICATIONS **430** CITATIONS
SEE PROFILE

Position-Based Rigid Body Dynamics

Crispin Deul, Patrick Charrier and Jan Bender
Graduate School of Excellence Computational Engineering
Technische Universität Darmstadt, Germany
{deul | charrier | bender}@gsc.tu-darmstadt.de

Abstract

We propose a position-based approach for large-scale simulations of rigid bodies at interactive frame-rates. Our method solves positional constraints between rigid bodies and therefore integrates nicely with other position-based methods. Interaction of particles and rigid bodies through common constraints enables two-way coupling with deformables. The method exhibits exceptional performance and stability while being user-controllable and easy to implement. Various results demonstrate the practicability of our method for the resolution of collisions, contacts, stacking and joint constraints.

Keywords: real-time, rigid body dynamics, two-way coupling, position-based dynamics

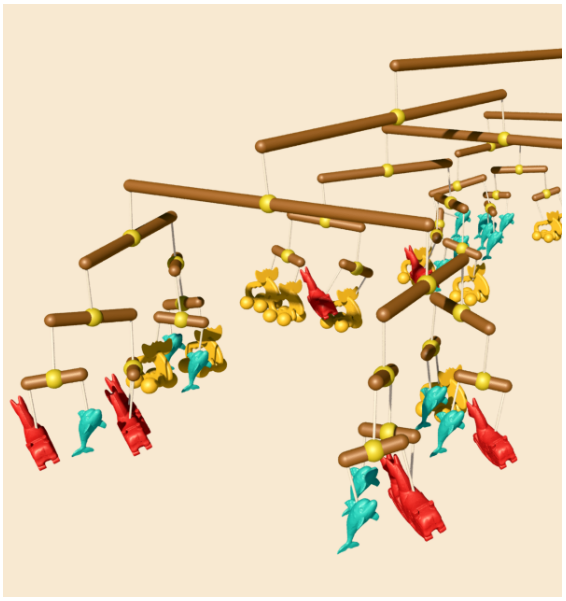

Figure 1: A large-scale mobile simulation involving 127 constraints. A simulation step requires 0.83 ms on average.

1 Introduction

Computer games and virtual reality applications strive for ever greater realism in increasingly complex virtual environments. Such environments typically include a vast number of dynamic deformable and rigid objects. Simulating these objects according to the laws of motion requires sophisticated methods and has a long history in computer graphics. In this context the animation of rigid bodies and their interactions under the influence of constraints such as collisions or joints is perhaps the most researched concern. Numerous physics engines, like Bullet [1] or PhysX [2] have come to public attention. While many methods achieve results of great fidelity, their computational effort can not be justified in real-time applications. The quest for methods that are both robust and fast at the same time is still ongoing.

The position-based dynamics method by Mueller et al. [3] has been applied to particle-based deformable bodies and fluids [4]. It allows for very stable animations at interactive frame-rates. However, it does not yet support rigid body dynamics under the influence of constraints and collisions.

We present a novel approach that extends position-based dynamics to rigid bodies while maintaining its significant properties. Our approach excels in large-scale simulations at interactive frame-rates while being robust under a

vast number of constraints, as depicted in Figure 1. Furthermore, the proposed method is easy to implement, controllable and supports two-way coupling of deformable and rigid bodies.

2 Related Work

The field of rigid body dynamics has a long history in computer animation and may be decomposed into the four subfields integration schemes, collision detection, collision response and constraint resolution. The survey of Bender et al. [5] describes all subfields in more detail, but only the two latter ones are of primary interest in this paper.

In previous work collision handling was performed by the application of forces [6], by solving linear complementary problems [7] or by using impulse-based methods [8, 9, 10]. Müller et al. [3] even introduce a method that resolves collisions by directly modifying positions for the simplified case of individual particles. However, an extension to rigid bodies has not been shown yet. In contrast to that, Kaufman et al. [11] present a method which projects the velocities of the bodies in order to prevent penetrations. Later, Kaufman et al. [12] introduced a staggering approach for frictional contacts. Multi-impact problems were focused by Smith et al. [13]. In order to increase the performance of large-scale simulations, shock propagation methods were introduced [9, 14]. Furthermore, different GPU-based methods were presented to simulate large systems in real-time [15, 16].

