Soft Articulated Characters in Projective Dynamics

Jing Li, Tiantian Liu, and Ladislav Kavan

Abstract—We propose a fast and robust solver to simulate continuum-based deformable models with constraints, in particular, rigid-body and joint constraints useful for soft articulated characters. Our method embeds the degrees of freedom of both articulated rigid bodies and deformable bodies in one unified constrained optimization problem, thus coupling the deformable and rigid bodies. Inspired by Projective Dynamics which is a fast numerical solver to simulate deformable objects, we also propose a novel local/global solver that takes full advantage of the pre-factorized system matrices to accelerate the solve of our constrained optimization problem. Therefore, our method can efficiently simulate character models, with rigid-body parts (bones) being correctly coupled with deformable parts (flesh). Our method is stable because backward Euler time integration is applied to both rigid and deformable degrees of freedom. Our unified optimization problem is rigorously derived from constrained Newtonian mechanics. When simulating only articulated rigid bodies as a special case, our method converges to the state-of-the-art rigid body simulators.

Index Terms—rigid body, deformable body, coupling, Projective Dynamics.

1 Introduction

Realistic animation of articulated characters plays an important role in computer games, virtual reality and other interactive applications. The animation of articulated characters often models the body as a collection of rigid bodies ("Ragdoll physics") which can be computed quickly. However, virtual characters are usually based on real-world creatures, which often have a deformable external part (flesh) that interacts with the environment (e.g., finger grasping). This external part is coupled with a rigid internal structure (skeleton) that provides support. For example, believable animations of a human-like character, a snail, or a mermaid require simulation of bones or similar rigid structures, often further constrained with joints because two adjacent bones (e.g. upper and lower arm) usually cannot move arbitrarily. The mechanical interplay between rigid and deformable bodies results in numerically challenging simulation problems.

Projective Dynamics [1] is an implicit Euler solver used in real-time deformable object simulation. It exploits a special potential energy structure which enables an efficient local/global solver, which often outperforms the classic Newton's method, making Projective Dynamics a suitable option for real-time simulation of deformable objects. However, Projective Dynamics (PD) assumes continuum-based elastic models. In theory, rigidity of the bones can be achieved by increasing the stiffness of the rigid parts. In practice however, this does not work very well, because with high stiffness ratios the convergence of PD deteriorates and more computationally expensive Newton-type solvers are recommended [2]. The problem is that the global step of PD can only translate groups of vertices (e.g. four vertices in a tetrahedron), but not rotate them (the same problem is present also in Position Based Dynamics). Therefore, constraints with high stiffness effectively "lock" the orientation. In this paper, we avoid this problem by linearizing the $SE(3)$ manifolds, enabling exact (i.e. infinitely stiff) rigid-body constraints without locking. In Figure 1, we demonstrate an example of the locking problem of Projective Dynamics and demonstrate how our proposed approach avoids it.

In this work, which is an extended version of a paper presented at the Symposium on Computer Animation 2019 [3], We propose a fast simulation framework for soft articulated characters. Our method is derived from constrained formulation of Newtonian dynamics by using backward Euler time integration. In terms of the numerical solution, we show that we can combine the principles of Projective Dynamics and constrained dynamics into a unified optimization problem.

Our contributions are as follows:

- We propose a unified method to simulate deformable characters with articulated skeletons in Projective Dynamics. Our method is fast and rigorously derived from Newton's laws via backward Euler time integration.
- We show that the special structure of potential energy from Projective Dynamics retains its numerical benefits when we reorganize the computations in a certain way. In addition to the standard linear solve required by Projective Dynamics, our global step introduces several smaller solves with system matrices whose sizes are independent of the mesh resolution and are instead proportional to the number of bones or joints.

- *J. Li is with School of Computing, University of Utah, Salt Lake City, UT, United States and with Advanced Innovation Center For Future Visual Entertainment, Beijing Film Academy, Beijing, China. E-mail: jingli2070769@gmail.com*
- *T. Liu is with Microsoft Research Asia, Beijing, China. E-mail: ltt1598@gmail.com*
- *L. Kavan is with School of Computing, University of Utah, Salt Lake City, UT, United States. E-mail: ladislav.kavan@gmail.com*

Manuscript received April 19, 2005; revised August 26, 2015.

- The vertices corresponding to rigid bodies are represented by only the 6-DOFs used in classical rigid-body dynamics, representing translation and rotation in three-dimensional spaces. Our formulation is monolithic, i.e., there is no separate treatment of rigid and deformable degrees of freedom, transfer of momenta etc., simplifying implementation.
- The rigidity constraints and joint constraints are considered to be satisfied given that our method is shown to converge empirically.

