Long Range Constraints for Rigid Body Simulations

Matthias Müller
NVIDIA

Nuttapong Chentanez
NVIDIA

Miles Macklin
NVIDIA

Stefan Jeschke
NVIDIA

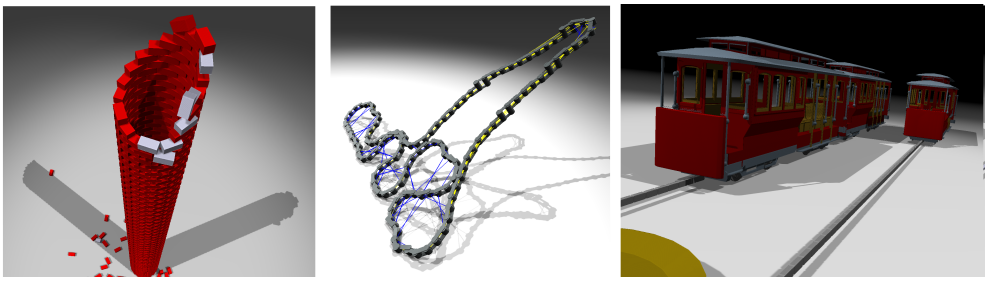


Figure 1: Long range constraints propagate errors through large stacks or long chains instantaneously and make it possible to simulate the depicted scenes with a small number of solver iterations.

ABSTRACT

The two main constraints used in rigid body simulations are contacts and joints. Both constrain the motion of a small number of bodies in close proximity. However, it is often the case that a series of constraints restrict the motion of objects over longer distances such as the contacts in a large pile or the joints in a chain of rigid bodies. When only short range constraints are considered, a large number of solver iterations is typically needed for long range effects to emerge because information has to be propagated through individual joints and contacts.

Our basic idea to significantly speed up this process is to analyze the contact or joint graphs and automatically derive long range constraints such as upper and lower distance bounds between bodies that can potentially be far apart both spatially and topologically. The long range constraints are either generated or updated at every time step in case of contacts or whenever their topology changes within a joint graph. The significant increase of the convergence rate due to the use of long range constraints allows us to simulate scenarios that cannot be handled by traditional solvers with a number of solver iterations that allow real time simulation.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**;

KEYWORDS

rigid body simulation, position based simulation, long range constraints

ACM Reference format:
Matthias Müller, Nuttapong Chentanez, Miles Macklin, and Stefan Jeschke. 2017. Long Range Constraints for Rigid Body Simulations. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 10 pages.
DOI: 10.1145/3099564.3099574

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SCA '17, Los Angeles, CA, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-5091-4/17/07…$15.00
DOI: 10.1145/3099564.3099574

1 INTRODUCTION

Rigid body simulation is an important field in robotics where it allows the study of systems virtually before they are built. Biomechanics is another research area in which rigid body simulations play a key role for modelling skeletal dynamics. In both cases, the number of simulated bodies is relatively small and accuracy is essential. This is in contrast to physically based animation in computer graphics, specifically in films and computer games where scenes can be composed of thousands of individual bodies. Here the emphasis lies on robustness and simplicity rather than accuracy. As a result, most physics engines use linear, velocity based projected Gauss-Seidel or Jacobi methods to solve the underlying linear complementarity problems.

It is a well known fact that iterative local solvers converge slowly which is problematic in the case of tall piles or long chains of jointed bodies. A variety of tricks has been proposed to alleviate this problem. An example is the so called "shock propagation method" [Guendelman et al. 2003] for the stable simulation of stacking. The basic idea is to solve the bodies in layers from bottom to top. Each solved layer is frozen by assigning infinite mass to the respective bodies. This method is effective but not physically correct and can yield noticeable visual artifacts.

In this paper we propose the concept of long range constraints that can be derived automatically from a series of local constraints,

specifically from joints or contacts. The two cases are treated similarly in that the constraints are derived from a graph. They differ in the way the graphs are generated. For joints, the graph is provided by the user and only changes when joints are added, removed or when joint properties changed. The contact graph is derived or updated at each time step from the generated contacts and their properties. We use a contact graph that is dual to the traditional version. Instead of interpreting bodies as nodes and contacts as edges, our nodes represent contacts and edges connect pairs of contacts that affect the same body.

We propose different types of long range constraints. Of the first type are unilateral lower and upper distance bounds between pairs of bodies that can potentially be far apart, both spatially and topologically. These can be interpreted as generalizations of Long Range Attachment (LRA) [Kim et al. 2012] for rigid body systems. The second type is a projection of joints into a plane for chains of bodies connected by hinge joints. Example applications of this type of constraint are tracks of tanks or excavators or robot arms. The third type is a shape matching constraint that can be used to simulate fracture with pre-fractured models. The fourth type of long range constraint restricts the direct distance to the ground of contact points and bodies at any height in stacks.

Our main contributions are

- The concept of long range constraints in rigid body simulations.
- Generalized unilateral long range distance constraints and their derivation from a joint graph taking potential angle limits into account.
- A global plane projection to satisfy hinge joint constraints in long chains.
- Generalized shape matching for handling graphs of fixed joints.
- An automatic derivation of long range distance constraints from the dual contact graph with contacts as nodes and bodies as links.

2 RELATED WORK

There is a large body of work on the simulation of rigid bodies from various research fields such as mechanics [Amirouche 2006], robotics [Siciliano and Khatib 2007] and biomechanics [Delp et al. 2007]. Armstrong and Green [1985], Hahn [1988] and Baraff [1989] were among the first to introduce rigid bodies to computer graphics. The most popular approach to rigid body simulation are impulse based methods [Mirtich 1996], [Eberly 2010] but a variety of others possibilities have been proposed. Discussing the literature on rigid body simulations in computer graphics is beyond the scope of our paper. Instead we refer the reader to the recent survey of [Bender et al. 2014]. Also, our approach is orthogonal to the choice of the underlying rigid body solver. Long range constraints can potentially be added to any existing rigid body engine. However, since the constraints we will discuss are all position based, they are most naturally integrated with a position based solver.

The position based approach was introduced to computer graphics by Jakobsen [2001] to solve a set of distance constraints using a Verlet integrator. Müller et al. [2006] generalize the method to handle arbitrary position based constraints such as bending and volume conservation. They store velocities explicitly, which makes damping and friction handling easier. The paper also introduced the term Position Based Dynamic (PBD) to refer to this method. Two main papers exist on the simulation of rigid bodies in the PBD framework. Macklin et al. [2014] used a collection of particles bound by shape matching constraints [Müller et al. 2005] to represent rigid objects. In contrast, Deul et al. [2014] added the handling of rotational degrees of freedoms to PBD which allowed them to simulate rigid bodies as single six degrees of freedom entities. They also discussed ways to simulate the two way interaction with deformable objects. A thorough overview of PBD methods can be found in [Bender et al. 2015]. While being stable for large time steps and versatile, the original PBD approach suffered from stiffness being time step and iteration count dependent. Macklin et al. [2016] introduced the XPBD method to addresses this problem. They store and sum up each constraint's scalar Lagrange multiplier which allows one to specify the compliance (true time step independent inverse stiffness) of each constraint. We employ XPBD in our work.

