Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision

Rachel Weinstein, Joseph Teran, and Ron Fedkiw

Abstract— **We propose a novel approach for dynamically simulating articulated rigid bodies undergoing frequent and unpredictable contact and collision. In order to leverage existing algorithms for nonconvex bodies, multiple collisions, large contact groups, stacking, etc., we use maximal rather than generalized coordinates and take an impulse based approach that allows us to treat articulation, contact and collision in a unified manner. Traditional constraint handling methods are subject to drift, and we propose a novel pre-stabilization method that does not require tunable potentially stiff parameters as does Baumgarte stabilization. This differs from post-stabilization in that we compute allowable trajectories before moving the rigid bodies to their new positions, instead of correcting them after the fact when it can be difficult to incorporate the effects of contact and collision. A post-stabilization technique is used for momentum and angular momentum. Our approach works with any black box method for specifying valid joint constraints, and no special considerations are required for arbitrary closed loops or branching. Moreover, our implementation is linear both in the number of bodies *and* in the number of auxiliary contact and collision constraints, unlike many other methods that are linear in the number of bodies but not in the number of auxiliary constraints.**

Index Terms— **I.3 Computer Graphics, I.3.5.i Physically based modeling,, I.6.8.a Animation, I.2.9.d Kinematics and dynamics**

I. Introduction

ARTICULATED rigid bodies have wide ranging applications across fields including molecular dynamics, robotics, machine assembly, human motion, and computer graphics for video games and feature films. In many applications, the focus is on kinematics or inverse kinematics, and dynamic behavior is of secondary or limited interest with only a few instances of predictable contact and collision, e.g. consider manipulating a robotic arm in a controlled environment with contact occurring only with the end effector as planned. However, more typically in computer animation for film and games, simulated dynamic behavior of articulated rigid bodies undergoing intricate interactions (via contact and collision) with arbitrary, complex environments is of interest. We present an articulated rigid body simulation framework that is more appropriate for graphics applications involving such complex scenes, and provide a number of examples that demonstrate our ability to attain visually plausible, compelling results.

Rachel Weinstein and Joseph Teran are with the Department of Computer Science, Stanford University. Ron Fedkiw is with the Department of Computer Science, Stanford University, and Industrial Light and Magic. Email: {rachellw,jteran}@graphics.stanford.edu, fedkiw@cs.stanford.edu.

There are traditionally two choices for maintaining the constraints that articulate a group of rigid bodies. The first class of algorithms are reduced coordinate (or generalized coordinate) formulations that parameterize the degrees of freedom in a manner consistent with the constraints of articulation, effectively reducing the overall degrees of freedom by eliminating those that could violate the constraints. Unless there are many closed loops or frequent and unpredictable contact and collision, generalized coordinates are typically preferred due to the increased efficiency. In the absence of closed loops and unpredictable contact and collision, any choice of the system state is admissible (i.e. satisfies the constraints). However, consistency conditions are required to ensure that closed loops are actually closed adding a nonlocal constraint to the system, and many popular methods violate this constraint proceeding as though there were no closed loops and applying penalty forces at loop closures. Note that limits on the *range* of allowable degrees of freedom are local constraints easily enforced by clamping, but adding a global closed loop constraint can complicate the treatment of these local constraints as well. Contact and collision provide even more challenging scenarios, since one cannot tell if a contact point is active or nonactive (separating velocity) until after the problem is solved. Moreover, active points essentially behave as new articulations so that frequent contact and collision effectively increase the number of closed loops.

[1] pointed out that linear complexity can readily be achieved using maximal coordinates. Closed loops and contact were incorporated drawing on the earlier work of [2] (see also [3]–[6]), but the proposed algorithm does *not* retain linear complexity with regards to the

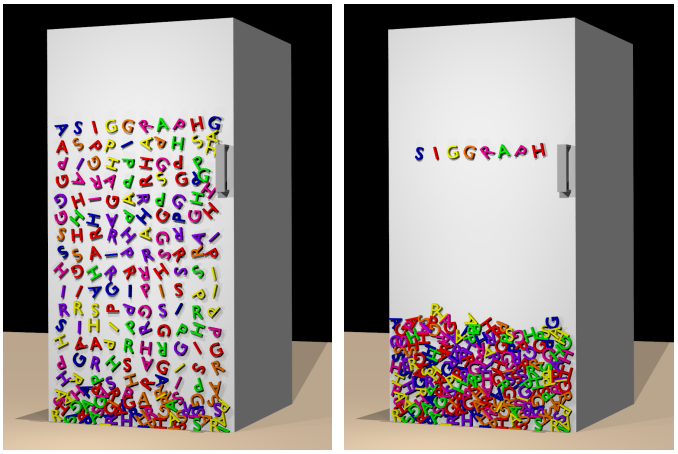

Fig. 1. The use of prismatic (translational) constraints forces "magnetic" letters to stay constrained to the surface of a refrigerator.

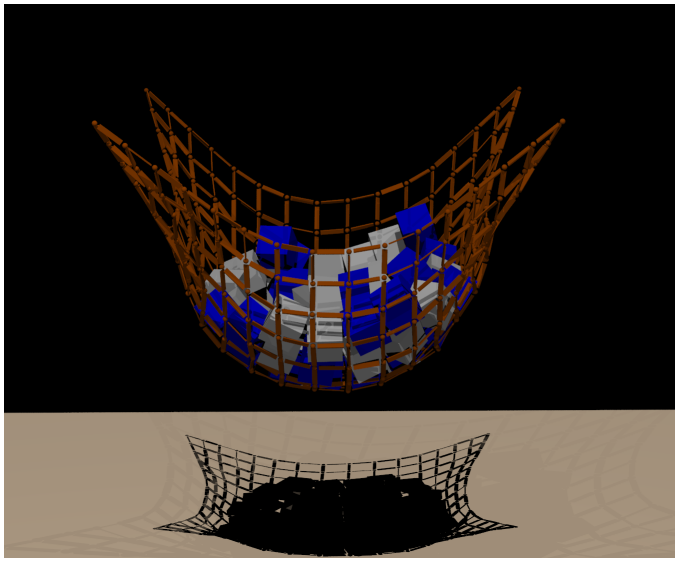

Fig. 2. The net represents a significant number of closed loops, but maintains stability even when weighed down by blocks.

number of loop closures and contact points (although [7], [8] worked to treat closed loops iteratively somewhat improving the situation). [9], [10] reformulated the algorithm of [1] in the context of generalized coordinates showing that one obtains essentially the same equations and complexity, but with less degrees of freedom since some of them are automatically eliminated. We have found that it has been easier to design algorithms to treat closed loops and frequent contact and collision in maximal coordinates using a unified treatment, and propose a novel maximal coordinates approach that builds on the previous work of [11] to treat large contact groups and stacking with linear complexity in *both* the number of bodies and the number of auxiliary constraints. However, it might be advantageous to subsequently reformulate our algorithm in generalized coordinates where gains can be achieved by reducing the number of variables in the problem. We note that the extension of our work to generalized coordinates is promising (but not obvious) in light of [12] showing that [13], [14] could be used to process collisions with an articulated rigid body simulated in generalized coordinates (see also [9]).

A number of energy minimization techniques were explored for constraint handling, see e.g. [15]–[18], and [19] (see also [20]) outlined a general methodology for handling constraints. Basically, if the constraint and its derivative are satisfied initially, one needs to ensure that the net force does not act to violate the constraint. Also, to combat constraint violation due to numerical drift, Baumgarte stabilization [21] is typically used applying penalty forces based on the error in the constraint and its derivatives. This amounts to a damped spring type of force, and there can be difficulties associated with choosing the proper parameters. The differential algebraic equations (DAE) community has pointed out many flaws with this technique recommending *post*-stabilization techniques instead, see e.g. [22]–[24]. These techniques project the solution back onto the acceptable constraint manifold *after* each time step. Ad hoc methods for correcting the positions and orientations were proposed in [25] (see also the procedural approach in [26]) correcting the velocities based on the final state, and an improved iterative method was proposed in [27] (see also [8]). Our approach has notable differences such as solving a set of nonlinear equations locally and using impulses (as opposed to displacements) so that the approach can be combined with state of the art contact and collision algorithms. [28] introduced post-stabilization into the LCP formulation of [1], [2] requiring the solution of a second LCP (although they state that this is not necessary). They noted stability problems when there were large errors in the constraints causing their linearization to be invalid. We avoid these difficulties by using *pre*-stabilization on a joint by joint basis whereby we only evolve the bodies *after* our iterative solver finds impulses that yield an acceptable state. Moreover, as stated above, our algorithm retains linearity in the number of auxiliary constraints allowing us to solve large problems with many closed loops, collisions, and large contact groups.