To simulate an articulated system with joints, equality constraints are defined for the rigid bodies. A classic approach to solve these constraints in real-time is to use reduced coordinate methods. Featherstone demonstrated that a simulation with reduced coordinates can be performed in linear time for acyclic systems [17, 18]. Redon et al. [6] introduced an adaptive variant of this approach which uses a reduced number of degrees of freedom to improve the performance. Baraff [19] introduced a Lagrange multiplier method which also has a linear-time complexity for acyclic models. Later, Bender [20, 21] demonstrated that the idea of Baraff can also be applied to impulse-based simulation. While Bender proposed to solve a linearized equation based on a prediction of the joint state [22], Weinstein et al. [23] solved the nonlinear equation by a Newton iteration method.

Position-based methods became popular in the last years since they are fast, robust and controllable while no implicit time integration is required. A survey of position-based methods is presented by Bender et al. [24]. Müller et al. [3] introduced position-based dynamics as a generalized framework capable of solving a large variety of constraints. They demonstrated its application on deformable solids [3, 25] and later also on fluids [4]. Diziol et al. [26] introduced a shape matching method with a volume conservation for deformable solids to achieve more realistic results. In order to make shape matching more robust, Müller and Chentanez [25] added an orientation to the particles of their model. This allows them to simulate bodies with solid components and basic joints. Although they treat each particle as a rigid body with position and orientation, they apply the standard position-based constraints that depend solely on particle positions. In constrast, our work incorporates position and orientation in the derivation of constraint equations. This allows us to form constraints between arbitrary points of rigid bodies.

3 Preliminaries

In this section we first start with a brief introduction to the position-based dynamics framework (PBD). Then we continue with the basics of rigid body simulation.

The original position-based dynamics approach [3] is used to simulate a model defined by particles and constraints between these particles. The constraints are solved at the position level by applying correction displacements onto the particle positions. In order to measure the constraint violation, preview positions of the particles are computed in a first step by integrating the particle velocities under the influence of external forces. Then the particle positions are integrated with the new velocities yielding a symplectic Euler scheme. In the second step the system of constraints is solved in a Gauss-Seidel type iteration. The third step updates the particle velocities depending on the difference between

the particle positions at the start of the time step and the new particle positions multiplied by the inverse time step size. In a final step the new velocities are modified to handle friction and damping.

As mentioned before, we want to solve constraints on rigid bodies, which are defined by six parameters. The translational motion parameters are the position $\mathbf{x}$, the velocity $\mathbf{v}$, and the mass m, which rigid bodies have in common with particles. In addition to that, the three rotational parameters are the orientation ϑ, the angular velocity ω, and the inertia tensor $\mathbf{I}$.

These parameters are employed in a symplectic Euler scheme as follows. The equations for velocity and position integration are defined by

$$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + \frac{\mathbf{F}_{external}}{m} h$$
$$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + \mathbf{v}(t_0 + h)h$$

while the equations for the rotational parameters are given by

$$\omega(t_0 + h) = \omega(t_0) + h\mathbf{I}^{-1} \cdot (\tau_{external} - (\omega(t) \times (\mathbf{I} \cdot \omega(t))))$$
$$\mathbf{q}(t_0 + h) = \mathbf{q}(t_0) + \frac{h}{2}\tilde{\omega}(t_0 + h) \cdot \mathbf{q}(t_0),$$

where $\tilde{\omega}$ is the quaternion $[0, \omega_x, \omega_y, \omega_z]$. After the preview positions of the bodies are integrated with the equations above position constraints are solved in several iterations. However, in constrast to the original PBD approach, that updates the velocities after the constraint solver step, we update the velocities of constraints whenever a correction displacement is applied to a rigid body. The required impulse to update the velocities is computed be multiplying the mass weighted displacement with the inverse time step size. By updating the velocities during the position iterations we can apply our friction resolution approach whose details are presented in section 6.