Our method and the method proposed in [3] share the same problem formulation (Eq. 6). Both methods reuse the prefactorization technique from Projective Dynamics. The difference lies in how we solve the constrained optimization problem (Eq. 6). Our method uses *Lagrange multipliers* followed by Schur complement which naturally partitions the deformable body from rigid body and joint constraints while our previous method [3] isolates the joint constraints in an additional optimization problem from the elastic energy.

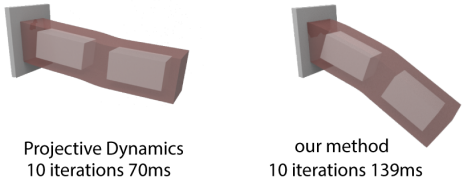


Fig. 1: Imitating rigid bodies by higher stiffness in Projective Dynamics results in locking (left), while our method (right) produces the correct result. The ratio of stiffness of "flesh" and "bones" is 1:500.

2 Related Work

2.1 Deformable body simulation

Deformable bodies can be simulated by mass-spring systems [4], finite difference method [5], finite element method (FEM) [6], [7], finite volume method (FVM) [8], boundary element method (BEM) [9] etc. Position Based Dynamics (PBD) [10] is one of the popular methods to simulate deformable bodies in real-time applications. PBD assumes infinitely stiff energy potentials and the material stiffness depends on iteration count and time step, which is problematic in a scene with objects of varied stiffness, e.g.: soft bodies interacting with nearly rigid bodies [11]. PBD is solved in a Gauss-Seidel fashion, which is stable and easy to implement, but does not converge rapidly. Extended position-based dynamics (XPBD) introduces a total Lagrange multiplier to PBD, which converges to the actual solution of backward Euler integration with physically correct stiffness. But similar to PBD, XPBD suffers from slow convergence speed of Gauss-Seidel iteration and low accuracy of only first order approximation of backward Euler integrator. Those shortcomings of PBD and XPBD are addressed by Projective Dynamics (PD) [1] with rigor and higher accuracy from continuum mechanics. Projective Dynamics was derived as an extension of ShapeUp [12] to dynamics. ShapeUp is an optimization system for geometry processing which also supports rigid-body constraints, but these are only soft constraints and dynamics is not considered. Increasing the stiffness of the rigid-body constraints would result in similar deterioration of convergence as in Projective Dynamics [2].

Projective Dynamics can be seen as a quasi-Newton method [13] or an alternating direction method of multipliers (ADMM) [14], and therefore supports more general materials. Fast GPU implementations of Projective Dynamics are also possible. Those methods can be further accelerated using Chebyshev semi-iterative approach [15], a colored Gauss-Seidel method [16], or a hyper-reduce scheme [17]. Kugelstadt et al. [18] follow a similar strategy of Projective Dynamics by using an operator splitting approach to speed up the stretching resistance part. Moreover, it formulates the resistance to volume change as compliant constraints and introduces analytic polar decomposition(APD) to compute the rotational part of the deformation gradient. Soler et al. [19] simulate Cosserat rods with Projective Dynamics by incorporating body orientation in standard PD solver. Peng et al. [20] speed up the convergence rate of Projective Dynamics for an accurate solution by applying Anderson acceleration, a well-established technique for fixed point iteration method.

2.2 Articulated rigid body simulation

Perfect rigid bodies do not exist in our real world. The concept of the rigid body is merely an idealization to simplify the computation. There is a large body of classical work in computer graphics to simulate articulated rigid bodies [21], [22], [23], as described in the detailed survey by J. Bender et al. [24]. Armstrong and Green [25] incorporate dynamics into the model of the human figure, given the forces and torques applied to joints at key points in the animation. Weinstein et al. [26] propose an iterative solution for handling joint and large numbers of unpredictable contact and collision events for rigid articulated bodies.

Witkin and Kass [27] propose a general tool to create articulated character animation by using spacetime constraints. Zordan et al. [28] incorporate motion capture data to make the articulated skeleton respond to unexpected impact.