Our pre-stabilization method relies on solving three by three nonlinear systems of equations that allow us to target any desired joint state. One simply temporarily evolves a pair of rigid bodies forward in time to obtain a predicted joint state, and then uses any black box joint model (see for example [29]–[31] and the references therein) to determine the closest allowable or desired joint state as input to our nonlinear solver. While pre-stabilization is used for the positions and orientations, a more standard post-stabilization technique is used for the velocities. Our impulse based framework allows us

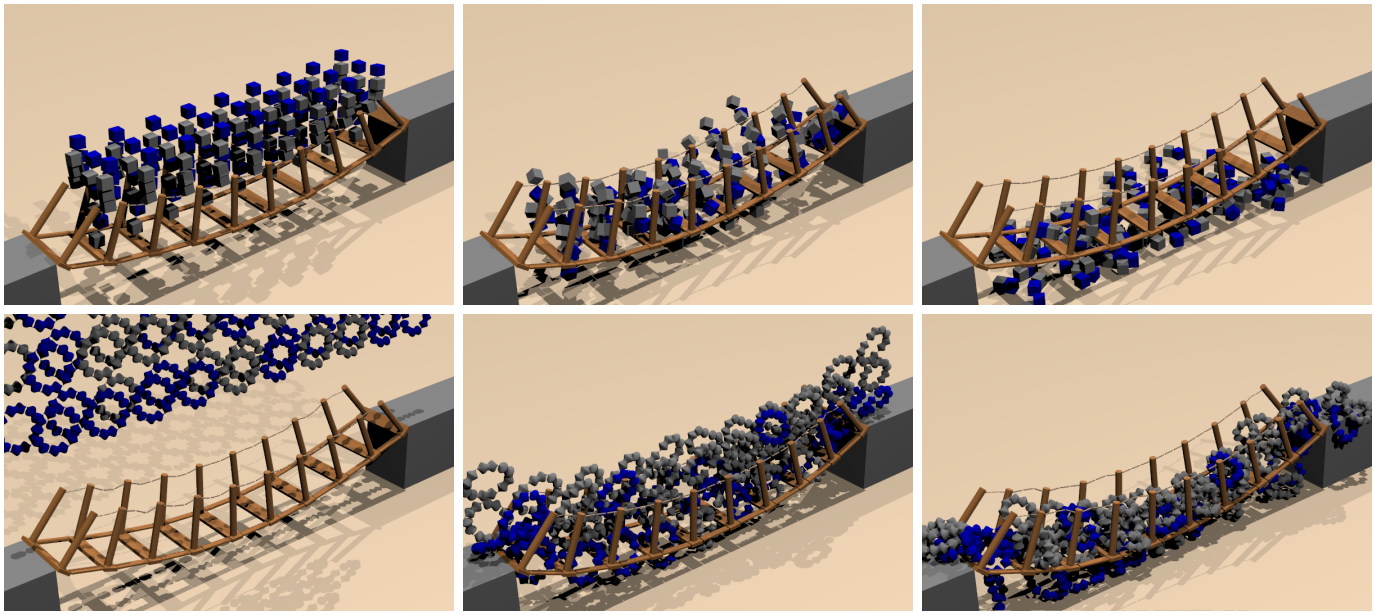

Fig. 3. These bridges demonstrate the stability of multiple closed loops as shown by the loops created in the base and those between the ropes and the bridge itself. The closed loops maintain their stability even when subjected to collisions from falling articulated blocks (top) or closed loops of non-convex rigid bodies (bottom).

to fully couple our new treatment of articulated rigid bodies to the rigid body contact and collision handling algorithms proposed in [11] including contact graphs, shock propagation, staggered integration of position and velocity with contact and collision, etc.

II. Previous Work

Although we have discussed the works needed to put our contributions in context, we discuss a few historical works below focusing on dynamic simulation of articulated rigid bodies as our algorithm is specific to that area. The problem of simulating articulated rigid bodies is a long-standing problem, e.g. see [32]. [33] built legged figures incorporating some simplified dynamics, [34], [35] simulated articulated rigid bodies using applied torques to control joint angles, and [36], [37] used similar techniques including spring forces to enforce joint limits and collisions with the ground. Spring based joint forces were also used in [38], which was later extended to handle closed loops and collisions using Lagrange multipliers including penalty terms to combat drift [39]. [40] dynamically simulated rigid bodies including contact and collision, as well as kinematically controlled articulated bodies that dynamically responded to joint motion and collisions with other objects (behaving as a single rigid body for these interactions). [41] proposed an impulse based approach to collisions between rigid bodies and articulated rigid bodies (not in generalized coordinates) solving a single matrix equation to find the impulses which handle collisions as well as those that properly constrain the velocities at articulation points (holding the joints together). See also [42].

[43] proposed spacetime constraints for controlling articulated characters. This has turned out to be quite useful for character animation, see e.g. [44], but departs from our focus on dynamic simulation. In the area of simulation, [45], [46] implemented recursive algorithms with [46] able to handle closed loops, [47] used a Lagrange multiplier approach for protein modeling, [48] proposed using impulses to treat collisions while inserting springs to model contact, and [7] used a mix of impulses, temporary joint constraints, and the method of [49] to model collision, contact and friction. [50], [51] proposed an $O(n)$ method although it required loop decomposition, dealt with only a handful of rigid bodies, and presented no clear method for handling large numbers of unpredictable contact and collision events. See also the Lagrange multiplier approach to rigid (and deformable) bodies in [52]. [53] extended the contact graph proposed in [40] for rigid body collisions to the mixed articulated rigid body and free rigid body case using it to assemble the global linear system of equations for the impulses. In the area of contact and collision detection, we follow the work of [11], but refer the interested reader to [54] and the references therein.

III. Equations

A rigid body is characterized by its world frame $\mathbf{F}^w = \{\mathbf{x}, \mathbf{q}\}$ consisting of position and orientation, and is evolved in time via $\mathbf{x}_t = \mathbf{v}$, $\mathbf{q}_t = (1/2)\omega\mathbf{q}$, $\mathbf{v}_t = \mathbf{f}/m$ and $\mathbf{L}_t = \tau$ where $\mathbf{v}$ and ω are the velocity and angular velocity, $\mathbf{f}$ is the net force, m is the mass, $\mathbf{L} = \mathbf{I}\omega$ is the