4 Constrained Rigid Bodies

In this section we describe how to extend the PBD solver to solve constraints between rigid bodies. The standard solver works by iteratively handling each constraint on its own by applying displacements to the according particles. The displacements are computed by solving constraints of the following form:

$$C(\mathbf{p} + \Delta\mathbf{p}) = 0, \tag{1}$$

where $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]^T$ is the concatenated vector of particle preview positions and $\mathbf{\Delta p} = [\Delta\mathbf{p}_1, \ldots, \Delta\mathbf{p}_n]^T$ contains the corresponding correction displacements.

In order to solve a constraint function a first-order Taylor approximation

$$C(\mathbf{p}+\Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p} = 0, \tag{2}$$

is used to linearize the constraint. However, this equation is underdetermined. By restricting the direction of the position correction to the gradient direction this problem is solved [3]. The final displacement vector is determined by

$$\Delta\mathbf{p_i} = -\frac{w_i C(\mathbf{p})}{\sum_j w_j \left|\nabla_{\mathbf{p}_j} C(\mathbf{p})\right|^2} \nabla_{\mathbf{p}_i} C(\mathbf{p}), \tag{3}$$

where w_i is the inverse mass of particle $\mathbf{p}_i$.

As an extension to the particle constraint handling of PBD our approach handles constraints between rigid bodies. In contrast to particles an orientation is associated to rigid bodies. Points $\mathbf{p}_i$ that are attached to a rigid body with index j can be described by the following formula

$$\mathbf{p}_i(\mathbf{x}_j, \mathbf{R}(\vartheta_j)) = \mathbf{x}_j + \mathbf{R}(\vartheta_j)\mathbf{r}_i, \tag{4}$$

where $\mathbf{x}_j$ is the center of mass and $\mathbf{R}(\vartheta_j)$ the rotation matrix of the rigid body with index j. The local position of the point i in the body frame is encoded in $\mathbf{r}_i$. Using the definition of body attached points (see Eq. 4) in the Taylor approximation of the constraint (see Eq. 2) yields:

$$C(\mathbf{p}(\mathbf{x} + \Delta\mathbf{x}, \mathbf{R}(\vartheta + \Delta\vartheta))) \approx C(\mathbf{p}(\mathbf{x}, \mathbf{R}(\vartheta))) + \mathbf{J}_C(\mathbf{x}, \vartheta) \cdot [\Delta\mathbf{x}_1^T, \Delta\vartheta_1^T, \ldots, \Delta\mathbf{x}_n^T, \Delta\vartheta_n^T]^T, \tag{5}$$

where $\mathbf{x} = [\mathbf{x}_1^T, \cdots .\mathbf{x}_n^T]^T$ and $\vartheta = [\vartheta_1^T, \cdots, \vartheta_n^T]^T$ are the vectors containing all positions and orientations of the n constrained bodies. Furthermore, the function $\mathbf{p}$ computes the concatenated vector of m positions $\mathbf{p} = [\mathbf{p}_1(\mathbf{x}_1, \mathbf{R}(\vartheta_1))^T, \ldots, \mathbf{p}_m(\mathbf{x}_n, \mathbf{R}(\vartheta_n))^T]^T$ that are constrained by $C(\mathbf{p})$. Due to the fact,

that the entries of $\mathbf{p}$ depend on a function (see Eq. 4), the chain rule has to be applied to the constraint function $C(\mathbf{p})$ to compute its derivative. As a result, the gradient $\nabla_{\mathbf{p}} C(\mathbf{p})$ in Eq. 2 is replaced by the $k \times 6n$ Jacobian $\mathbf{J}_C(\mathbf{x}, \vartheta) = \left(\frac{\partial C}{\partial \mathbf{x}} \frac{\partial C}{\partial \vartheta}\right)$ of the constraint function with respect to the rigid body positions and orientations, where k is the dimension of the codomain of $C(\mathbf{p})$.