2.3 Coupling between rigid and deformable bodies

Coupling rigid and deformable bodies is an interesting topic that has been explored by many authors [29], [30], [31], [32], [33]. One common approach is to simulate each subsystem with specialized technique and then bridge the two together [34]. Specifically, Shinar et al. [34] designed a full two-way coupling of rigid and deformable bodies. It consists of 5 major steps to interleave the rigid and deformable body simulation in addition to the simulation of both the rigid and deformable degrees of freedom. ArtiSynth [35], an open source package used for modeling and simulating complex anatomical systems, is composed of both rigid and deformable bodies. It uses FEM to simulate deformable bodies and uses multibody techniques [36] to simulate rigid bodies separately, and couple these two parts together using attachment constraints. It is also common to simulate characters via physically-based skinning [37], [38], [39], [40], [41], however, with these methods the motion of the skeleton

is specified kinematically, i.e., is not subject to physics-based simulation. Galoppo et al. [42] combines articulated-body dynamics and skin deformation, which is expressed in pose space (rest configuration). It then applies displacement corrections from deformation to skinning. However, this method is not suitable for large global deformations, such as highly flexible characters. Tournier et al. [2] propose an offline method for high stiffness material and constraints. Tournier et al. [2] also pointed out that Projective Dynamics results in artificial damping or even locking if the relative material stiffness is too high. Instead of using complete geometric stiffness [43], Projective Dynamics only keeps the constant term in the geometric stiffness. Kim and Pollard [44] simulate skeleton-driven deformable body characters, which uses mesh embedding to reduce the DOFs of the deformable bodies. Verschoor et al. [45] define the shape of the bone implicitly as a capsule. The distance between the representative point on the bone and the interpolation point (from 4 vertices in a tetrahedron) is minimized as the energy term, whereas our method can simulate bones of various shape such as skulls. Sifakis et al. [33] proposed a framework to augment mesh-based simulation of deformable body with point-based simulation technology. This hybrid framework can be extended to simulate two-way coupling of rigid and deformable bodies, but it is not applicable to scenarios involving complex contact, stacking, friction and articulation. O'Brien et al. [30] proposed a modular approach to couple passive objects and the actively controlled main character. It focuses on specific issues to couple passive and active systems. This method is a force-based method where a constraint force that prevents further penetration, a stabilizing damping force that absorbs a portion of the impact energy, and a restoring force corrects any penetration error are all included. Recently, Wang et al. [46] proposed an efficient and flexible approach for computing rigid body dynamics where FEM nodes and their attachment to rigid bodies can be incorporated as constraints. Their method is a fully implicit two-way coupled method, but not a monolithic method, thus stabilization is needed to mitigate the unavoidable constraint drift. They designed a preconditioned conjugate gradient solver, which is faster than direct solvers at the problem sizes of around 1000 degrees of freedom.

2.4 Monolithic Methods

The philosophy of a monolithic simulation system of both rigid and deformable bodies is explored by many authors. For example, Barraff and Witkin [29] proposed a modular approach to combine different simulation systems, each of which is treated as a black box. The interaction between systems is handled as constraint forces, which are approximated from velocities by making the system step forward. The procedure of the method treats the two systems asymmetrically. Therefore, it works best when there is significant mass disparity between the two systems. In the method of Jansson and Vergeest [31], the rigid body has both the rigid body representation and the deformable body representation as mass-spring system, which entails a number of conversions between the deformable body representation and the rigid body representation. One advantage of both [31] and our method is DOF reduction in the simulation, caused by the rigid body idealization. All the examples shown in [31] consist of only a very small number of particles, whereas our method can simulate a significantly larger system fast because our method is sped up by Projective Dynamics. Lenoir and Fontenea [32] models both the deformable and rigid body as constraints: the deformable body as inter constraints and the rigid body as inner constraints, both of which are solved by Lagrange Multipliers. Both [32] and our method treats rigid body dynamics as constraints. The difference is that Lenoir and Fontenea [32] use Newton-Euler formalism for the rigid constraints while our method minimizes the inertia term and constrain the transformation to $SE(3)$. While these two approaches to realize the rigid constraints are intrinsically the same, our method is more convenient because there's no need to compute the inertia of the rigid body explicitly when the shape of the rigid body is irregular. Both [32] and our method decomposes the full system matrix into smaller blocks. But the way of decomposing is completely different. Recently, Li et al. [3] proposed a real-time simulation method to couple rigid and deformable bodies in Projective Dynamics framework. This method can be a good candidate for real-time application of rigid-deformable body coupling because the computation overhead is small. However, this method solves the joint constraint in a separate step where the deformable body is ignored temporarily, which could lead to disharmony between the rigid and deformable body in the next iteration. In contrast, our method solves for unknown of the deformable body and joint constraints together in one global solve, which conciliates the deformable body, rigid body and the joint constraints better.