As mentioned before, we address the problem of slow convergence of iterative solvers in the presence of a large number of constraints such as in large stacks or long chains. To improve the convergence and stability of stacks, Hahn et al. [1988] construct a contact graph which approximately identifies layers of bodies resting on top of other. From this graph they derive a bottom-up order in which the contacts are solved. Shock propagation is based on a similar idea and was introduced in [Guendelman et al. 2003]. Here, the contacts are also solved in layers from bottom to top. However, for each layer, all bodies below the current layer are treated as having infinite mass. This method effectively prevents bodies from sinking into each other but can, as the authors mention, create noticeable artifacts such as unstable stacks being steady. Therefore, they recommend to run enough traditional solver iterations before applying their method. Erleben [2007] utilizes shock propagation in velocity-based rigid body simulation. Tsuda [2009] proposes to increase the mass of the already solved bodies instead of making it infinite which reduces the problems with unstable stacks. The shock propagation method is inherently non parallel. To solve this problem, Macklin et al. [2014] do not specify the order in which contacts are solved but scale the masses of objects in stacks dependent on their height above the ground to improve convergence speed. Another idea used in rigid body engines to stabilize stacking is to reduce the gravitational force inside piles of objects. All these methods with the exception of only modifying the solution order, are non-physical and can potentially introduce visual artifacts.

A number of methods have been proposed to accelerate the simulation of jointed objects like robot arms, ropes, cables and hair strands. One idea is to use generalized coordinates which always satisfy the positional constraints of joints and therefore, make stretching impossible. Probably the most popular approach in this field is Featherstone's algorithm [Featherstone 1987]. The method is not as popular in graphics as in other fields, however, because it is not applicable to circular structure and is more complicated, less parallel and less stable than other methods. Baraff [1996] proposed a constraint solver that takes advantage of the sparse structure of the system matrix arising from a hierarchical articulation. In contrast to Featherstone's method, it operates in maximal coordinates rather

than reduced coordinates, and can be extended to closed loops with some additional cost and complexity. Unlike their method, our solver does not assume any particular structure in the constraint set which makes it more general and simplifies parallelization and the integration into existing solvers. Tomcin et al. [Tomcin et al. 2014] proposed techniques to efficiently solve closed loop articulation constraints. In contrast to our approach, they do not introduce new constraints but improve the structure of the given system matrix with techniques like regularization and body splitting.

Applying the shock propagation idea to strands yields the the Follow The Leader (FTL) approach in which the particles are handled from head to tail, always assuming the upstream particle to have infinite mass. This method has been used for quasi-static simulations of knot-tying in [Brown et al. 2004]. Similar to shock propagation, this method is non-physical, a fact that does not show in the quasi static case but becomes apparent in dynamic simulations as discussed in [Müller et al. 2012]. To make fur and hair inexensible, Kim et al. [2012] introduced Long Range Attachments (LRA). Instead of propagating information along adjacent particles/bodies, they attach each particle directly to the attachment point. Since the attachment point on a character is kinematic, assigning infinite mass to it is physically correct. Therefore the method does not introduce artifacts and significantly reduces the number of required solver iterations. Our first type of long range constraint is a generalization of this idea. For circular structures our method is more closely related to hierarchical PBD [Müller 2008]. Finally Sueda et al. [2011] proposed a method to simulate long strands in highly constrained scenarios effectively by introducing a number of specialized constraints. However, these have to be setup by hand for the various objects in the scene which makes the method less practical than other approaches in general scenarios.

3 LONG RANGE CONSTRAINTS

We define long range constraints as constraints that restrict the mutual motion of pairs or groups of bodies not immediately connected via joints or contacts. Our goal is to derive long range constraints automatically from a set of local constraints. We propose four types of long range constraints which we will describe in the following sections. Of course, these do not cover all possibilities. From our basic idea, many more long range constraint types can be devised and we hope to stimulate further research in this direction.

3.1 Long Range Attachments

Our first type of long range constraint can be viewed as a generalization of long range attachments (LRA) [Kim et al. 2012]. The traditional LRA are upper distance bounds for particles attached via a series of edges to a kinematic attachment point. In the case of a hair strand, each particle in the strand is constrained to remain in the sphere with radius r around the attachment point on the head, where r is the sum of all the edge lengths between the particle and the attachment point. These additional long range constraints let the hair look inextensible with very few iterations.

To generalize this idea to rigid bodies, we replace the particle chain by a chain of jointed rigid bodies which is attached at one end to a kinematic object. As Figure 2 shows, the particles are now replaced by joint locations (red dots) on the bodies and the LRA edges by the connections between joint locations within one body (blue segments). We assume that the joints have no positional degrees of freedom, which is true for the two most important joint types, namely hinge and spherical joints. Since the yellow body at the top is kinematic and its joint location fixed, we can create upper distance constraints from all joints directly to the attachment point (red lines).

Figure 2(a) shows a chain in its rest configuration. When released under gravity, it stretches and reaches the equilibrium configuration shown in 2(b). It is easy to see that the joints (red dots) need to stay within the dotted circles at all times. This results in uni-lateral long range constraints. Handling unilateral constraints in Gauss-Seidel or Jacobi solvers is straightforward. In each iteration of our iterative solver, we project only the joints that are outside their respective spheres on to the closest point on the sphere.

We generalize the LRA and also take angular joint limits into account – angle limits for hinge joints and swing limits for spherical joints. In addition to tighter upper bounds, this also yields lower distance bounds, a concept that is not discussed in [Kim et al. 2012]. Figure 2(c) shows the chain with 90 degree joint limits. Now, the chain cannot be fully stretched restricting the joints to remain in smaller spheres. Instead of summing up the edge lengths to compute the distance bounds, we have to take the maximal opening angles into consideration as shown in Figure 2(d). These angle bounds are the sum of the angle γ between the segments in the rest configuration (Figure 2(a)) and the upper angular joint limit δ.

The chain shown in Figure 2(e) shows a configuration, in which both upper and lower distance bounds can be derived. Here, the rest angles γ are 180 degrees. In this case, the upper joint limits δ restrict the overall bending. The chain is only allowed to be between the configurations (e) and (f). This fact restricts the joints to remain inside the gray bands.