Let $\mathbf{M}_j$ be the 6×6 mass matrix of rigid body j with the mass on the first three entries of the diagonal and the moment of inertia tensor $\mathbf{I}_j$ in the lower right 3×3 submatrix. By rearranging Eq. 5 the same way that led from Eq. 2 to Eq. 3 and replacing the inverse particle mass w_j with the inverse mass matrix $\mathbf{M}_j^{-1}$, the formula to compute the vector of rigid body corrections for constraint $C(\mathbf{p})$ is:

$$\begin{aligned} &[\Delta \mathbf{x}_i^T, \Delta \vartheta_i^T, \ldots, \Delta \mathbf{x}_n^T, \Delta \vartheta_n^T]^T = \\ &-\mathbf{M}^{-1} \mathbf{J}_C^T \left[\mathbf{J}_C \mathbf{M}^{-1} \mathbf{J}_C^T\right]^{-1} C(\mathbf{p}), \end{aligned} \tag{6}$$

where the matrix $\mathbf{M}^{-1}\mathbf{J}_C^T$ converts the mass weighted displacement to a displacement in world space. The matrix $\left[\mathbf{J}_C \mathbf{M}^{-1} \mathbf{J}_C^T\right]^{-1}$ is the mass matrix in constraint space.

The question now is how to parameterize the rotations of the rigid bodies to compute a Jacobian $\mathbf{J}_C$ that can be multiplied with the inverse mass matrix $\mathbf{M}_j^{-1}$. Orientations can be parameterized in different ways like Euler angles or quaternions. Accordingly, all these parameterizations lead to a different Jacobian. Nevertheless, a solution can be found by starting at the velocity level. The relationship between angular momentum and angular velocity is $\mathbf{L} = \mathbf{I}\omega$. Taking the first order integration, with the assumption that the axis of rotation stays constant during the time step, and rearranging the result gives $\mathbf{I}^{-1}\mathbf{L}h = \omega h = \vartheta$, where h is the time step size. The vector ϑ represents a rotation of $|\vartheta|$ about the axis $\vartheta / |\vartheta|$ and is known as the exponential map from $\mathbb{R}^3$ to $\mathbb{S}^3$ [27], where $\mathbb{S}^3$ is the space of rotation quaternions . The paper of Grassia [27] also presents the derivation of the rotational part $\partial \mathbf{R}(\vartheta)/\partial \vartheta$ of the Jacobian $\mathbf{J}_C$ which is not repeated here.

5 Joints

Defining translational joint constraints between two rigid bodies is straight forward using the constraint definition of section 4. The translational constraints have in common that the distance of two joint points $\mathbf{a}$ and $\mathbf{b}$, each attached to one of the two bodies A and B, is constrained to be zero. The only difference is the dimension of the constraint space. A simple translational constraint is the ball joint

$$\mathbf{C}(\mathbf{a}, \mathbf{b}) = \mathbf{a} - \mathbf{b} = \mathbf{0}$$

that removes all translational degrees of freedom between the linked bodies. It follows that the joint points and with them the according bodies can not move away from each other. However, the two bodies can freely rotate around the joint position. In order to define a translational constraint that removes only two translational degrees of freedom the constraint space has to be reduced to a plane. The plane is defined by the joint point of one of the two bodies and the plane normal that is also attached to this body. It follows that the joint points have to be projected into the plane to measure the constraint violation. Then, the correction is computed in the two-dimensional constraint space. In a final step the mass weighted correction displacement must be projection back into the three-dimensional space. Similarly, the constraint space of a joint constraining one translational degree of freedom is defined by attaching a unit vector to one of the rigid bodies. Again, the constraint violation is measured by first projecting the joint points onto the constraint space and then computing the distance.

Constraints that remove only rotational degrees of freedom can be defined analogously by constraining the orientation of the linked bodies.

6 Collision Handling

In this section we present our solution to handle collisions, contact, and friction. In the following paragraphs we combine the terms collision and contact under the term proximity and use the special terms only where differences occur.