3 Background

3.1 Constrained Dynamics

Continuous equations of motion with constraints can be written as [47]:

$$\mathbf{M}\mathbf{a}(t) = -\nabla E(\mathbf{x}(t)) + \nabla C(\mathbf{x}(t))^{\mathsf{T}}\lambda(t) + \mathbf{f}_{\text{ext}} \tag{1a}$$

$$C(\mathbf{x}(t)) = 0 \tag{1b}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{x}(t), \mathbf{v}(t), \mathbf{a}(t) \in \mathbb{R}^{3n\times 1}$ denote the position, velocity and acceleration respectively. $-\nabla E(\mathbf{x}(t))$ and $\mathbf{f}_{\text{ext}} \in \mathbb{R}^{3n\times 1}$ represent the internal and external forces. And $\lambda(t) \in \mathbb{R}^{n_c\times 1}$ is a vector of Lagrange multipliers which correspond to the system constraints $C = \mathbf{0}$.

We integrate our system according to the backward Euler integration rules:

$$\begin{aligned} \mathbf{x}_{n+1} - \mathbf{x}_n &= h\mathbf{v}_{n+1} \\ \mathbf{v}_{n+1} - \mathbf{v}_n &= h\mathbf{a}_{n+1} \end{aligned} \tag{2}$$

where the subscript $n+1$, n and $n-1$ denotes the future, current and previous state respectively, and h is the timestep size. By substituting Eq. 2 into Eq. 1a, we obtain a discretized version root finding problem:

$$\frac{\mathbf{M}}{h^2}(\mathbf{x} - \mathbf{y}) = -\nabla E(\mathbf{x}) + \nabla C(\mathbf{x})^{\mathsf{T}}\lambda \tag{3}$$

where we drop the subscript of the only unknown variable $\mathbf{x}_{n+1}$ and denote it as a shorthand $\mathbf{x}$. $\mathbf{y} := 2\mathbf{x}_n - \mathbf{x}_{n-1} +$

$h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ is a constant vector that aggregates the known values.

Together with $C(\mathbf{x}) = 0$, Eq. 3 can be interpreted as setting the gradient of the Lagrangian of the following constrained optimization problem to zero:

$$\min_{\mathbf{x}} \quad \frac{1}{2h^2}\|\mathbf{x}-\mathbf{y}\|^2_{\mathbf{M}} + E(\mathbf{x}) \tag{4a}$$
$$\text{s.t.} \quad C(\mathbf{x}) = 0 \tag{4b}$$

3.2 Projective Dynamics

Projective Dynamics [1] treats a deformable body simulation as an unconstrained optimization problem as described in Eq. 4a and solves it using a local/global solver. For each element i, the Projective Dynamics energy $E_i(\mathbf{x}, \mathbf{p}_i)$ is of the form $\frac{1}{2}\|\mathbf{G}_i\mathbf{x} - \mathbf{p}_i\|^2_F$ where $\mathbf{G}_i$ is a discrete finite differential operator. Projective Dynamics re-formulates Eq. 4a into a larger optimization problem with both $\mathbf{x}$ and $\mathbf{p}$ as the system degrees of freedom.

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\mathbf{x}^\mathsf{T}\left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{x} - \mathbf{x}^\mathsf{T}\left(\frac{\mathbf{M}}{h^2}\mathbf{y} + \mathbf{J}\mathbf{p}\right) + \mathbf{b} \tag{5}$$

where $\mathbf{L} = \sum w_i \mathbf{G}_i^\mathsf{T}\mathbf{G}_i$ and $\mathbf{J} = \sum w_i \mathbf{G}_i^\mathsf{T}\mathbf{S}_i$ (w_i is a weight scalar for each element and $\mathbf{S}_i$ is a selector matrix) are two state-independent matrices which can be pre-computed. $\mathbf{b}$ is a constant vector irrelevant to the minimization problem and can be therefore dropped. See [13] and [48] for more details on $\mathbf{L}$ and $\mathbf{J}$. Projective Dynamics iteratively solves for $\mathbf{x}$ and $\mathbf{p}$ by alternating between fixing $\mathbf{x}$ and computing $\mathbf{p}$ in the local step and fixing $\mathbf{p}$ and computing $\mathbf{x}$ in the global step.