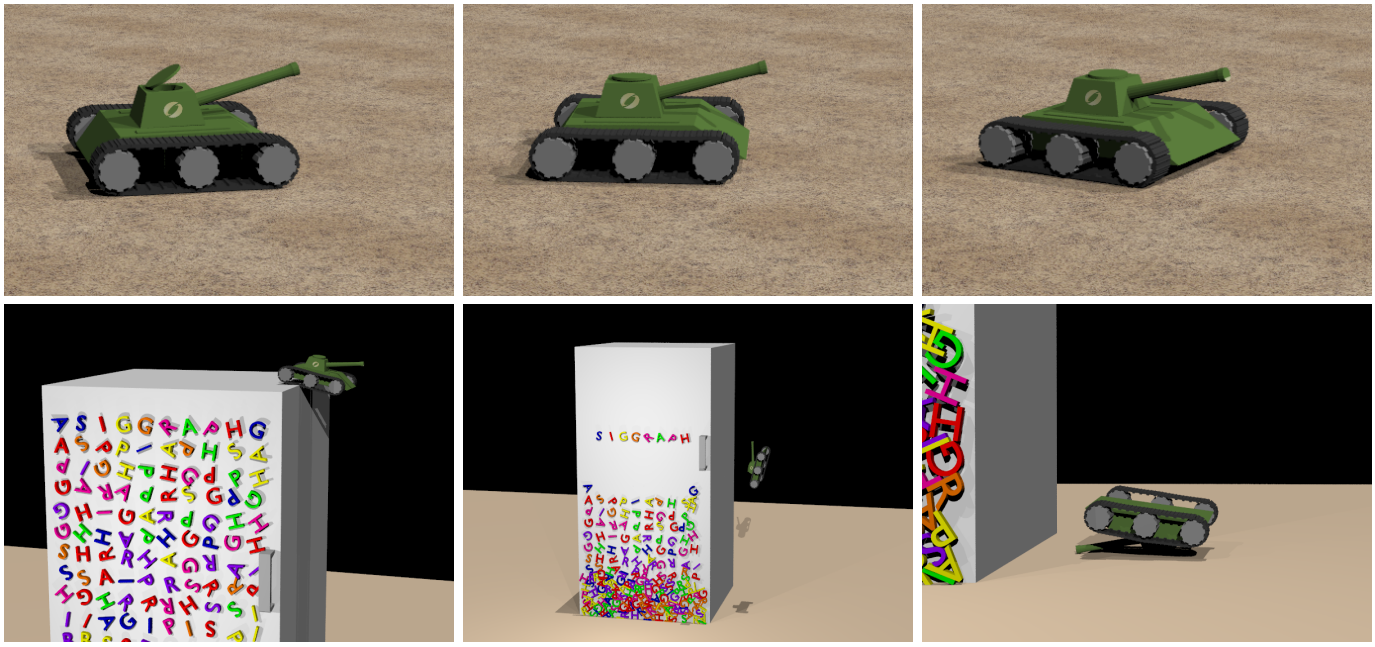

Fig. 4. Dynamic constraints between the treads and gears give control over the movement of the tank. The tank's differential steering allows for controllable turning (top), and the treads remain firmly attached to the gears even when gravity no longer holds them on (bottom).

angular momentum with inertia tensor $\mathbf{I} = \mathbf{R}\mathbf{D}\mathbf{R}^T$ ($\mathbf{R}$ is the orientation matrix and $\mathbf{D}$ is the diagonal inertia tensor in object space), and τ is the net torque. The position and orientation are time integrated according to

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \triangle t \mathbf{v}^n, \quad \mathbf{q}^{n+1} = \hat{\mathbf{q}}\left(\triangle t \omega^n\right)\mathbf{q}^n \qquad (1)$$

where $\hat{\mathbf{q}}(\omega) = (\cos(|\omega|/2), sin(|\omega|/2)\omega/|\omega|)$ is a unit quaternion so no extra normalization step need be applied (although occasional normalization combats drift). This particular form of the position and orientation time integration is quite important, since it is used in our stabilization procedure. The velocity and angular momentum are updated with standard forward Euler time integration.

Consider an articulation between a parent and child rigid body with world frame states $\mathbf{F}_p^w$ and $\mathbf{F}_c^w$ respectively. We denote the joint centers rigidly attached to each body by $\mathbf{F}_p^j$ and $\mathbf{F}_c^j$, and the joint state $\mathbf{J} = \{\mathbf{x}_j, \mathbf{q}_j\}$ by $\mathbf{J} = \mathbf{F}_p^j \mathbf{F}_w^p \mathbf{F}_c^w \mathbf{F}_j^c$ where we define $\mathbf{F}_a^b = (\mathbf{F}_b^a)^{-1}$ and multiplication by $\{\mathbf{x}_1, \mathbf{q}_1\}\{\mathbf{x}_2, \mathbf{q}_2\} = \{\mathbf{x}_1 + \mathbf{q}_1[\mathbf{x}_2], \mathbf{q}_1\mathbf{q}_2\}$. The articulation constraint comes from limiting the class of acceptable joint frames, e.g. purely revolute joints have constant $\mathbf{x}_j$ and purely prismatic joints have constant $\mathbf{q}_j$. Given a current joint state $\mathbf{J}^n$, equation 1 evolves the bodies forward in time to obtain a new joint state $\mathbf{J}^{n+1} = \mathbf{F}_p^j(\mathbf{F}_w^p)^{n+1}(\mathbf{F}_c^w)^{n+1}\mathbf{F}_j^c$. The predicted joint state $\mathbf{J}^{n+1}$ is typically not in the feasible region, and any black box joint model (see e.g. [29]–[31]) can be used to project $\mathbf{J}^{n+1}$ to the desired state $\mathbf{J}^{target}$. The goal of our *pre*-stabilization method is to apply impulses to the rigid bodies *before* integrating with equation 1 with the intention of achieving $\mathbf{J}^{target}$ after integrating with equation 1. Since we exactly target the desired $\mathbf{J}^{target}$ every time step, there are no cumulative errors (no drift) as the simulation proceeds in time. All errors at a given time step are incurred from tolerances on the iterative schemes applied in that time step.

We apply an impulse $\mathbf{j}$ to a pair of bodies via

$$\mathbf{v}^{new} = \mathbf{v}^n \pm \mathbf{j}/m, \quad \omega^{new} = \omega^n \pm \mathbf{I}^{-1}\left(\mathbf{r} \times \mathbf{j}\right) \qquad (2)$$

where $\mathbf{r}$ is the vector from the rigid body's center of mass to the application point of the impulse. For articulation, we apply the impulse at the midpoint between $\mathbf{x}_p^w + \mathbf{q}_p^w[\mathbf{x}_j^p + \mathbf{q}_j^p[\mathbf{x}^{target}]]$ and $\mathbf{x}_c^w + \mathbf{q}_c^w[\mathbf{x}_j^c]$ at time n noting that these two points are coincident when the constraint is satisfied (see equation 5). For the orientation

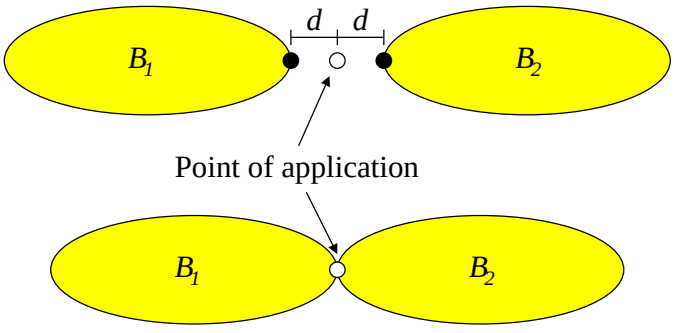

Fig. 5. The application point of impulses is the midpoint (shown as a white circle) of the two articulation points $\mathbf{x}_p^w + \mathbf{q}_p^w[\mathbf{x}_j^p + \mathbf{q}_j^p[\mathbf{x}^{target}]]$ and $\mathbf{x}_c^w + \mathbf{q}_c^w[\mathbf{x}_j^c]$ (shown as black circles). In the top image, the constraint is not satisfied and the distance d to each articulation point is the same. In the bottom image, the constraint is satisfied so the two articulation points are coincident.

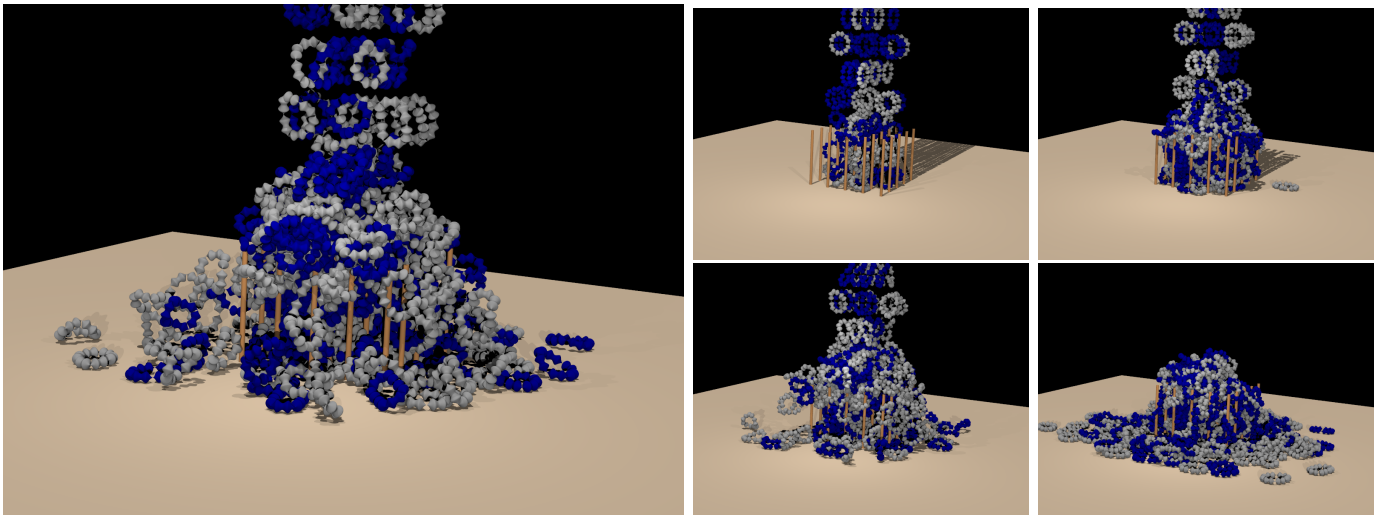

Fig. 6. We stacked 360 rings, each constructed by point constraining 6 non-convex objects together, for a total of 2160 rigid bodies and 2.9 million triangles.