Figure 3 shows how we derive lower and upper distance bounds to the attachment points while taking angular joint limits into account. We describe how the lower bound distance d_{i+1} from joint $i+1$ to the attachment point is computed based on the bound d_i for joint i and the segment length l_i between the joints. For a lower bound, we have to make d_{i+1} as small as possible. In addition to d_i and l_i, we also need α_i (light blue), the angle of the last segment w.r.t. the direct line to the attachment point. To make d_i as small as possible we have to turn l_i as much as possible towards the attachment point. We are allowed to rotate it at most by the angle $\gamma_i + \delta_i$, as described in Figure 2. The minimum angle between d_i and l_i is therefore $\max(\alpha_i - \gamma_i - \delta_i, 0)$ (dark blue). Knowing this angle allows us to compute d_{i+1}. The next angle α_{i+1} turns out to be $\alpha_{i+1} = \max(\alpha_i - \gamma_i - \delta_i, 0) + \beta$, where β is the angle between d_i and d_{i+1}. The same method can also be used to compute an upper bound distance. Now we try to bend l_i as much as possible away from the attachment which is achieved by the angle $\min(\alpha_i + \gamma_i + \delta_i, \pi)$. Note that this algorithm not only works for simple chains but for branching structures as well.

3.2 Free chains

There are many use cases in which chains or cables are not attached to static or kinematic objects. An example is the track of a tank or an excavator depicted in Figure 4(a). So far we have asymmetrically

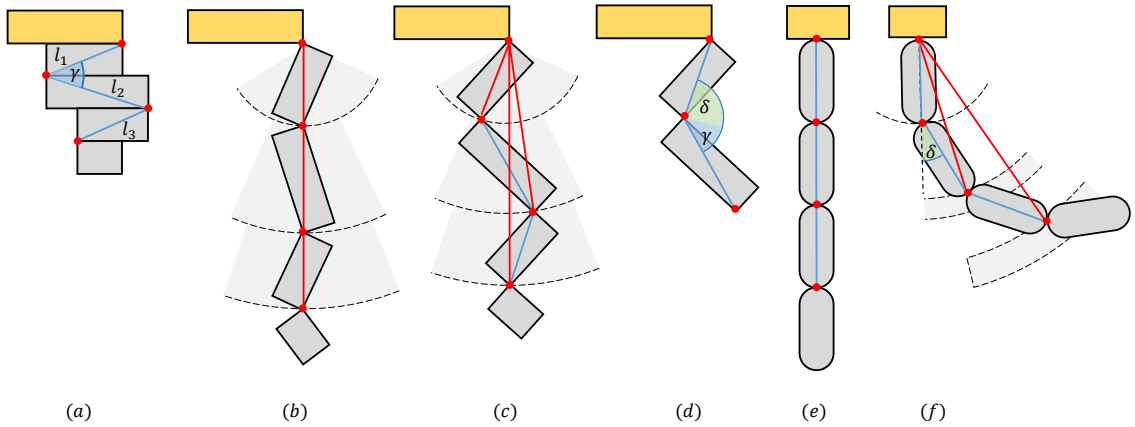


Figure 2: Chains of rigid bodies (gray) attached to a kinematic object (yellow) and connected via joints (red dots). (a): The chain in the rest configuration. (b): Released under gravity without joint limits. Upper distance bounds can be computed from each joint directly to the attachment location. (c): Taking 90 degree upper angular joint limits into account yields tighter bounds. (d): The maximal opening angle between segments (blue) is the sum of the rest angle γ and the joint angle limit δ. (e): A chain with a stretched rest configuration. (f): With angle limits δ, both lower and upper distance constraints can be derived.

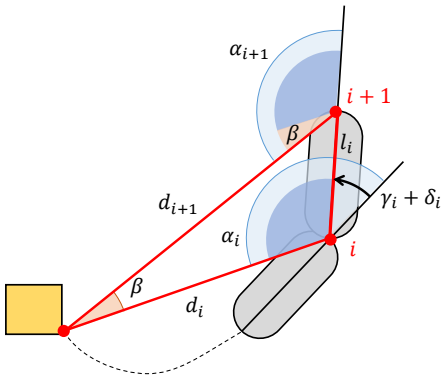


Figure 3: Computation of the lower distance bound d_{i+1} from joint $i+1$ to the attachment point given the lower distance bound d_i of joint i and the segment length l_i. In addition to the previous distance d_i, the angle α_i of the last segment w.r.t. the attachment point is needed to find the new bound.

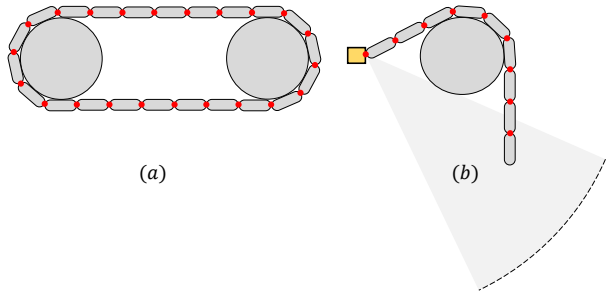


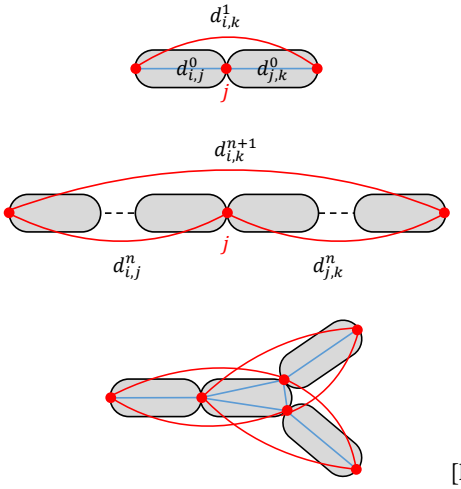
Figure 4: Scenarios in which long range attachments are not applicable (a) or not effective (b).

moved the joints towards the attachment point to satisfy the upper and lower distance bounds. This is not possible for free chains. There are also other situations in which LRA constraints are not effective, even with a fixed attachment point as shown in Figure 4(b). Here we drew the upper bound constraint for the last joint. Keeping joint within this sphere does not prevent stretching of the second part of the chain.

To solve both problems we derive long range constraints hierarchically from the local joints in such cases as shown in Figure 5. Level 0 constraints are the local segments connecting all pairs of joints within a given body shown in blue. We denote as $d^0_{i,j}$ the local segment between joint i and joint j on a given body. Layer 0 and the segments in blue are shown in the top row of Figure 5.

We do not generate constraints on this level because the segments connect joints adjacent to the same body and therefore keep their lengths automatically. To create level $n+1$ constraints based on level n constraints we proceed as follows. For each joint j we create one level $n+1$ constraint for all pairs of level n constraints that are adjacent to this joint. In the first case of Figure 5, two level 0 constraints are adjacent to joint j, and therefore, one level 1 constraint is created at joint j by combining the two segments. In the simple case of a linear chain, at most one level $n+1$ constraint is created at each joint because there can be at most two adjacent level n constraints as shown in the second row of Figure 5. However, our method also works for general graphs as depicted in the third row.