4 METHOD

4.1 Problem Formulation

We can formulate our articulated deformable body simulation as a constrained optimization problem by instantiating $C(\mathbf{x}) = 0$ in Eq. 4b with rigid body and joint constraints,

$$\min_{\mathbf{x}} \quad \frac{1}{2h^2}\|\mathbf{x}-\mathbf{y}\|^2_{\mathbf{M}} + E(\mathbf{x}) \tag{6a}$$
$$\text{s.t.} \quad \mathbf{T}_k \in SE(3) \quad \text{for k = 1...m} \tag{6b}$$
$$\mathbf{T}_j\mathbf{q} = \mathbf{T}_k\mathbf{q} \quad \text{if the } j\text{- and } k\text{-th bones share a joint } \mathbf{q} \tag{6c}$$

where $\mathbf{T}_k \in \mathbb{R}^{3\times4}$ is implicitly dependent on the vertex positions $\mathbf{x}$, representing the transformation matrix of bone k; $\mathbf{q} \in \mathbb{R}^{4\times1}$ is the rest-pose position of the shared joint connecting bone j and bone k, represented in homogeneous coordinates; m is the number of bones. Eq. 6c means that a joint transformed in the frame of bone k should be in the same position after being transformed in the frame of bone j when bone k and j share the joint $\mathbf{q}$. There is no need for explicit handling of rigid-deformable or rigid-rigid body interaction because all physical interactions are implicitly modeled as internal forces between individual elements (tetrahedron). The rigid body dynamics are also taken into account by having an inertia term in the objective for the rigid body part. Currently, our method only supports ball joints formulated by Eq. 6c. Our method treats linearized rigid constraints as bilateral constraints and solves the overall problem as a constrained optimization problem while in [3], the violation of rigid constraints is penalized by solving a separate minimization problem.

Once the constrained optimization problem Eq. 6 is formulated, one can use any off-the-shelf optimization method to solve it. We tried a primal dual interior point method using IPOPT [49] and got a pleasant result which can be seen in Figure 9. However, using an interior point method to solve our optimization problem could be excessive for two reasons. First, the degrees of freedom corresponding to the rigid vertices in Eq. 6a is redundant because every bone needs only 6 DOFs to perform a rigid body transformation instead of the DOFs of the actual number of vertices to model the bone. Second, our joint constraints in Eq. 6c are linear and the rigid body constraints in Eq. 6b can be linearized using special treatments [50], which invite a better numerical solution to solve the constrained problem. These observations enable us to accelerate the solve of Eq. 6 using our modified local/global solver.

4.2 DOF Reduction

Let us first represent the position vector as $\mathbf{x} = [\mathbf{x}_f; \mathbf{x}_b]$, where $\mathbf{x}_f \in \mathbb{R}^{3n_f\times1}$ and $\mathbf{x}_b \in \mathbb{R}^{3n_b\times1}$ are the positions of the flesh vertices and bone vertices respectively. The notation n_f and n_b denotes the number of vertices of the flesh and the bone. As we stated before, the bone vertices $\mathbf{x}_b$ do not need that many degrees of freedom, because no matter how many vertices are used to model one bone, they all move under the same rigid body transformation matrix which only has 6 DOFs – 3 for translation and 3 for rotation. Note that a linear function is not sufficient to map the positions of the bone vertices to a rigid body transformation because of the rotation DOFs. To remedy this, we design a two-step DOF reduction scheme where the first step groups all bone vertices to move in the same affine transformation space, and the second step further reduce the DOFs of the affine transformation to translation and linearized rotation.

In the first step, we simply want all vertices belong to the same bone to move under one single transformation matrix – not necessarily a rigid body transformation matrix yet. The relationship between the bone vertices and their corresponding transformation can be written as:

$$\mathbf{x}_{b_k,i} = \mathbf{T}_k\mathbf{V}_{b_k,i} \tag{7}$$

where $\mathbf{x}_{b_k,i} \in \mathbb{R}^{3\times1}$ is the i-th vertex on bone k, $\mathbf{V}_{b_k,i}$ is the homogeneous coordinate of restpose position of i-th vertex on bone k and $\mathbf{T}_k$ is the transformation matrix as defined in Eq. 6b.

Note that the transformation matrix $\mathbf{T}_k$ has 12 DOFs instead of 6 because it encodes some unwanted DOFs like shearing or scaling, we can further reduce the transformation DOFs to 6 by representing $\mathbf{T}_k$ as a combination of translation and linear rotation as our second step:

$$\mathbf{T}_k = (\mathbf{I}_3 + \omega_k^*)\,\mathbf{T}_k^r + \begin{bmatrix}\mathbf{0}_3 & \ell_k\end{bmatrix} \tag{8}$$

where $\mathbf{T}_k^r$ represent the closest rigid body transformation matrix to $\mathbf{T}_k$, initialized to an identity transformation matrix $\begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times1}\end{bmatrix}$ at the rest-pose configuration. $\mathbf{I}_3$ and $\mathbf{0}_3$ are 3×3 identity and zero matrices respectively. The translation vector $\ell_k \in \mathbb{R}^{3\times1}$ and the linearized rotation vector $\omega_k \in R^{3\times1}$ become the actual variables to determine $\mathbf{T}_k$.