Proximity Detection We perform only one discrete proximity detection step to ensure a

high performance of our method. The drawback is that, intersections may occur after the position integration or during the position solver iterations. These intersections are corrected in the next time step. It follows that, we can not guarantee an intersection free state at the end of the time step. But, we did not encounter severe artifacts in our examples. A further implication of performing only one collision detection step is, that we need to classify proximity constraints into collisions or contacts by using a threshold velocity. Therefore, we apply the threshold proposed by Mirtich [8].

Intersection resolution Intersections of rigid bodies are handled during the position solver iterations. If an intersection between bodies A and B is detected, a proximity constraint is created. Let $\mathbf{n}$ be the intersection normal defined by the surface normal of body B. In order to solve the intersection, a displacement has to be computed that moves the intersection point $\mathbf{a}$ of body A along the positive direction of the intersection normal and moves the intersection point $\mathbf{b}$ of body B along the negative direction of the intersection normal. The constraint is given by

$$C(\mathbf{a}, \mathbf{b}) = \mathbf{n}^T (\mathbf{a} - \mathbf{b}) \geq 0.$$

It follows that the constraint formula resembles a distance constraint along the intersection normal. The exception to the distance constraint is, that the displacements should be only repulsive. To ensure that the total displacement of the constraint stays repulsive during the solver iterations, every intersection constraint tracks the sum of the already applied displacements. If the sum from the previous steps with the current displacement in iteration step i becomes smaller than zero $\Delta\mathbf{p}_{corrected} = -\sum_{j=1}^{i-1} \Delta\mathbf{p}_j$ corrects the displacement.

A problem that arises when correcting the whole intersection in a single time step is, that objects of the scene might receive an unwanted high velocity change, when deep intersections occur. The intersections either are the result of high body velocities in combination with the discrete collision detection or they are caused by the position corrections in the previous time step. Therefore, we use a stiffness parameter s as introduced in the original PBD approach also for proximity constraints. Furthermore, to avoid even deeper penetrations caused by the correction of other constraints, the required correction displacement is split into two parts. Before performing the position solver iterations the current intersection depth d is computed. In addition d is clamped to negative values including zero $d = min(d, 0)$ to avoid cases when the collision detection finds a collision due to the collision tolerance value while the colliding objects do not intersect. Then, during each iteration step the current constraint violation value c is modified as follows:

$$c_{modified} = min(0, c - d) + s \cdot max(c, d).$$

The first term is unequal to zero if a deeper intersection than the initial intersection is computed. The second term corrects the initial intersection depth. In this equation the max function is used, because c and d are negative values in case of an intersection.

After performing the position correction iterations a shock propagation method similar to the approach of Guendelman [9] corrects the remaining violated proximity constraints.

Friction Besides solving or preventing intersections, a plausible rigid body simulation also has to handle friction and collisions. Coulomb's friction model can easily be expressed in the velocity domain. Therefore, we compute friction effects at the velocity level. Furthermore, we apply a second friction approximation at the position level to correct changes in tangential velocity of objects in proximity caused by position correction constraints.

At the velocity level a Gauss-Seidel type solver iterates over all proximity constraints in five iterations to handle contact, collision and friction constraints. This is done immediately after the velocity integration and before the position integration of rigid bodies. As a result, an object under static friction on an inclined plane will nearly stay at the same position. Without this velocity correction, the object would move under gravity. As a result, the position correction would push the object out of the plane along the plane normal. And finally the uncorrected position change tangential to the inclined plane would cause the object to slide.

The second stage of friction correction takes place during the position correction iterations. Although the position constraint solver works with weighted displacements we apply the friction corrections as impulses to reuse the same friction handling as at the velocity level. In order to compute the friction impulse the current velocity between the objects in contact is needed. As a result of our velocity update immediately after a displacement has been applied in the position iterations, the current tangential velocity is available for each proximity constraint. Then the friction impulse is computed to reduce the tangential velocity. This impulse is corrected to lie in the friction cone before being applied to the rigid bodies. It follows that the impulse along the intersection normal is required to correct the friction impulse. Therefore, we sum up the mass weighted displacements of the proximity constraints during the position correction step. Then the mass weighted displacement is multiplied with the inverse time step size. Thus, an approximation for the impulse in the direction of the intersection normal is computed. This impulse is used in conjunction with the friction law to correct the friction impulse.