constraint, we use a purely angular impulse setting the linear impulse $\mathbf{j}$ to zero. If the point of application (and thus $\mathbf{r}$) goes to infinity as $\mathbf{j}$ goes to zero, the quantity $\mathbf{j}_\tau = \mathbf{r} \times \mathbf{j}$ can remain finite. Then we obtain

$$\mathbf{v}^{new} = \mathbf{v}^n \pm \mathbf{0}, \quad \omega^{new} = \omega^n \pm \mathbf{I}^{-1}\mathbf{j}_\tau \tag{3}$$

which is a solely angular impulse. Note that we use the convention that the positive (negative) impulse is applied to the parent (child) rigid body.

IV. Enforcing Articulation Constraints

In order to maintain the proper articulation, we apply impulses at each joint at time n so that after time integration with equation 1 we obtain

$$\left(\mathbf{F}_p^w\right)^{n+1} \mathbf{F}_j^p \mathbf{J}^{target} = \left(\mathbf{F}_c^w\right)^{n+1} \mathbf{F}_j^c \tag{4}$$

where we have moved the frames related to the parent to the left hand side. This entails solving a $6n$ degree of freedom nonlinear equation where n is the number of joints in the articulated rigid body, and the situation becomes worse for contact and collision. Thus, we take an iterative approach similar to and commensurate with the approach for contact and collision proposed in [11], i.e. we apply impulses one joint at a time. Furthermore, we iteratively solve the 6 degree of freedom nonlinear system for each joint by first finding and applying a linear impulse to satisfy the positional constraint, and then finding and applying an angular impulse to satisfy the orientation constraint.

We define the desired target position of the joint via

$$\left(\mathbf{x}_p^w\right)^{n+1} + \left(\mathbf{q}_p^w\right)^{n+1} \left[\mathbf{x}_j^p + \mathbf{q}_j^p \left[\mathbf{x}^{target}\right]\right] = \left(\mathbf{x}_c^w\right)^{n+1} + \left(\mathbf{q}_c^w\right)^{n+1} \left[\mathbf{x}_j^c\right] \tag{5}$$

from equation 4. We first apply an impulse to the time n states according to equation 2, and then integrate in time according to equation 1, plugging the results into equation 5 and rearranging to obtain

$$\begin{aligned}\mathbf{f}(\mathbf{j}) \;=\; & \left(\mathbf{x}_p^w\right)^n + \Delta t \left(\mathbf{v}_p^w\right)^n + \frac{\Delta t}{m_p}\mathbf{j} \\ & + \hat{\mathbf{q}}\left(\triangle t \left(\omega_p^w\right)^n + \triangle t \mathbf{I}_p^{-1}\mathbf{r}_p^*\mathbf{j}\right)\left[\left(\mathbf{q}_p^w\right)^n \left[\mathbf{x}_j^p + \mathbf{q}_j^p \left[\mathbf{x}^{target}\right]\right]\right] \\ & - \left(\mathbf{x}_c^w\right)^n - \Delta t \left(\mathbf{v}_c^w\right)^n + \frac{\Delta t}{m_c}\mathbf{j} \\ & - \hat{\mathbf{q}}\left(\triangle t \left(\omega_c^w\right)^n - \triangle t \mathbf{I}_c^{-1}\mathbf{r}_c^*\mathbf{j}\right)\left[\left(\mathbf{q}_c^w\right)^n \left[\mathbf{x}_j^c\right]\right] = \mathbf{0}.\end{aligned}$$

Note that this equation is nonlinear, since points on rigid bodies follow nonlinear paths. We solve this equation with Newton iteration where each consecutive iterate $\mathbf{j}^{m+1}$ is obtained by solving the 3x3 linear system $(\partial \mathbf{f}/\partial \mathbf{j})|_{\mathbf{j}^m} \triangle \mathbf{j} = -\mathbf{f}(\mathbf{j}^m)$ for $\triangle \mathbf{j} = \mathbf{j}^{m+1} - \mathbf{j}^m$ (as an initial guess, we use $\mathbf{j}^0 = \mathbf{0}$). The computation of $\partial \mathbf{f}/\partial \mathbf{j}$ can be found in the appendix.

We define the desired target orientation of the joint

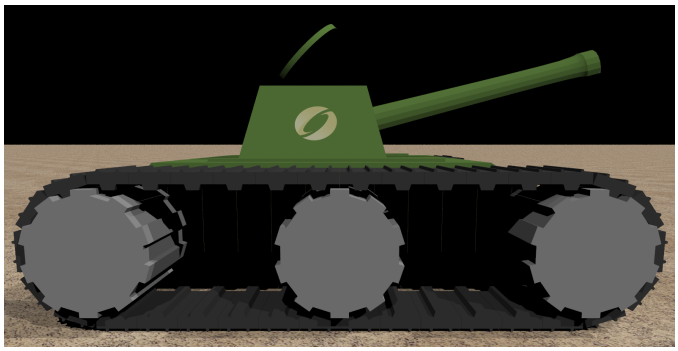

Fig. 7. Dynamic constraints between the treads and the front and back gears are updated each frame (added or deleted as necessary). The middle gear is passively attached to the base of the tank with a twist joint. A hinge constraint keeps the lid attached, and a constraint between the gun and the base allows for gun movement.

via

$$\left(\mathbf{q}_p^w\right)^{n+1}\mathbf{q}_j^p\mathbf{q}^{target} = \left(\mathbf{q}_c^w\right)^{n+1}\mathbf{q}_j^c \tag{6}$$

from equation 4. We first apply an impulse to the time n states according to equation 3, and then integrate in time according to equation 1, plugging the results into equation 6 and rearranging to obtain

$$\begin{aligned}\mathbf{f}\left(\mathbf{j}_\tau\right) &= \hat{\mathbf{q}}\left(\triangle t\left(\omega_p^w\right)^n + \triangle t\mathbf{I}_p^{-1}\mathbf{j}_\tau\right)\left(\mathbf{q}_p^w\right)^n\mathbf{q}_j^p\mathbf{q}^{target} \\ &\quad -\hat{\mathbf{q}}\left(\triangle t\left(\omega_c^w\right)^n - \triangle t\mathbf{I}_c^{-1}\mathbf{j}_\tau\right)\left(\mathbf{q}_c^w\right)^n\mathbf{q}_j^c = \mathbf{0}.\end{aligned}$$

We again use Newton iteration noting that $\left(\partial\mathbf{f}/\partial\mathbf{j}_\tau\right)|_{\mathbf{j}_\tau^m}$ is a 4x3 matrix, since equation 6 has an equation for each of the 4 components of the quaternion. We use the normal equations to solve the linear system for $\triangle\mathbf{j}_\tau$ at each step. The computation of $\partial\mathbf{f}/\partial\mathbf{j}_\tau$ for angular constraints can be found in the appendix.

Since we iterate through all the joints correcting errors in the individual joint states one at a time, we obtain higher accuracy by correcting each joint a small amount at a time. To accomplish this, we accept a solution if the error is reduced by ϵ times the initial error amount where the parameter ϵ gradually increases from a small positive number to 1 as the iterations proceed. This targets zero error in the final sweep through the joints, but allows for larger errors in initial sweeps. A similar technique was used in [11] to improve the accuracy of the impulses used when processing contacts. Also, to increase robustness and accuracy while solving each small nonlinear system, we slightly modify the Newton iteration via $\mathbf{j}^{m+1} = \mathbf{j}^m + \epsilon\triangle\mathbf{j}$ (and $\mathbf{j}_\tau^{m+1} = \mathbf{j}_\tau^m + \epsilon\triangle\mathbf{j}_\tau$). A second technique we employ is to use $\triangle\mathbf{j}$ (and $\triangle\mathbf{j}_\tau$) as a search direction employing the golden section search method to compute a more robust minimum. This also typically reduces the number of Newton iterations required.