The superscript * turns the vector ω_k into a skew-symmetric cross-product matrix. This parameterization further restricts the space of $\mathbf{T}_k$ from a 12-DOF affine space to a 6-DOF tangent space of $SE(3)$ at state $\mathbf{T}_k^r$. The choice of $\mathbf{T}_k^r$ is updated with the system. We will elaborate it at Section 4.3.

If we apply Eq. 8 to Eq. 7, we can rewrite $\mathbf{x}_{b_k,i}$ as a function of ω_k and ℓ_k:

$$\mathbf{x}_{b_k,i} = \underbrace{\begin{bmatrix}-[\mathbf{T}_k^r\mathbf{V}_{b_k,i}]^* & \mathbf{I}_3\end{bmatrix}}_{\tilde{\mathbf{V}}_{k,i}} \begin{bmatrix}\omega_k \\ \ell_k\end{bmatrix} + \underbrace{[\mathbf{T}_k^r\mathbf{V}_{b_k,i}]}_{\mathbf{V}_{k,i}^0} \tag{9}$$

We can keep stacking Eq. 9 to replace the original system variable $\mathbf{x}$ in Eq. 6 as follows:

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_f \\ \mathbf{x}_b\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{I}_{n_f} & 0 \\ 0 & \tilde{\mathbf{V}}\end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix}\mathbf{x}_f \\ \mathbf{s}\end{bmatrix}}_{\tilde{\mathbf{x}}} + \mathbf{V}^0 \tag{10}$$

where $\mathbf{I}_{n_f}$ is an $n_f \times n_f$ identity matrix, $\mathbf{s}$ is the stack of $\begin{bmatrix}\omega_k \\ \ell_k\end{bmatrix}$ for all rigid bodies, $\tilde{\mathbf{V}} \in \mathbb{R}^{3n_b \times 6m}$ is a block diagonal matrix whose diagonal entries are stacks of $\tilde{\mathbf{V}}_{k,i}$ in the same bone, and $\mathbf{V}^0 \in \mathbb{R}^{3n\times 1}$ is the stack of $\mathbf{V}_{k,i}^0$. Both of $\tilde{\mathbf{V}}$ and $\mathbf{V}^0$ depend only on the rest-pose position of the bone vertices and the rigid body transformation matrices $\mathbf{T}_k^r$. The detailed layout of $\tilde{V}$ and $\mathbf{V}^0$ can be seen in the appendix. The original system variable $\mathbf{x} \in \mathbb{R}^{(3n_f+3n_b)\times 1}$ is now linearly controlled by our reduced system variable $\tilde{\mathbf{x}} \in \mathbb{R}^{(3n_f+6m)\times 1}$.

4.3 Local/global Solver

Even in a reduced space, the optimization problem in Eq. 6 is still nonlinear due to the rigid body constraints (Eq. 6b). We resort to Projective Dynamics [1] as an accelerator to solve this problem. The key idea of Projective Dynamics is to isolate all the nonlinear components of an optimization into mutually-independent small problems in its local step, leaving only a linear problem to solve in the global step. Inspired by this, we can first reformulate our objective function Eq. 6a by plugging Eq. 10 into Eq. 5 in the reduced space as:

$$\begin{aligned}\min_{\tilde{\mathbf{x}},\mathbf{p}} \quad & \frac{1}{2}\tilde{\mathbf{x}}^\mathsf{T}\underbrace{\mathbf{B}^\mathsf{T}\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{B}}_{\mathbf{Q}}\tilde{\mathbf{x}} \\ & + \tilde{\mathbf{x}}^\mathsf{T}\underbrace{\mathbf{B}^\mathsf{T}\left(\left(\frac{\mathbf{M}}{h^2}+\mathbf{L}\right)\mathbf{V}^0 - \left(\frac{\mathbf{M}}{h^2}\mathbf{y}+\mathbf{J}\mathbf{p}\right)\right)}_{\mathbf{c}}\end{aligned} \tag{11}$$

where the nonlinearity in the elastic energy implicitly grouped into the auxiliary variable $\mathbf{p}$. Now all the nonlinear components in our constrained problem are isolated in the projection vector $\mathbf{p}$ and the rigid body transformation matrices $\mathbf{T}_k^r$, which naturally invites a local/global solver to solve it.