7 Two-way Coupling

A main advantage of integrating rigid body simulation in the PBD framework is to gain stable two-way coupling between rigid bodies and deformable models simulated with particles. The alternative of using for example an impulse based framework for the rigid bodies and interleaving the simulation with a PBD simulation may lead to stability issues. The problem with interleaving is, that both simulations do not "see" constraints induced by the other framework.

With the extension of PBD constraints to handle rigid bodies like in section 4 it is very easy to couple rigid bodies and particles with a single constraint by using the concept of connectors introduced by Witkin et al. [28]. The idea is to split the Jacobian $\mathbf{J}_C$ into a constraint and a connector part. For a rigid body the splitting is

$$\mathbf{J}_C = \frac{\partial C(\mathbf{p}(\mathbf{x}, \mathbf{R}(\mathbf{v})))}{\partial \mathbf{p}(\mathbf{x}, \mathbf{R}(\mathbf{v}))} \cdot \left(\frac{\partial \mathbf{p}(\mathbf{x}, \mathbf{R}(\mathbf{v}))}{\partial \mathbf{x}} \frac{\partial \mathbf{p}(\mathbf{x}, \mathbf{R}(\mathbf{v}))}{\partial \mathbf{v}} \right)^T,$$

where the first factor is the constraint Jacobian with respect to the constrained points and the second factor is the rigid body connector Jacobian. Besides rigid bodies also particles can be handled by Eq. 4. Therefore, the Jacobian has to be computed with respect to the position parameter $\mathbf{x}$ of the particle. As a result the splitting is

$$\mathbf{J}_C = \frac{\partial C(\mathbf{p})}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \mathbf{x}} = \frac{\partial C(\mathbf{p})}{\partial \mathbf{p}}.$$

The particle connector Jacobian $\partial \mathbf{p} / \partial \mathbf{x}$ is the identity matrix. Furthermore, the mass matrix of the rigid body has to be exchanged with the 3×3 particle mass matrix containing the particle mass on each of the diagonal elements. To sum up, by using the concept of connectors, points defined on rigid bodies and points defined by particles can be simultaneously used in position constraints. Consequently a full two-way coupling is achieved.

8 Results

We tested our algorithm on a total of four examples that are shown in the accompanying video:

1. The mobile example in Figure 1 demonstrates the exceptional stability of a system under a large number of constraints. Figure 4 shows that the average computation times scale linearly with the number of bodies.

2. A cloth example, exemplifying the two-way interaction between deformable and rigid bodies (see Figure 3). A number of tori deform the cloth by dropping on it. They react on the cloth's response, but settle in a steady and jitter-free state after a few seconds.

3. A pile example (see Figure 2) involving a massive number of collisions between two types of dropped rigid bodies with each

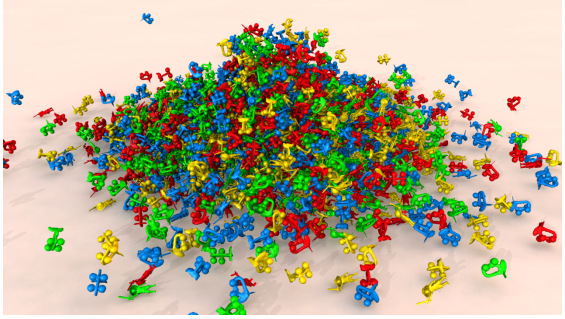

Figure 2: Massive collision scene with 2000 rigid bodies stacking up on the ground.