Our method yields the hierarchy shown in the top two rows of Figure 6 for linear chains. It contains $O(n \log n)$ constraints. Another option would be to create higher-level constraints only for increasingly smaller subsets of the joints in a multigrid fashion as shown in the bottom two rows of Figure 6 yielding $O(n)$ constraints. The reason we use the dense hierarchy is that the sparse variant yields visual artifacts because not all joints are solved to the same accuracy with a fixed number of solver iterations. For instance, if



[h]

Figure 5: Construction of a hierarchy of long range distance constraints. Top: Layer 0 contains segments between joints adjacent to the same body (blue). For each pair of level n constraints adjacent to the same joint, one new level $n+1$ constraint is generated by combining the two. Middle: The procedure applied to a linear chain. Bottom: The procedure also works for arbitrary graphs.

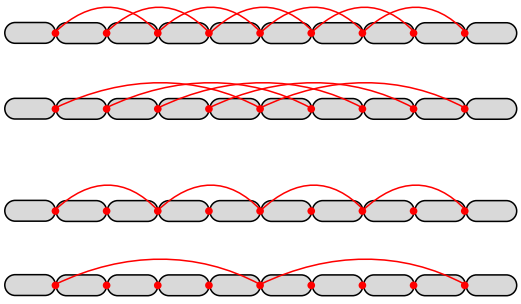


Figure 6: First and second row: The resulting hierarchy on a linear chain. $O(n \log n)$ long range constraints are generated. Third and fourth row: Reducing the joint set in higher levels produces $O(n)$ constraints but can yield visual artifacts.

the chain is picked up at a joint which is present in a higher level, it behaves differently than if picked joint only existed in the first layer. The higher number of constraints is not problematic in our examples because we only use a fixed number of layers, typically less than five.

The remaining question is how to combine two long range constraints. The procedure to do this is illustrated in Figure 7 and is similar to the incremental algorithm for long range attachments shown in Figure 3. As in the incremental case, we need to store the angles α_1 and α_2 of the end segments w.r.t. the distance constraint. Figure 7 shows how a lower bound constraint is constructed. Here, d_l and d_r denote the lower bound distances for the left and right constraints. At the center joint, the smallest angle between the adjacent segments is the rest angle γ minus the joint limit δ or

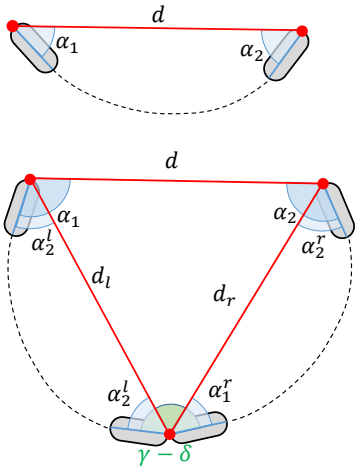


Figure 7: Computation of the lower distance bound d given the lower distance bounds d_l and d_r of the two constraints that are combined. Top: As in the attachment case, the angles α_1 and α_2 between the outermost segments w.r.t. direction of the distance bound have to be stored. Bottom: The angle limit on the center joint limits the angle between the adjacent segments. The knowledge of the extremal center angle and the boundary angles and distance bounds of the adjacent constraints allows the computation of all other quantities.

zero if the difference is smaller than zero as in the incremental case. Knowing this angle, d_l and d_r, we can compute the combined bound d. With all three distances of the red triangle, we can then compute the upper left and right angles of the triangle and the end angles α_1 and α_2 of the combined joint. For upper bounds, we make the angle between the center segments as large as possible.

3.3 Soft constraints

So far we have considered hard limits only. A constraint that prevents a structure from overstretching must be hard. There are other cases, however, in which softening the constraints creates new useful effects. For instance, applying softened lower bounds from bending limits creates bending resistance. In Figure 16 we used the length of the upper bound constraint as a soft lower bound with the effect that the bow is forced to be fully stretched and fires the arrow. Here the computation of the upper bound yields the information necessary to simulate the cross bow although the artist modeled it in a taut state.

3.4 Hinge chains

For a further type of long range constraint, we consider a special type of chain, namely a sequence of bodies connected by hinge joints with parallel axes as depicted in Figure 8. Examples for such chains are the tracks of tanks and excavators, bicycle chain, or parts of robot arms. In this case, all joints are restricted to remain in a common plane which is a long range positional constraint. In addition, all joint axes need to be parallel which yields a long range rotational constraint. If one end of the chain is attached

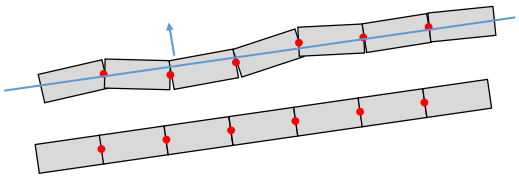


Figure 8: If a chain is linked by hinge joints with parallel axes, all joints must lie in a common plane (blue line) and all axes must be parallel at all times which are long range constraints. Due to the principle of least action, this plane best fits the joint locations in a mass weighted least squares sense.

kinematically, the plane is fixed in space and all the joints can simply be projected into this particular plane. If the chain is free, however, the plane itself is free to move and we have to find it at every time step as the plane for which the projection step does not introduce linear or angular momentum. Due to the principle of least action, the plane we are looking for is the plane that is closest to the joint locations in a mass-weighted sense or in other words, the plane for which the projection step performs the least amount of work.

We derive it using the method of orthogonal multivariate regression and first compute the mass weighted co-variance matrix

$$A = \sum_i (\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T m_i, \tag{1}$$

using the center of mass $\mathbf{x} = \sum_i \mathbf{x}_i m_i$. The $\mathbf{x}_i$ are the joint locations – two per body – and m_i the masses of the corresponding bodies. The closest plane's normal is parallel to the Eigenvector of A that corresponds to the smallest Eigenvalue and the plane itself passes through the center of mass. All joints are then projected into this plane and all joint axes are aligned with the plane normal by applying the necessary positional corrections.

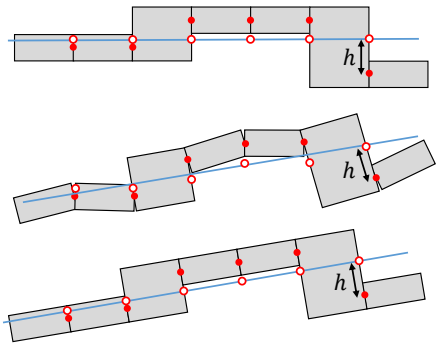


Figure 9: If the joint locations are not in a plane in the rest configuration, we find the best fit plane whose normal is parallel to the hinge axes and store the individual offsets h. At run time, we fit the plane to the offset joint locations.