V. Velocity Post-Stabilization

We use impulses to enforce the position and orientation based articulation constraints so that the technique can be mixed in directly with contact and collision. If post-stabilization (instead of pre-stabilization) was used, we would have to correct the positions of objects after integrating them forward in time, and these corrections would need to be contact and collision aware both causing errors and reducing accuracy. Moreover, it is not clear how this could be accomplished. However, once the objects have been moved to their new positions in a contact, collision and articulation constraint aware fashion, we can apply a post-stabilization technique to project the velocities onto the desired manifold. In fact, we apply this technique quite often whenever consistent velocity values are needed.

Given current values for the velocities, we project them to acceptable values by applying impulses in an iterative fashion one joint at a time. The desired value is found in a straightforward manner by projecting the velocity in the constrained degrees of freedom leaving the other degrees of freedom unchanged. For example, a point constraint dictates that the relative linear velocity should be identically zero while the angular velocities should remain unchanged. For a completely rigid joint, both the linear and angular relative velocities should be identically zero.

The relative velocities at an articulation point are given by $\mathbf{u}_{rel} = \mathbf{u}_p - \mathbf{u}_c$ and $\omega_{rel} = \omega_p - \omega_c$ where $\mathbf{u} = \mathbf{v} + \omega \times \mathbf{r}$ for each body. If both a linear and an angular impulse are simultaneously applied we obtain updated values for the relative linear and angular velocities as given by equations 2 and 3, i.e.

$$\begin{aligned}\mathbf{u}_{rel}^{new} &= \mathbf{u}_{rel} \\ &\quad + \left(\left(\frac{1}{m_p} + \frac{1}{m_c}\right)\delta + \mathbf{r}_p^{*T}\mathbf{I}_p^{-1}\mathbf{r}_p^* + \mathbf{r}_c^{*T}\mathbf{I}_c^{-1}\mathbf{r}_c^*\right)\mathbf{j} \\ &\quad + \left(\mathbf{r}_p^{*T}\mathbf{I}_p^{-1} + \mathbf{r}_c^{*T}\mathbf{I}_c^{-1}\right)\mathbf{j}_\tau \\ \omega_{rel}^{new} &= \omega_{rel} + \left(\mathbf{I}_p^{-1}\mathbf{r}_p^* + \mathbf{I}_c^{-1}\mathbf{r}_c^*\right)\mathbf{j} + \left(\mathbf{I}_p^{-1} + \mathbf{I}_c^{-1}\right)\mathbf{j}_\tau.\end{aligned}$$

This is a linear system of two vector equations in two vector unknowns ($\mathbf{j}$ and $\mathbf{j}_\tau$), and can be solved by solving one equation for $\mathbf{j}$, plugging the results into the second to obtain $\mathbf{j}_\tau$, and then using $\mathbf{j}_\tau$ to find $\mathbf{j}$ (as is typical).

Once again, these corrections can be applied a small amount at a time by using a parameter ϵ that gradually increases from a small positive number to 1 as the iterations proceed. That is, instead of projecting velocities from their current value to their target value, one projects them by an ϵ fraction amount towards their target value.

VI. Hybridization with Contact and Collision

The primary benefit of our impulse based pre-stabilization technique is that the impulses used to enforce the position and orientation articulation constraints can be seamlessly integrated with those in an impulse based contact and collision algorithm such as that proposed in [11]. We integrate our algorithm with theirs as follows.

- Process Collisions (and Velocity Post-Stabilization)
- Integrate Velocities (and Velocity Post-Stabilization)
- Resolve Contacts & Articulation Pre-Stabilization
- Integrate Positions (and Velocity Post-Stabilization)

Since processing collisions and integrating the velocity forward in time both change the values of the velocity

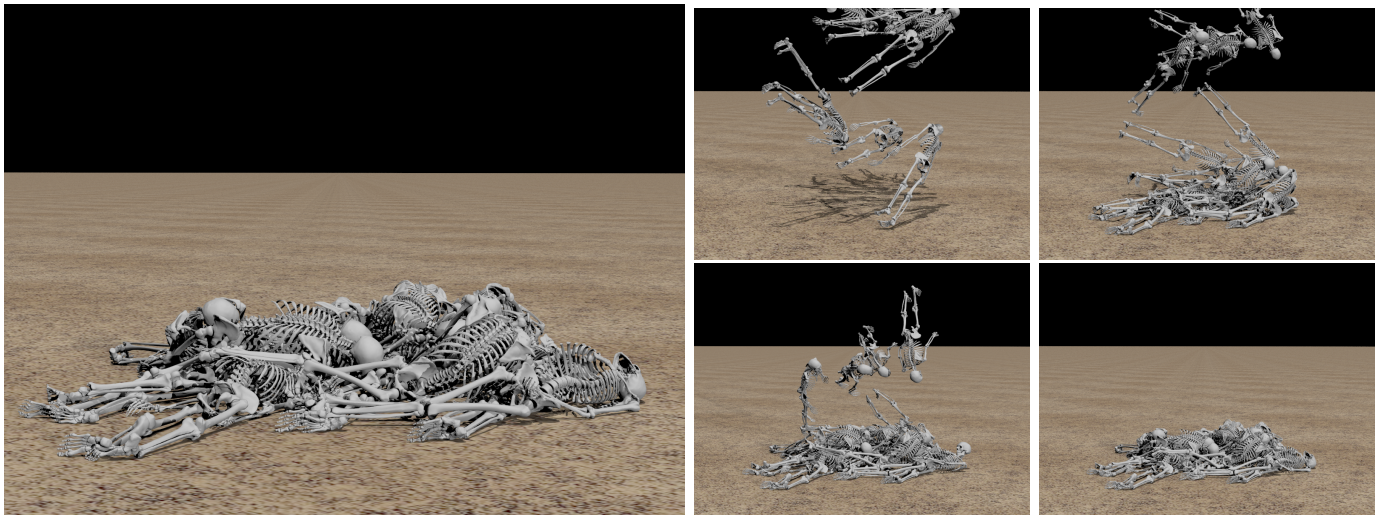

Fig. 8. Twenty skeletons, each consisting of over 1 million triangles, were stacked on top of each other. The largest computational cost was in resolving contact and collision, while the overhead of articulation constraints was nearly negligible in comparison.

field, the velocity post-stabilization technique is used after each of these steps. Pre-stabilization is tightly integrated with the contact processing algorithm. First a contact graph is constructed as usual, and impulses are used to resolve contact one level of the graph at a time. After processing all the contacts of all the bodies in a certain level with bodies at lower levels (as usual), the articulation constraints between any body in that level with a body of a lower level are processed. Shock propagation is the last step of contact handling, and a similar process can be used. However, we noticed non-intuitive results processing articulation constraints between two bodies when one of them temporarily has infinite inertia, so we remove the infinite inertia aspect of the shock propagation algorithm *only* when computing the impulses due to articulation (otherwise, it is used as usual). Alternatively, the articulation constraints can be ignored completely during the shock propagation phase. Then an optional sweep through the contact graph processing *only* the articulation constraints can be added after shock propagation. While this is not necessary, we have found that it improves the accuracy of the articulation constraints.

Finally, the positions and orientations are integrated forward in time using the velocities calculated by applying all the impulses necessary to handle both the contact and articulation constraints. This has allowed us to treat large stacks of articulated bodies with robust contact *and* articulation handling. Note that we do not apply velocity post-stabilization before integrating the positions forward in time, since the velocities obtained during contact processing and pre-stabilization should not be modified until the positions and orientations are corrected for contact and constraint enforcement. However, after moving the bodies, we once again apply velocity post-stabilization. In order to further increase the robustness of the optional final sweep through the contact graph processing only articulation constraints (i.e. after shock propagation), we optionally save the velocities before doing this step. Then these saved velocities are restored after integrating the positions but before the final application of velocity post-stabilization. This further insulates the articulation constraints from the shock propagation for added stability.