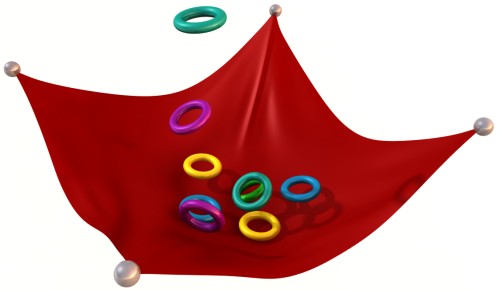

Figure 3: Scene demonstrating the two-way interaction of rigid and deformable bodies. A number of tori are falling onto a piece of cloth, thereby deforming it.

other, the ground plane and a few surrounding poles. The system settles in an almost steady state with only small movements. No visually noticeable jumps and irregularities are observed after reaching this state.

4. An elk aggressively collides with a duck that blocks its way. The scene demonstrates interaction between rigid bodies and deformable cloth balloons [3].

For the four examples we measured the average computation times over 10 s of simulation time on an Intel Core i7 3820 CPU with a clock rate of 3.8 GHz. All times were measured running on a single core. Collision Detection timings are excluded from the measurements. We used a constant time step size of $h = 0.01$ s, and a maximum number of five position and five velocity iterations for the first three examples. The simulation of the cloth balloon in example four required to increase the position iteration count to 150. This high iteration count is required to get a stable deformable model and does not depend on the coupling.

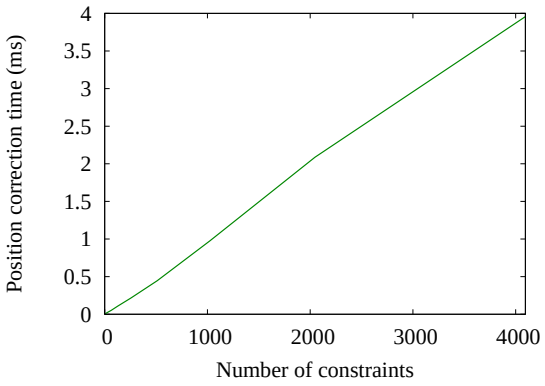

Figure 4: The time in milliseconds required for the position correction step of the mobile example in Figure 1 for a varying number of constraints.

Most notably the pile model (see Figure 2) robustly handles about 4000 contacts without jittering, which clearly demonstrates the stability of our method. In terms of performance our method required a computation time of 17 ms on average to resolve the 4000 contacts. The cloth model with 1681 particles and ten tori needs about 7.13 ms and the mobile model with 127 constraints needs about 0.17 ms.

For the mobile example we varied the number of constraints between 3 and 4095 and measured the time required on the position correction step. As expected, this time scaled linearly with respect to the number of constraints in the scene, as depicted in Figure 4. Moreover, our method is about 2.5 times faster than real time even for a system with 4095 constraints.

9 Conclusion

We have presented a method for the simulation of rigid bodies that is fast and behaves robust under a large number of constraints. We have demonstrated its practical application in four complex scenes involving large-scale simulations and two-way coupling. Like all position-based methods, it is easy to implement and controllable from a user's point of view.

The method not only offers the advantages described above, but fits nicely into the already existing position-based dynamics framework. Be-

Figure 5: Interaction between a toy elk and a cloth balloon duck. The elk model consists of five rigid bodies representing the body of the elk and the four wheels. The collision between the elk and the duck is simulated by our two-way coupling.

sides the demonstrated two-way coupling of cloth and rigid bodies, a fluid-solid interaction is feasible and subject of our future work.

The simulation has been implemented exclusively for single-core processing so far. A multi-core or GPU implementation is an advantageous goal for the future and has in principal already been solved for force-based methods. Its application to position-based rigid bodies still has to be evaluated. A few other components such as motors are still missing in the position-based rigid body framework. However, their implementation should be feasible and straightforward.