Projecting the joints into the common plane is only correct if all joints are in a plane in the rest configuration. A chain in which this is not the case is shown in Figure 9. Again, if this case is not handled in the physically correct way, the chain immediately starts to rotate and flies to infinity as was the case for us until we found the correct solution which works as follows: As a pre-computation step, we determine the plane perpendicular to the joint axes and through the center of mass of the joint locations in the rest configuration. Here the mass of a joint corresponds to the sum of the masses of the adjacent bodies. We also pre-compute the distances h of the joints to this plane. At run time, we displace the joint locations along the current joint axes by the distance h resulting in the white dots in Figure 9. We then fit the plane through these displaced points. Finally, it is important to apply the positional corrections to the bodies at the displaced locations.

3.5 Generalized shape matching

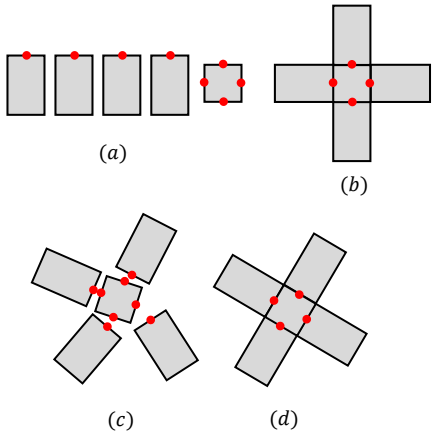


Figure 10: (a) A set of rigid bodies that are connected via fixed joints that fully restrict the relative orientation of two neighboring bodies. (b) A conforming rest state satisfying all joint restrictions. (c) The current poses of the bodies. (d) The shape match constraint matches the conforming rest state into the current state.

If a set of rigid bodies is connected via a graph of fixed joints that fully restrict the relative orientation of two bodies, the entire set is only allowed to move as a rigid union. Figure 17 (left) shows a vase that is pre-fractured into pieces. These pieces are held together with fixed joints. With 10 solver iterations the vase behaves like a soft body. With 100 iterations, it becomes stiffer but still looks deformable.

To make a rigid union completely stiff we propose a shape matching constraint which is a generalization of the shape matching constraint proposed in [Müller et al. 2005] for point particles. The basic idea is to match the current configuration of the bodies with a configuration that satisfies all fixed joint constraints. A simpler way to make a rigid union completely stiff would be to simulate it as one rigid body. However, working with a shape matching constraint allows us to simulate fracture. To do this we run the solver for a number of iterations on the individual bodies and joints. During this phase, if a joint force exceeds a threshold, we delete

the corresponding joint. After joint deletion, we re-compute the rigid unions by flooding the joint graph. Finally we apply the shape match constraint to the rigid unions.

Figure 10 illustrates this process in more detail. Figure 10(a) shows a set of rigid bodies in their rest state. As a first step we flood the graph defined by the bodies and all fixed joints. Each island becomes a shape matching constraint. If the bodies do not satisfy the fixed joint constraints in the rest state, we first have to compute a conforming rest state as shown in Figure 10(b). If the joints are consistent, such a configuration exists and is unique up to a rigid motion of the entire group. We are free to choose any such motion because it will be cancelled out in the shape matching process. During the simulation after an unconstrained integration of the individual bodies, the fixed joint constraints are violated as Figure 10(c) shows. To solve all constraints in one step in a momentum conserving way we match the conforming rest state with the current state and use the matched poses for the rigid bodies shown in Figure 10(d).

To find the momentum conserving transformation, the original shape matching method for point masses proceeds as follows: For a set of particles with current positions $\mathbf{p}_i$, rest positions $\bar{\mathbf{p}}_i$ and masses m_i first compute the center of mass $\mathbf{p}$ of the current positions and the center of mass $\bar{\mathbf{p}}$ of the rest positions. Next compute the moment matrix $\mathbf{A} = \sum_i m_i \mathbf{r}_i \bar{\mathbf{r}}_i^T$, where $\mathbf{r}_i = \mathbf{p}_i - \mathbf{p}$ and $\bar{\mathbf{r}}_i = \bar{\mathbf{p}}_i - \bar{\mathbf{p}}$. The optimal transformation of the rest positions into the current positions is then given by the translation $\mathbf{p} - \bar{\mathbf{p}}$ and the rotational part of $\mathbf{A}$ given by the polar decomposition.

We now have to replace the point masses by rigid bodies. For the translational part we can simply replace the bodies by point masses at their individual center of mass and use the conforming rest state instead of the original rest state. The computation of the moment matrix $\mathbf{A}$ becomes slightly more involved. For an individual rigid body i whose current rotation from the conforming rest state to the current state is given by the matrix $\mathbf{R}_i$, the moment matrix becomes

$$\mathbf{A}_i = \rho \int_V \mathbf{r}\bar{\mathbf{r}}^T dV = \rho \int_V \mathbf{R}_i \bar{\mathbf{r}}\bar{\mathbf{r}}^T dV = \mathbf{R}_i \rho \int_V \bar{\mathbf{r}}\bar{\mathbf{r}}^T dV = \mathbf{R}_i \bar{\mathbf{A}}_i, \quad (2)$$

where ρ is the body's density, V its volume, $\mathbf{r}$ the distance to its center of mass and $\bar{\mathbf{A}}_i = \rho \int_V \bar{\mathbf{r}}\bar{\mathbf{r}}^T dV$ is the co-variance matrix of the body in the conforming rest state which can be pre-computed. In [2004] Blow describes an elegant way to compute this quantity for any 3D solid body represented by a triangle mesh. $\bar{\mathbf{A}}_i$ is computed relative to the center of mass of the body. The moment matrix relative to the center of the rigid union can be computed using the parallel axis theorem as $\mathbf{A}'_i = \mathbf{A}_i + m_i \mathbf{r}_i \bar{\mathbf{r}}_i^T$, where $\mathbf{r}_i$ is the distance of the center of mass of the rigid body to the center of the rigid union. This yields the final formula

$$\mathbf{A} = \sum_i \mathbf{R}_i \bar{\mathbf{A}}_i + m_i \mathbf{r}_i \bar{\mathbf{r}}_i^T. \quad (3)$$

If $\bar{\mathbf{A}}_i$ was pre-computed in the original rest state, it needs to be transformed into the conforming rest state as $\mathbf{Q}_i \bar{\mathbf{A}}_i \mathbf{Q}_i^T$, where $\mathbf{Q}_i$ is the rotation from the original into the conforming rest state.

3.6 Contact graphs

In the same way we defined a joint graph, we can also define a contact graph with contacts as nodes. Two nodes are connected if the corresponding contacts act on the same body (see the top of Figure 12). The contact graph lets us derive long range constraints to speed up and stabilize rigid body stacks. Stable stacking is a challenging problem in rigid body simulations. As mentioned in the introduction, our constraints are based on physical principles, in contrast to less physical stabilization methods such as shock propagation.