Local step, Our first nonlinear component is contained in $\mathbf{p}$ which is a stacked vector of projections from the deformation gradient of each deformable element to a wanted manifold such as $SO(3)$ or $SL(3)$. Another nonlinear component is the rigid body transformation matrix $\mathbf{T}_k^r$ for each bone. Similar to Projective Dynamics, we run a signed singular value decomposition (SSVD) on the current deformation gradients of the flesh elements and on the first three columns of the current transformation matrices $\mathbf{T}_k$ for the bone vertices to evaluate $\mathbf{p}$ and $\mathbf{T}_k^r$. Note that all those projections are independent and involve only 3×3 SSVDs, we can execute them efficiently in batch.

Global step, The objective of our global step is already defined in Eq. 11, we further want to handle the constraints in Eq. 6 together in this step. By substituting $\mathbf{x}$ in Eq. 6c with Eq. 10, we get the linear constraints of the minimization problem with reduced DOF as:

$$\underbrace{\begin{bmatrix}-[\mathbf{T}_j^r\mathbf{q}]^* & \mathbf{I} & [\mathbf{T}_k^r\mathbf{q}]^* & -\mathbf{I}\end{bmatrix}}_{\mathbf{A}} \begin{bmatrix}\omega_j \\ \ell_j \\ \omega_k \\ \ell_k\end{bmatrix} = \underbrace{-\mathbf{T}_j^r\mathbf{q} + \mathbf{T}_k^r\mathbf{q}}_{\mathbf{b}} \tag{12}$$

With the simplified notations defined in Eq. 11 and Eq. 12, we can combine them into a linearly constrained quadratic optimization problem as our global step:

$$\begin{aligned}\min_{\tilde{\mathbf{x}}} \quad & \frac{1}{2}\tilde{\mathbf{x}}^\mathsf{T}\mathbf{Q}\tilde{\mathbf{x}} + \mathbf{c}^\mathsf{T}\tilde{\mathbf{x}} \\ \text{s.t.} \quad & \mathbf{A}\tilde{\mathbf{x}} = \mathbf{b}\end{aligned} \tag{13}$$

Note that we released the nonlinear constraint Eq. 6b in our global step Eq. 13. This is because Eq. 6b is implicitly handled in the local step where $\mathbf{T}_k$ is projected onto $SE(3)$ to get $\mathbf{T}_k^r$. When a solution of Eq. 6 is found, our global step Eq. 13 also reaches its optimal value since we restrict the transformations of the bones to stay at the tangent spaces of $SE(3)$. In other words, our global step is not able to break the rigid body constraints for the bone transformations from a converged state of Eq. 6. We therefore solely rely on our local step to handle the rigidity constraints in Eq. 6b.

4.4 Numerical Solution for Global Step

The special structure (quadratic objective and linear constraints) of Eq. 13 enables us to solve it using a closed form solution. We can apply *Lagrange multipliers* to Eq. 13, yielding

$$\mathcal{L}(\tilde{\mathbf{x}},\boldsymbol{\lambda}) = \frac{\mathbf{1}}{\mathbf{2}}\tilde{\mathbf{x}}^\mathsf{T}\mathbf{Q}\tilde{\mathbf{x}} + \mathbf{c}^\mathsf{T}\tilde{\mathbf{x}} + (\mathbf{A}\tilde{\mathbf{x}}-\mathbf{b})^\mathsf{T}\boldsymbol{\lambda} \tag{14}$$

where the first-order optimality condition of the Lagrange multipliers implies the following KKT system:

$$\begin{bmatrix}\mathbf{Q} & \mathbf{A}^\mathsf{T} \\ \mathbf{A} & 0\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}} \\ \lambda\end{bmatrix} = \begin{bmatrix}-\mathbf{c} \\ \mathbf{b}\end{bmatrix} \tag{15}$$

Solving Eq. 15 would optimize our global step in one shot. One can solve it using any iterative solver like minimum residual method or direct solver based on Cholesky factorization. However, keeping in mind that the majority of $\mathbf{Q}$ matrix is very likely to be constant, we are able to even accelerate it using a pre-factorization strategy like Projective Dynamics. To further investigate this possibility, we divide $\mathbf{Q}$ into 4 blocks along the boundary between the flesh vertices and bone vertices and rewrite Eq. 15 as:

$$\underbrace{\left[\begin{array}{c|cc}\mathbf{Q}_f & \multicolumn{2}{c}{\mathbf{Q}_c^\mathsf{T}} \\ \hline \mathbf{Q}_c & \mathbf{Q}_b & \mathbf{A}_b^\mathsf{T} \\ & \mathbf{A}_b & \end{array}\right]}_{\text{KKT matrix}} \left[\begin{array}{c}\mathbf{x}_f \\ \hline \mathbf{s} \\ \lambda\end{array}\right] = \left[\begin{array}{c}-\mathbf{c}_f \\ \hline -\mathbf{c}_s \\ \mathbf{b}\end{array}\right] \tag{16}$$