10 Acknowledgments

This work was supported by the Excellence Initiative of the German Federal and State Governments and the Graduate School of Excellence Computational Engineering at Technische Universität Darmstadt. The authors would like to thank Daniel Thul who investigated the applicability of our basic idea in his bachelor thesis. Furthermore, we acknowledge the source of the models shown in our examples. The elk model used in scenes one, three and four is provided courtesy of MPII by the AIM@SHAPE Shape Repository. The dolphin model and the isidore horse used in scene one is provided courtesy of INRIA by the AIM@SHAPE Shape Repository. Additionally, scene 3 uses the model 'Adventure Kid' by Clint Bellanger available at `http://opengameart.org/content/adventure-kid` under a Creative Commons Attribution 3.0 Unported. Full terms at `http://creativecommons.org/licenses/by/3.0/`. Furthermore, a modified version of the duck model 'Rubberduck' by rubberduck available at `http://opengameart.org/content/rubberduck` under a CC0 1.0 Universal Public Domain Dedication is used in scene four.

References

[1] Erwin Coumans. The bullet physics library. `http://www.bulletphysics.org`, February 2014.

[2] NVIDIA. PhysX. `http://developer.nvidia.com/physx`, February 2014.

[3] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, April 2007.

[4] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12, July 2013.

[5] Jan Bender, Kenny Erleben, and Jeff Trinkle. Interactive simulation of rigid body dynamics in computer graphics. *Computer Graphics Forum*, 33(1):246–270, 2014.

[6] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. *ACM Trans. Graph.*, 24:936–945, July 2005.

[7] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. SIGGRAPH*, 1994.

[8] Brian V. Mirtich and John F. Canny. Impulse-based simulation of rigid bodies. In *Proc. Interactive 3D graphics*, pages 181–ff. ACM Press, 1995.

[9] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 2003.

[10] Jan Bender and Alfred Schmitt. Constraint-based collision and contact handling using impulses. In *Proc. Computer Animation and Social Agents*, pages 3–11, 2006.

[11] Danny M. Kaufman, Timothy Edmunds, and Dinesh K. Pai. Fast frictional dynamics for rigid bodies. *ACM Trans. Graph.*, 24(3):946–956, 2005.

[12] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.*, 27(5), 2008.

[13] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Reflections on simultaneous impact. *ACM Trans. Graph.*, 31(4):106:1–106:12, July 2012.

[14] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2):12, 2007.

[15] A. Tasora, D. Negrut, and M. Anitescu. Large-scale Parallel Multi-body Dynamics with Frictional Contact on the Graphical Processing Unit. In *Proc. of Institution of Mech. Eng., Part K: Journal of Multi-body Dynamics*, pages 315–326, 2008.

[16] Daniel Weber, Jan Bender, Markus Schnoes, André Stork, and Dieter Fellner. Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum*, 32(1):16–26, 2013.

[17] Roy Featherstone and David Orin. Robot dynamics: Equations and algorithms. *International Conference on Robotics and Automation*, pages 826–834, 2000.

[18] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, USA, 2007.

[19] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proc. SIGGRAPH*, pages 137–146. ACM Press, 1996.

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

[21] Jan Bender. *Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität*. PhD thesis, University of Karlsruhe, Germany, 2007.

[22] Jan Bender, Dieter Finkenzeller, and Alfred Schmitt. An impulse-based dynamic simulation system for VR applications. In *Proc. Virtual Concept*. Springer, 2005.

[23] Rachel Weinstein, Joseph Teran, and Ron Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE TVCG*, 12(3):365–374, 2006.

[24] Jan Bender, Matthias Müller, Miguel A. Otaduy, and Matthias Teschner. Position-based methods for the simulation of solid objects in computer graphics. In *EUROGRAPHICS State of the Art Reports*. Eurographics Association, 2013.

[25] Matthias Müller and Nuttapong Chentanez. Solid simulation with oriented particles. *ACM Trans. Graph.*, 30(4):92:1–92:10, July 2011.

[26] Raphael Diziol, Jan Bender, and Daniel Bayer. Robust real-time deformation of incompressible surface meshes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2011.

[27] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3:29–48, 1998.

[28] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. In *Proc. Interactive 3D Graphics*, pages 11–21. ACM, 1990.