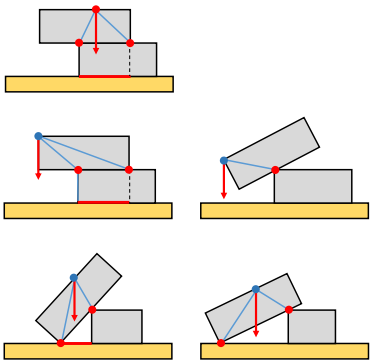


Figure 11: Supporting contacts. Top: A contact is supporting if its projection along gravity (red arrow) lies inside the convex hull (red bar) of the projections of the adjacent supporting contacts below it. Middle: The top contact is outside the convex hull and can therefore not support a load. Bottom: The top contact passes the convex hull test but is not supporting because the adjacent contacts are not static.

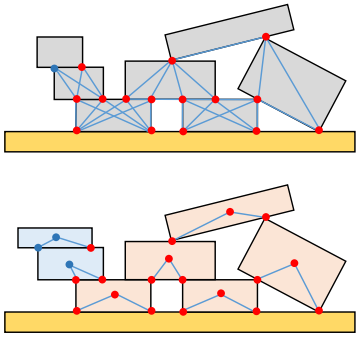


Figure 12: Top: Supporting (red) and non-supporting contacts (blue) in a more complex scenario. Bottom: If the projection of the center of mass of an object lies inside the convex hull of the projections of all adjacent supporting contacts below it, it is supported (red).

In the previous sections we have considered joints with no positional degrees of freedom. These counteract any mutual motion of the joint locations. Also, angular limits for joints are typically defined explicitly. Therefore, the derivation of upper and lower distance constraints from joint graphs is relatively simple. In contrast, contacts only prevent motion against the negative contact normal in case of dynamic friction or against any non-separating motion if the contact satisfies the static friction criterion $|\mathbf{f}_t| < \mu f_n$, where f_n is the component of the contact force along the negative

contact normal, $\mathbf{f}_t$ the tangential part of the contact force and μ the coefficient of friction. To derive the angular limits, the geometry of the objects near the contacts has to be analyzed. This is a complex task for arbitrarily shaped objects and requires the knowledge of the geometry of the bodies in the solver. To derive all possible distance constraints, the rigidity structure of the graph is needed as well. However, despite the existence of considerable mathematical theory, a general method to analyze the rigidity of a structure mathematically has not been found so far.

Therefore, we consider a special yet important case, namely a stack of rigid bodies on fixed ground under gravity as depicted in in Figure 11. Because we only consider static contacts, neither the ground nor the contacts normals need to be aligned with gravity for our method to work. Whether a contact is dynamic or static is determined by running the solver a fixed number of iterations before the contacts are analyzed.

Our goal is to identify supporting contacts based on the contact graph. We call contacts supporting if they prevent motion against gravity. A supporting contact can be interpreted as a direct, long range lower distance bound from the contact to the ground. Static contacts with the ground are supporting because the ground is fixed and the contacts prevent tangential motion. To determine the state of all other contacts, we process them from bottom to top against gravity and test the criterion depicted in Figure 11. For a given contact we determine all adjacent contacts that are marked as being supporting. Due to the processing order, they are all below the current contact. We then compute the 2d convex hull of their projections onto the plane perpendicular to gravity – typically the ground plane (the red bar). If the projection of the contact in question lies within this convex hull, it is supported by the adjacent contacts (top case). Otherwise it is not able to support a load (middle case). The convex hull test is a crucial difference between our method and shock propagation. If objects are frozen bottom to top without taking their horizontal relationship into account the three leftmost boxes in the first scene of Figure 12 remain still, which is wrong. It is important to note that dynamic (sliding) contacts cannot support other contacts (bottom case). The top image of Figure 12 shows a more complex situation with supporting (red) and non-supporting (blue) contacts.

Once we know which contacts are supporting, we can determine whether entire bodies are supported by their adjacent supporting contacts (see bottom image of Figure 12). This is the case if the projection of the center of mass of a body lies inside the convex hull of the projections of the adjacent supporting contacts below it (red bodies). To compute the 2d convex hulls we use Andrew's monotone chain method [Andrew 1979] which is a fast, simple and elegant 2d hull algorithm.

4 RESULTS

Since all our constraints are position based, they are most naturally integrated into a position based rigid body engine. This is why we used our own implementation of a position based rigid body solver based on XPBD. However, position based constraints can also be handled by impulse based solvers that support joint limits or included in the manifold projection step that most impulse based solvers perform to prevent drift.

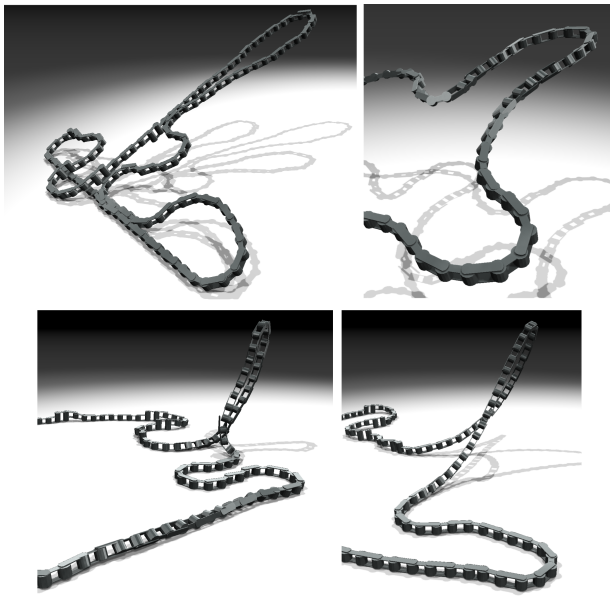


Figure 13: A circular hinge chain composed of 132 segments with joint offsets. First: The long range position and rotation constraints keep it planar and all joints parallel with only 5 iterations. Second: Long range rotation constraints are turned off. Third: Long range position constraints are also turned off. Fourth: Iteration count is increased to 100.

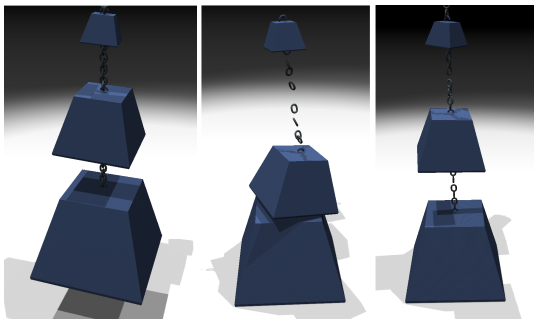


Figure 14: Weights attached to the ceiling via a chain. Left: 10 iterations and long range attachments keep the weights off the ground. Center: 10 iterations regular iterations yield substantial stretch. Right: even with 1000 iterations, the bottom weight stays on the ground.