VII. Examples

Although a theoretical proof is beyond the scope of this paper, the implementation of our algorithm is provably linear in complexity. We take a finite number of sweeps through all the bodies (always less than 10) for all examples. The number of iterations was typically chosen based on running an example for several frames. For example, 9 iterations was sufficient for over 2,000 contact/collision coupled bodies, while 5 iterations was insufficient and 15 iterations gave results visually identical to 9 iterations. During each sweep, each articulation and auxiliary constraint is considered exactly once, and thus each sweep is $O(n)$ in these constraints.

We assessed our algorithm using a series of examples which demonstrate typical articulated rigid bodies as well as those which are traditionally more difficult. Examples with complex articulation, branching and closed loops ran nearly in interactive time as long as the rigid bodies themselves were simple. For more complex rigid bodies, the main computational cost was incurred during the contact and collision handling with the articulation enforcement adding only a small overhead. For the problems we consider with many contacts and collisions, the articulation constraints only required about 5% of the CPU time. We used the friction model described in

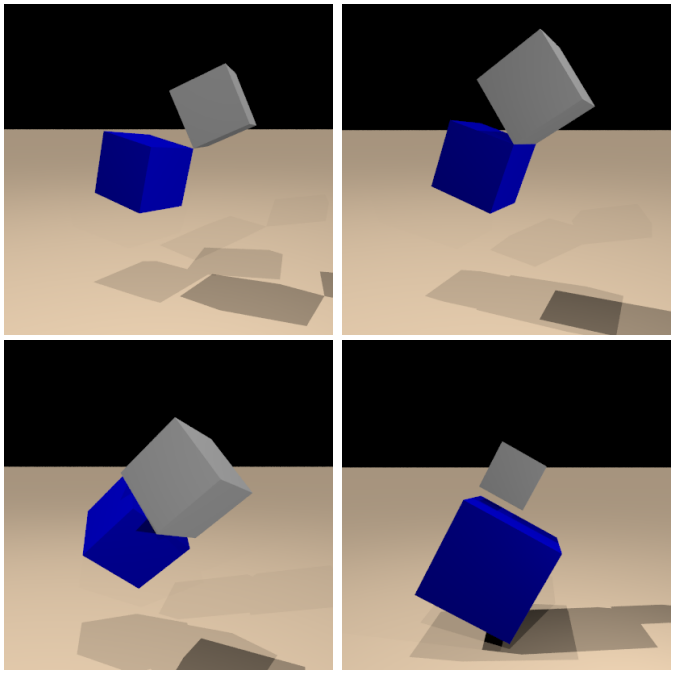

Fig. 9. Any black box joint type which can produce a desired joint frame from a given joint frame may be used with our algorithm. Examples of such joint types are point constraint joints with three degrees of freedom (top left), hinge joints with one degree of freedom (top right), twist joints with one degree of freedom (bottom left) and rigid joints with no degrees of freedom (bottom right).

[11] for friction during contact and collision. In addition, we used quite large time steps without limits based on contact and collision events or joint stability.

Our algorithm works with any black box definition for the joints and constraints, and we demonstrate some of the most basic joints in figure 9. A simple joint with three degrees of angular freedom is created by constraining two points to always remain in contact (upper left). A hinge joint may be formed by specifying that movement only be along an axis aligned with two edges (upper right). Similarly, a twist joint can be formed by placing the axis of rotation to pass through two faces (lower left). Finally, a rigid joint constrains all linear and angular movement, and holds up even when one body is much larger than the other (lower right). Prismatic joints which constrain movement in one or more of the three axial directions are demonstrated by the 210 magnets sliding down the refrigerator in figure 1. Each letter is constrained by a joint which allows prismatic movement on the face of the refrigerator as well as twisting in that plane, but no other degrees of freedom. Since the algorithm allows for any joint model, the user may choose one which incorporates internal friction or any sort of restricted movement.

Our algorithm requires no special treatment for branching or closed loops, and we show many examples of the typically more difficult closed loop case. See for example the tank in figure 7, the net in figure 2, and the bridge in figure 3 which all contain a number of closed loops and undergo frequent contact and collision. The bridge in figure 3 contains 102 rigid bodies and multiple closed loops between the different rungs in the base of the bridge as well as between the ropes and the bridge itself. To demonstrate the stability of the bridge and the efficacy of closed loops in our system, we dropped a number of objects on the bridge including 81 pairs of articulated blocks (figure 3 top) and 60 closed loop chains each containing 6 non-convex rigid bodies (figure 3 bottom). Typical simulation times for the bridge were 30 seconds per frame. Another example of complex closed loops was the net in figure 2, which is essentially a 15 by 15 lattice of rigid bodies (736 total bodies). Using over 960 constraining joints, this net was able to pick up and hold boxes or other bodies with simulation times under one minute per frame.

We also developed a tank model which took about thirty seconds a frame to simulate. The treads were each constructed as a single closed loop of 34 links using hinge joints and were wrapped around the gears. The tank uses differential steering so that the treads on each side can be moved independently allowing it to turn in place as shown in figure 4 (top). In order to prevent the treads from slipping off the gears as well as give the gears more control over the treads, we use dynamic constraints. In each frame, we use proximity to create or delete new attachment joints between the treads and the gears. Note that this is only done for the far front and far back gears which drive the tank motion, and the middle gear on each side is passively connected to the tank with a simple twist joint. Moreover, to provide for natural joint release as tread links separate from the gears, we examined each tread for unattached links and also unattached the neighbors of each of these. We experimented with driving the tank in numerous situations, such as off of a refrigerator in figure 4 (bottom), and observed good performance for the gear and tread mechanism. (Note that as the tank is falling, the spinning of the treads in one direction causes the tank to rotate in the opposite direction.)

Finally, we explored simulations involving large stacks of articulated rigid bodies. We dropped 2160 rigid bodies (with 2160 articulation constraints) modeled as 360 closed loops onto 25 poles, letting them come to rest as demonstrated in figure 6. The scene consisted of approximately 2.9 million triangles and the simulation time was approximately one day. Furthermore, we dropped twenty skeletons into a pile as shown in figure 8. Each skeleton consisted of 21 bones and 19 articulated joints for a total of 420 rigid bodies and 380 joints. With over 20 million triangles in the simulation, this scene required two days

to simulate.

VIII. Conclusions and Discussion

We proposed a novel pre-stabilization technique that iteratively finds and applies impulses to enforce articulation constraints. Moreover, we integrated this approach with a state of the art contact and collision processing algorithm illustrating the ability to robustly handle large stacks of articulated objects and complicated contact and collision scenarios such as that exhibited by a tank with gears driving treads. An additional post-stabilization technique was proposed for the velocities and angular velocities. No special treatment was required for branching, closed loops, contact, collision, friction, stacking, etc.

Our pre-stabilization method parallels [55] which addresses cloth self-collision by starting with an interference free state, and adjusting velocities to ensure that the next state will also be interference free. Post-stabilization parallels [56] which first evolves the cloth to its new possibly self-interfering state, and then attempts to untangle the resulting self-intersection. As can be seen from [56], projecting to a collision free state can be difficult or even impossible. That is, there are configurations such as edge intersections that their method does not handle. The same is true for post-stabilization of articulated rigid bodies where it is not clear how to project back onto the constraint satisfying manifold while handling contact, collisions, and interference.

A simple illustration of the difference between pre-stabilization and post-stabilization is shown in figure 10. In the left figure, the bodies are in a valid configuration. In the next time step, continuing along their current trajectories, they enter the configuration on the right. Post-stabilization allows the bodies to enter this invalid configuration on the right and then attempts to project back onto the constraint manifold. However, this requires moving body B_3 out of the way in order to satisfy contact and collision constraints. Looking only at the figure on the right, it's not clear whether B_3 should be moved to the right or to the left, and one needs to consider the figure to the left in order to see that B_3 should be moved to the right in order to avoid the appearance of passing through the other bodies. General post-stabilization methods that can project onto the constraint manifold while handling the complicated path planning needed to avoid collisions and interference as the objects are moved around do not exist. Although taking small time steps can alleviate the problem to some degree by providing less complicated states that are closer to the acceptable manifold, small time steps increase the computational burden. Instead, our prestabilization allows for big time steps and avoids this path planning scenario by staying on (or quite close to) the constraint manifold to begin with. Thus, we avoid the situation on the right of the figure, instead pushing B_3 to the right while maintaining the constraint between the other two bodies.

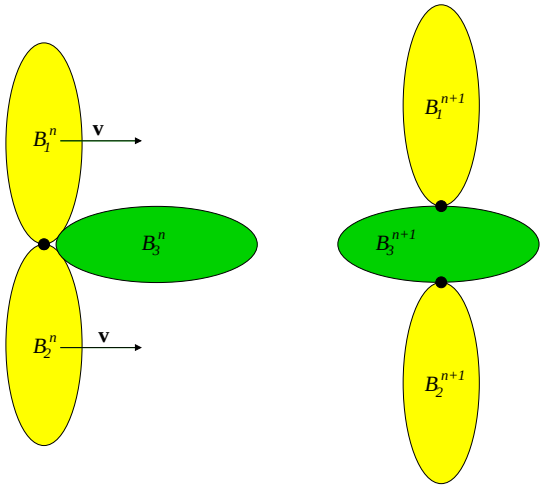

Fig. 10. This example situation highlights a difficulty with post-stabilization. The left diagram shows bodies B_1, B_2, and B_3 in a valid configuration. The bodies are moving with the given velocities at time n as shown on the left. Post-stabilization techniques will move the bodies to the positions shown on the right at time $n+1$ and then attempt to bring the articulation points, shown as black dots, back together. Our technique avoids this situation altogether by not allowing the bodies to enter the configuration on the right in the first place.

As future work, we will consider adapting our impulse based approach to a generalized coordinates model for the articulated bodies.

IX. Acknowledgements

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ARO DAAD19-03-1-0331 and NIH U54-GM072970. R.W. was supported in part by a National Science Foundation Graduate Research Fellowship. We would like to thank Mike Houston, Christos Kozyrakis, Mark Horowitz, Bill Dally and Vijay Pande for computing resources.

Appendix

Computing the partial derivative for linear constraints

Computing $\partial \mathbf{f}/\partial \mathbf{j}$ for the terms involving m_p and m_c is trivial, but dealing with the appearance of $\mathbf{j}$ in the functional argument for $\hat{\mathbf{q}}$ is more involved. To do this, first consider the action of the quaternion $\hat{\mathbf{q}}(\omega)$ on a vector $\mathbf{r}$, indicated by the

notation $\hat{\mathbf{q}}(\omega)[\mathbf{r}]$,

$$\begin{aligned}\hat{\mathbf{q}}(\omega)[\mathbf{r}] &= \mathbf{r}\left(\cos^2\theta - \sin^2\theta\right) - 2\mathbf{r}^*\mathbf{w}\cos\theta\sin\theta \\ &\quad +2(\mathbf{r}\cdot\mathbf{w})\,\mathbf{w}\sin^2\theta\end{aligned}$$

where $\theta = |\omega|/2$ and $\mathbf{w} = \omega/|\omega|$. This can be differentiated to obtain

$$\begin{aligned}\frac{\partial}{\partial \mathbf{j}}\hat{\mathbf{q}}(\omega)[\mathbf{r}] = &-4\mathbf{r}\dot{\theta}\cos\theta\sin\theta - 2\mathbf{r}^*\mathbf{w}\dot{\theta}\cos^2\theta + 2\mathbf{r}^*\mathbf{w}\dot{\theta}\sin^2\theta \\ &-2\mathbf{r}^*\dot{\mathbf{w}}\cos\theta\sin\theta + 4(\mathbf{r}\cdot\mathbf{w})\mathbf{w}\dot{\theta}\sin\theta\cos\theta \\ &+2\mathbf{w}\mathbf{r}^T\dot{\mathbf{w}}\sin^2\theta + 2(\mathbf{r}\cdot\mathbf{w})\dot{\mathbf{w}}\sin^2\theta\end{aligned}$$

where $\dot{\theta} = \partial\theta/\partial\mathbf{j}$ and $\dot{\mathbf{w}} = \partial\mathbf{w}/\partial\mathbf{j}$. For both the parent and the child terms, ω has the form $\triangle t(\omega^w)^n \pm \triangle t\mathbf{I}^{-1}\mathbf{r}^*\mathbf{j}$, and thus

$$\begin{aligned}\dot{\theta} &= \pm\left(\frac{\triangle t}{2}\right)\mathbf{w}^T\mathbf{I}^{-1}\mathbf{r}^* \\ \dot{\mathbf{w}} &= \pm\left(\frac{\triangle t}{2\theta}\right)\left(\delta - \mathbf{w}\mathbf{w}^T\right)\mathbf{I}^{-1}\mathbf{r}^*\end{aligned}$$

where δ is the identity matrix. Note that $\mathbf{r}$ is $(\mathbf{q}_p^w)^n[\mathbf{x}_j^p + \mathbf{q}_j^p[\mathbf{x}^{target}]]$ for the parent and $(\mathbf{q}_c^w)^n[\mathbf{x}_j^c]$ for the child.

Computing the partial derivative for angular constraints

To compute $\partial\mathbf{f}/\partial\mathbf{j}_\tau$, we first consider the action of the quaternion $\hat{\mathbf{q}}(\omega)$ on another quaternion $\mathbf{q} = (a, \mathbf{b})$, i.e.

$$\begin{aligned}\hat{\mathbf{q}}(\omega)\,\mathbf{q} &= (A, \mathbf{B}) \\ &= (a\cos\theta - (\mathbf{w}\cdot\mathbf{b})\sin\theta, \\ &\qquad \mathbf{b}\cos\theta + a\mathbf{w}\sin\theta - (\mathbf{b}\times\mathbf{w})\sin\theta)\end{aligned}$$

where $\theta = |\omega|/2$ and $\mathbf{w} = \omega/|\omega|$. It is more convenient to differentiate the first component separately from the last three. Thus,

$$\begin{aligned}\frac{\partial A}{\partial \mathbf{j}_\tau} &= -a\dot{\theta}\sin\theta - (\mathbf{b}\cdot\mathbf{w})\,\dot{\theta}\cos\theta - (\mathbf{b}\cdot\dot{\mathbf{w}})\sin\theta \\ \frac{\partial \mathbf{B}}{\partial \mathbf{j}_\tau} &= -\dot{\theta}\mathbf{b}\sin\theta + a\dot{\theta}\mathbf{w}\cos\theta + a\dot{\mathbf{w}}\sin\theta \\ &\quad -\dot{\theta}\mathbf{b}^*\mathbf{w}\cos\theta - \mathbf{b}^*\dot{\mathbf{w}}\sin\theta\end{aligned}$$

where $\dot{\theta} = \partial\theta/\partial\mathbf{j}_\tau$ and $\dot{\mathbf{w}} = \partial\mathbf{w}/\partial\mathbf{j}_\tau$. For both the parent and the child terms, ω has the form $\triangle t(\omega^w)^n \pm \triangle t\mathbf{I}^{-1}\mathbf{j}_\tau$, and thus

$$\begin{aligned}\dot{\theta} &= \pm\left(\frac{\triangle t}{2}\right)\mathbf{w}^T\mathbf{I}^{-1} \\ \dot{\mathbf{w}} &= \pm\left(\frac{\triangle t}{2\theta}\right)(\delta - \mathbf{w}\mathbf{w}^T)\mathbf{I}^{-1}.\end{aligned}$$

Note that $\mathbf{q}$ is $(\mathbf{q}_p^w)^n\mathbf{q}_j^p\mathbf{q}^{target}$ for the parent and $(\mathbf{q}_c^w)^n\mathbf{q}_j^c$ for the child.

REFERENCES

[1] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Proc. of ACM SIGGRAPH 1996*, 1996, pp. 137–146.

[2] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proc. SIGGRAPH 94*, 1994, pp. 23–34.

[3] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," *Comput. Graph. (Proc. SIGGRAPH 89)*, vol. 23, no. 3, pp. 223–232, 1989.

[4] D. Baraff, "Curved surfaces and coherence for non-penetrating rigid body simulation," *Comput. Graph. (Proc. SIGGRAPH 90)*, vol. 24, no. 4, pp. 19–28, 1990.

[5] D. Baraff, "Coping with friction for non-penetrating rigid body simulation," *Comput. Graph. (Proc. SIGGRAPH 91)*, vol. 25, no. 4, pp. 31–40, 1991.