We measured speed-ups not in terms of computation time, but in terms of comparing the number of solver iterations needed to create the same result with and without long range constraints. Of course, there is a computational overhead in solving the long distance constraints. However, since projecting a distance constraint is cheaper than solving a single joint with joint limits and since their number are of the same order, the cost of performing 10 iterations with long range constraints is typically smaller than the cost of 20 simple iterations but more effective than performing 100 or 1000

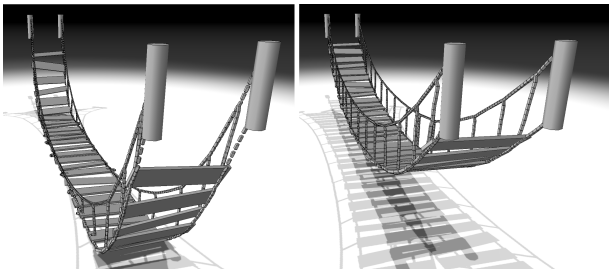


Figure 15: A rope bridge containing of 1132 joints arranged in a general graph. Left: regular simulation with 20 iterations. Right: using a hierarchy of 3 levels with the same number of solver iterations.

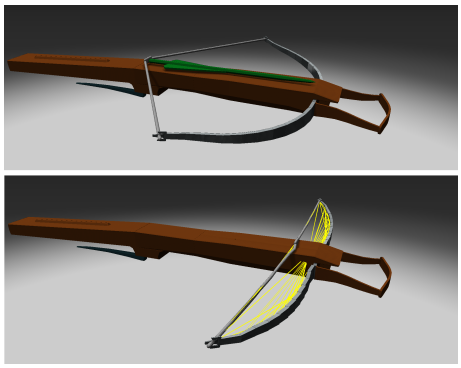


Figure 16: Applying the upper bound distance of the bow as a soft lower bound constraint stretches the bow and fires the arrow.

regular iterations. The plane and shape match constraints are even cheaper. Summing up the moment matrices is fast and Eigenvalue decomposition and polar decompositions have to be performed on 3 x 3 matrices only, independent of the constraint size.

To demonstrate the effect of long range attachment constraints we attached heavy weights connected by a chain to the ceiling as shown in Figure 14. The mass ratio between a chain segment and the heaviest weight is 1:10,000 which is difficult to handle with traditional solvers. The leftmost image shows that with long range constraints and 10 iterations, the chain keeps its length. The second image shows the same scene simulated with the same number of solver iterations but without long range constraints. Even with 1000 iterations, the heaviest weight stays on the floor, as shown in the third image.

The cable car scene shown on the right of Figure 1 is a case in which long range attachments are not applicable because the cable is not fixed at the top. Here we use a hierarchy of constraints to get a cable which is strong enough to pull one of the cars up by the weight of the other two. The mass ratio between the chain elements and the car is again on the order of 1:10,000.

Figure 13 shows a bicycle chain with 132 segments. We offset part of the chain by inserting taller segments to demonstrate the more general case. For the first image we used 5 solver iterations. In each iteration we solve all joints and perform one global projection

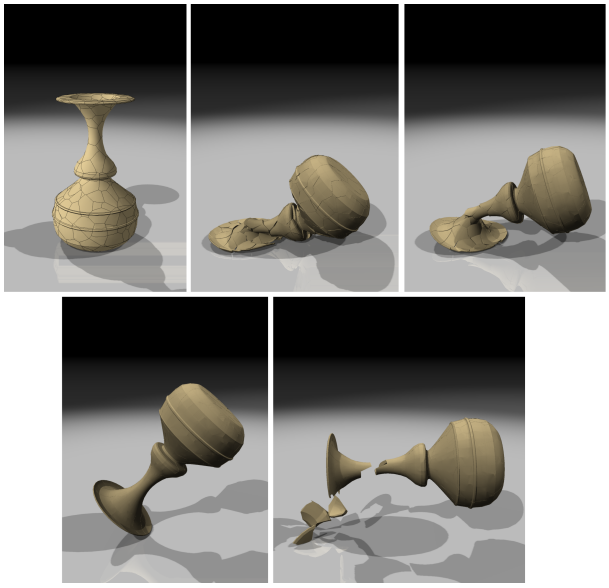


Figure 17: From top left to bottom right: a pre-split vase whose pieces are held together by fixed joints. The vase is dropped and hits the floor at an angle. With 10 iterations the vase behaves like a soft body. Increasing the iteration count to 100 increases the stiffness. With 10 iterations and shape matching, it becomes completely stiff. A mixed simulation yields force information for fracture simulation.

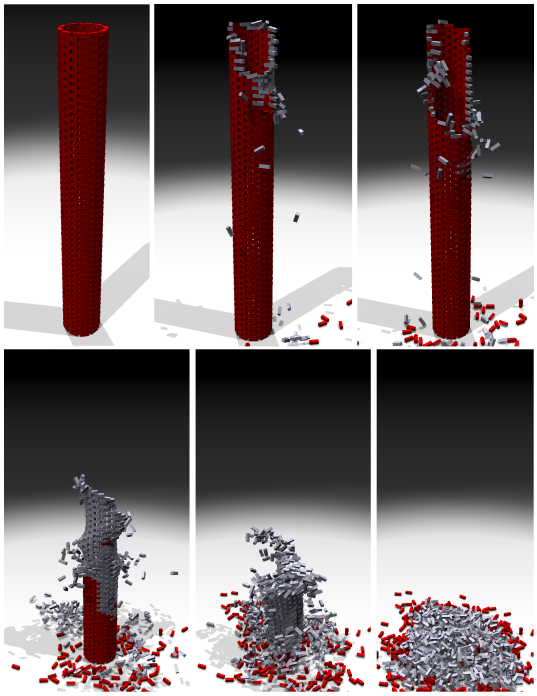


Figure 18: Destruction of a tower of 1000 rigid bodies. The red bodies are identified as being supported and are kept in position via a long range constraint to the ground.

step. In this example, projection takes about 10 percent of the time of joint projection. The second image shows what happens when long range rotational constraints are turned off. While the chain stays planar, the orientations are still propagated via short range constraints and do not align after 5 iterations. What happens when long range attachments are fully turned off and the iteration count is kept at 5 is shown in the third image. Now the chain becomes completely floppy. Increasing the iteration count to 100 as in the fourth image improves the behavior but the result is still far from correct.

To demonstrate how our method works with general networks of joints we simulated the rope bridge shown in Figure 15 which contains 1132 joints. With 20 regular solver iterations, it sags to the ground as the image on the left shows. Using a hierarchy of 3 levels yields the result on the right.