[6] D. Baraff, "Issues in computing contact forces for non-penetrating rigid bodies," *Algorithmica*, no. 10, pp. 292–352, 1993.

[7] F. Faure, G. Debunne, M.-P. Cani, and F. Multon, "Dynamic analysis of human walking," in *8th Eurographics Workshop on Computer Animation and Simulation*, Sep 1997, published under the name Marie-Paule Cani-Gascuel.

[8] F. Faure, "Fast iterative refinement of articulated solid dynamics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 268–276, jul 1999.

[9] D. Ruspini and O. Khatib, "Collision/contact models for the dynamic simulation of complex environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems:IROS'97*, September 1997.

[10] D. Ruspini and O. Khatib, "A framework for multi-contact multi-body dynamic simulation and haptic display," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[11] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 871–878, 2003.

[12] B. Mirtich, "Hybrid simulation: combining constraints and impulses," in *Proc. of First Workshop on Simulation and Interaction in Virtual Environments*, July 1995.

[13] B. Mirtich and J. Canny, "Impulse-based dynamic simulation," in *Alg. Found. of Robotics*, K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, Eds. A. K. Peters, Boston, MA, 1995, pp. 407–418.

[14] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proc. of 1995 Symp. on Int. 3D Graph.*, 1995, pp. 181–188, 217.

[15] A. Witkin, K. Fleischer, and A. Barr, "Energy constraints on parameterized models," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 225–232, 1987.

[16] A. H. Barr, B. V. Herzen, R. Barzel, and J. Snyder, "Computational techniques for the self assembly of large space structures," in *Proc. 8th Princeton/SSI Conference on Space Manufacturing*. American Institute of Aeronautics and Astronautics, 1987, pp. 275–282.

[17] R. Barzel and A. H. Barr, "A modeling system based on dynamics constraints," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 179–188, 1988.

[18] J. C. Platt and A. H. Barr, "Constraint methods for flexible models," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 279–288, 1988.

[19] A. Witkin, M. Gleicher, and W. Welch, "Interactive dynamics," *Computer Graphics*, vol. 24, no. 2, pp. 11–21, 1990.

[20] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures," in *Computer Graphics (Proc. SIGGRAPH '90)*, 1990, pp. 243–252.

[21] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, pp. 1–16, 1972.

[22] U. M. Ascher, H. Chin, and S. Reich, "Stabilization of daes and invariant manifolds," *Numerische Mathematik*, vol. 67, no. 2, pp. 131–194, 1994.

[23] U. Ascher, H. Chin, L. Petzold, and S. Reich, "Stabilization of constrained mechanical systems with daes and invariant manifolds," *J. Mech. Struct. Machines*, vol. 23, pp. 135–158, 1995.

[24] U. Ascher, "Stabilization of invariants of discretized differential systems," *Numerical Algorithms*, vol. 14, pp. 1–23, 1997.

[25] J.-D. Gascuel and M.-P. Gascuel, "Displacement constraints for interactive modeling and animation of articulated structures," *The Visual Computer*, vol. 10, no. 4, pp. 191–204, 1994.

[26] J. Lee, N. Baek, D. Kim, and J. Hahn, "A procedural approach to solving constraints of articulated bodies," in *Eurographics 2000, Short Presentations Programme*, Interlaken, Switzerland, August 20-25 2000.

[27] F. Faure, "Interactive solid animation using linearized displacement constraints," in *Computer Animation and Simulation '98*, B. Arnaldi and G. Hégron, Eds., 1999, pp. 61–72.

[28] M. Cline and D. Pai, "Post-stabilization for rigid body simulation with contact and constraints." in *Proc. of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003*, Taipei, Taiwan, September 14-19 2003, pp. 3744–3751.

[29] W. Shao and V. Ng-Thow-Hing, "A general joint component framework for realistic articulation in human characters," in *Proc. of ACM Symposium on Interactive 3D Graphics*, 2003, pp. 11–18.

[30] W. Maurel and D. Thalmann, "Human upper limb modeling including scapulo-thoracic constraint and joint sinus cones," *Computers & Graphics*, vol. 24, no. 2, pp. 203–218, 2000.

[31] J. Wilhelms and A. Van Gelder, "Fast and easy reach-cone joint limits," *Journal of Graphics Tools*, vol. 6, no. 2, pp. 27–41, 2001.

[32] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster, 1987.

[33] M. Girard and A. A. Maciejewski, "Computational modeling for the computer animation of legged figures," in *Proc. of SIGGRAPH 1985*, 1985, pp. 263–270.

[34] W. W. Armstrong and M. Green, "The dynamics of articulated rigid bodies for purposes of animation," in *Graphics Interface '85*, May 1985, pp. 407–415.

[35] W. Armstrong, M. Green, and R. Lake, "Near-real-time control of human figure models," *IEEE Computer Graphics and Applications*, vol. 7, no. 6, pp. 51–61, 1987.

[36] J. Wilhelms and B. A. Barsky, "Using dynamic analysis to animate articulated bodies such as humans and robots," in *Graphics Interface '85*, May 1985, pp. 97–104.

[37] J. Wilhelms, "Using dynamic analysis for realistic animation of articulated bodies," *IEEE Computer Graphics and Applications*, vol. 7, no. 6, pp. 12–27, 1987.

[38] P. Isaacs and M. Cohen, "Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics," in *Proc. of SIGGRAPH 1987*, 1987, pp. 215–224.

[39] P. Isaacs and M. Cohen, "Mixed methods for complex kinematic constraints in dynamic figure animation," *The Visual Computer*, vol. 4, no. 6, pp. 296–305, 1988.

[40] J. K. Hahn, "Realistic animation of rigid bodies," *Comput. Graph. (Proc. SIGGRAPH 88)*, vol. 22, no. 4, pp. 299–308, 1988.

[41] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," *Comput. Graph. (Proc. SIGGRAPH 88)*, vol. 22, no. 4, pp. 289–298, 1988.

[42] J. Wilhelms, M. Moore, and R. Skinner, "Dynamic animation: Interaction and control," *The Visual Computer*, vol. 4, pp. 283–295, 1988.

[43] A. Witkin and M. Kass, "Spacetime constraints," in *Computer Graphics (Proc. SIGGRAPH '88)*, vol. 22, 1988, pp. 159–168.

[44] Z. Popović and A. Witkin, "Physically based motion transformation," in *Computer Graphics (Proc. SIGGRAPH '99)*, 1999, pp. 11–20.

[45] M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion," in *Computer Graphics (Proc. SIGGRAPH '90)*, 1990, pp. 29–38.

[46] P. Schröder and D. Zeltzer, "The virtual erector set: Dynamic simulation with linear recursive constraint propagation," in *Computer Graphics (Proc. SIGGRAPH '90)*, 1990, pp. 23–31.

[47] M. C. Surles, "An algorithm with linear complexity for interactive, physically-based modeling of large proteins," in *Proc. of SIGGRAPH '92*. ACM Press, 1992, pp. 221–230.

[48] E. Kokkevis, D. Metaxas, and N. Badler, "User-controlled physics-based animation for articulated figures," in *Proc. Comput. Anim. '96*, 1996.

[49] F. Faure, "An energy-based method for contact force computation," in *Computer Graphics Forum (Proceedings of EUROGRAPHICS'96)*, vol. 15, no. 3, Aug. 1996, pp. 357–366.

[50] J. Critchley and K. Anderson, "A generalized recursive coordinate reduction method for multibody dynamic systems," *Journal of Multiscale Computational Engineering*, vol. 1, no. 2, pp. 181–200, 2003.

[51] K. Anderson and J. Critchley, "Improved order-n performance algorithm for the simulation of constrained multi-rigid-body systems," *Multibody Systems Dynamics*, vol. 9, pp. 185–212, 2003.

[52] J. Lenoir and S. Fonteneau, "Mixing deformable and rigid-body mechanics simulation," in *Computer Graphics International*, june 16-19 2004, pp. 327–334.

[53] G. Baciu and S. K. Wong, "The impulse graph: a new dynamic structure for global collisions," *CGForum*, vol. 19, no. 3, pp. 229–238, 2000.

[54] S. Redon, Y. Kim, M. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," in *ACM Symposium on Solid Modeling and Applications 2004*, 2004.

[55] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 28–36.

[56] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, pp. 862–870, 2003.