Figure 16 shows the simulation of a crossbow. Here we used the upper bounds as soft upper bounds with the effect that the bow stretches and fires the arrow even though it is modeled in a bent state.

We used generalized shape matching constraints to simulate the scenes shown in Figure 17. Using 10 iterations on the fixed joints yields a soft body instead of a brittle vase. Increasing the number of iterations to 100 improves the behavior but still does not yield the desired behavior. With a shape matching constraint and using only 10 iterations, the vase becomes perfectly rigid.

Finally for the destruction of the tower of 1000 bodies shown in Figure 18, we used the contact graph to identify supporting contacts and bodies (show in red). These are fixed in space which corresponds to having a long range constraint all the way to the ground.

5 CONCLUSION AND FUTURE WORK

We have presented the idea of long range constraints for rigid body simulations, in particular lower and upper distance bounds derived from joint graphs, plane projection and angular constraints for hinge chains and supporting contacts derived from contact graphs. Many other long range constraint types are conceivable, which opens a variety of possible future work. For instance, as mentioned, more constraints can be derived from the contact graph by considering more general cases. Moreover, the joint and contact graphs can be combined by connecting joints and contacts that act on the same body. This unified graph could allow the extraction of new types of long range constraints.

Typically, modifications to joint graph are rare and so the additional cost of deriving long range constraints is negligible. However, identifying supporting contacts can be expensive for large scenes. In our prototype, we perform a full analysis at every time step. However, this cost could be significantly reduced with the support of persistent contacts that allow for performing the analysis incrementally.

REFERENCES

Farid Amirouche. 2006. *Fundamentals Of Multibody Dynamics. Theory And Applications.* Birkhäuser.

A. M. Andrew. 1979. Another Efficient Algorithm for Convex Hulls in Two Dimensions. *Info. Proc. Letters* 9 (1979), 231–240.

William W. Armstrong and Mark W. Green. 1985. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer* 1, 4 (1985), 231–240. DOI: https://doi.org/10.1007/BF02021812

D. Baraff. 1989. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. *SIGGRAPH Comput. Graph.* 23, 3 (July 1989), 223–232. DOI: https://doi.org/10.1145/74334.74356

David Baraff. 1996. Linear-time Dynamics Using Lagrange Multipliers. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. 137–146.

Jan Bender, Kenny Erleben, and Jeff Trinkle. 2014. Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Comput. Graph. Forum* 33, 1 (Feb. 2014), 246–270. DOI: https://doi.org/10.1111/cgf.12272

Jan Bender, Matthias Müller, and Miles Macklin. 2015. Position-Based Simulation Methods in Computer Graphics. In *EUROGRAPHICS 2015 Tutorials*. Eurographics Association.

Jonathan Blow and Atman J Binstock. 2004. How to find the inertia tensor (or other mass properties) of a 3D solid body represented by a triangle mesh. (2004). http://number-none.com/blow/inertia/bb_inertia.doc

Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. 2004. Real-time Knot-tying Simulation. *Vis. Comput.* 20, 2 (May 2004), 165–179. DOI: https://doi.org/10.1007/s00371-003-0226-y

S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. 2007. OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement. *IEEE Transactions on Biomedical Engineering* 54, 11 (2007), 1940–1950. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4352056

Crispin Deul, Patrick Charrier, and Jan Bender. 2014. Position-Based Rigid Body Dynamics. *Computer Animation and Virtual Worlds* 27, 2 (2014), 103–112. DOI: https://doi.org/10.1002/cav.1614

David H. Eberly. 2010. *Game Physics.* Morgan Kaufmann, Amsterdam; Boston.

Kenny Erleben. 2007. Velocity-based Shock Propagation for Multibody Dynamics Animation. *ACM Trans. Graph.* 26, 2, Article 12 (June 2007). DOI: https://doi.org/10.1145/1243980.1243986

Roy Featherstone. 1987. *Robot Dynamics Algorithms.* Springer.

Eran Guendelman, Robert Bridson, and Ronald Fedkiw. 2003. Nonconvex Rigid Bodies with Stacking. *ACM Trans. Graph.* 22, 3 (July 2003), 871–878. DOI: https://doi.org/10.1145/882262.882358

James K. Hahn. 1988. Realistic Animation of Rigid Bodies. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 299–308. DOI: https://doi.org/10.1145/378456.378530

T. Jakobsen. 2001. Advanced Character Physics Ű The Fysix Engine. *www.gamasutra.com* (2001).

Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. 2012. Long Range Attachments - a Method to Simulate Inextensible Clothing in Computer Games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 305–310.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games (MIG '16)*. ACM, New York, NY, USA, 49–54. DOI: https://doi.org/10.1145/2994258.2994272

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-time Applications. *ACM Trans. Graph.* 33, 4, Article 153 (July 2014), 12 pages. DOI: https://doi.org/10.1145/2601097.2601152

Brian Vincent Mirtich. 1996. *Impulse-based Dynamic Simulation of Rigid Body Systems.* Ph.D. Dissertation. AAI9723116.

Matthias Müller. 2008. Hierarchical Position Based Dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)*, Francois Faure and Matthias Teschner (Eds.). The Eurographics Association. DOI: https://doi.org/10.2312/PE/vriphys/vriphys08/001-010

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.* 24, 3 (July 2005), 471–478. DOI: https://doi.org/10.1145/1073204.1073216

M. Müller, B. Heidelberger M. Hennix, and J. Ratcliff. 2006. Position Based Dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2006), 71–80.

Matthias Müller, Tae-Yong Kim, and Nuttapong Chentanez. 2012. Fast Simulation of Inextensible Hair and Fur. In *Workshop on Virtual Reality Interaction and Physical Simulation*, Jan Bender, Arjan Kuijper, Dieter W. Fellner, and Eric Guerin (Eds.). The Eurographics Association. DOI: https://doi.org/10.2312/PE/vriphys/vriphys12/039-044

Bruno Siciliano and Oussama Khatib. 2007. *Springer Handbook of Robotics.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Shinjiro Sueda, Garrett L. Jones, David I. W. Levin, and Dinesh K. Pai. 2011. Large-scale Dynamic Simulation of Highly Constrained Strands. *ACM Trans. Graph.* 30, 4 (July 2011), 39:1–39:10.

Robin Tomcin, Dominik Sibbing, and Leif Kobbelt. 2014. Efficient Enforcement of Hard Articulation Constraints in the Presence of Closed Loops and Contacts. *Computer Graphics Forum* (2014). DOI: https://doi.org/10.1111/cgf.12322

Jumpei Tsuda. 2009. Practical Rigid Body Physics for Games. In *ACM SIGGRAPH ASIA 2009 Courses (SIGGRAPH ASIA '09)*. ACM, New York, NY, USA, Article 14, 83 pages. DOI: https://doi.org/10.1145/1665817.1665831