Algorithm 1: local/global solver to solve Eq. 6

Initialize $\mathbf{x}_f^{(1)}$, $\mathbf{s}^{(1)}$ and $\mathbf{T}_k^{(1)}$ from the previous frame.
for $i = 1, \ldots,$*max iteration count* **do**
  Local Step: *fix* $\tilde{\mathbf{x}}$,
  compute $\mathbf{p}$ *using [1];*
  compute $\mathbf{T}_k^r$ *by projecting* $\mathbf{T}_k^{(i)}$, *for* $k = 1 \ldots m$.
  Global Step: *fix* $\mathbf{p}$ *and* $\mathbf{T}_k^r$ *for* $k = 1 \ldots m$,
  compute $\mathbf{x}_f^{(i+1)}$, $\mathbf{s}^{(i+1)}$, $\lambda^{(i+1)}$ *using Eq. 17.*
  Update: $\tilde{\mathbf{x}}^{(i+1)} = \begin{bmatrix} \mathbf{x}_f^{(i+1)} \\ \mathbf{s}^{(i+1)} \end{bmatrix}$.
  Update: $\mathbf{T}_k^{(i+1)} = \left(\mathbf{I}_3 + \omega_k^{*(i+1)}\right)\mathbf{T}_k^r + \begin{bmatrix} \mathbf{0}_3 & \ell_k^{(i+1)} \end{bmatrix}$.
$\mathbf{x} = \mathbf{B}\tilde{\mathbf{x}}$.

where the KKT matrix is a $(3n_f + 6m + 3n_{jp}) \times (3n_f + 6m + 3n_{jp})$ matrix that consists of $\mathbf{Q}_f \in \mathbb{R}^{3n_f \times 3n_f}$, $\mathbf{Q}_c \in \mathbb{R}^{6m \times 3n_f}$, $\mathbf{Q}_b \in \mathbb{R}^{6m \times 6m}$ and $\mathbf{A}_b \in \mathbb{R}^{3n_{jp} \times 6m}$. $\mathbf{c}_f \in \mathbb{R}^{3n_f \times 1}$ is the top $3n_f$ rows in $\mathbf{c}$ that corresponds to the flesh vertices and $\mathbf{c}_s \in \mathbb{R}^{6m \times 1}$ is the bottom $6m$ rows. As defined in Eq. 11, $\mathbf{Q}_f = \mathbf{M}/h^2 + \mathbf{L}$ is a constant matrix from Projective Dynamics, we can split the KKT matrix in Eq. 16 into constant and dynamics parts. The constant Cholesky factorization of $\mathbf{Q}_f$ can be reused when we solve Eq. 16 using Schur complement:

$$\lambda = \left(\mathbf{A}_b\mathbf{S}^{-1}\mathbf{A}_b^\mathsf{T}\right)^{-1}\left(-\mathbf{b} + \mathbf{A}_b\mathbf{S}^{-1}\left(-\mathbf{c}_s + \mathbf{Q}_c\mathbf{Q}_f^{-1}\mathbf{c}_f\right)\right) \tag{17a}$$

$$\mathbf{s} = \mathbf{S}^{-1}\left(-\mathbf{c}_s + \mathbf{Q}_c\mathbf{Q}_f^{-1}\mathbf{c}_f - \mathbf{A}_b^\mathsf{T}\lambda\right) \tag{17b}$$

$$\mathbf{x}_f = \mathbf{Q}_f^{-1}\left(-\mathbf{c}_f - \mathbf{Q}_c^\mathsf{T}\mathbf{s}\right) \tag{17c}$$

where the upper case $\mathbf{S}$ is the Schur complement: $\mathbf{S} = \mathbf{Q}_b - \mathbf{Q}_c\mathbf{Q}_f^{-1}\mathbf{Q}_c^\mathsf{T}$. For a typical soft articulated character, the number of joints and the number of rigid bones are much smaller compared to the number of flesh vertices, hence the dimension of the linear system to solve for λ and $\mathbf{s}$ is small. Therefore we are able to compute and factorize $\mathbf{S}$ explicitly in every global step. Thanks to the constant matrix $\mathbf{Q}_f$, we only need to factorize it once and can reuse its factor throughout the entire simulation.

4.5 Summary

To summarize our method, we solve Eq. 6 by alternating between the local and global steps. In the local step, we compute $\mathbf{p}$ by projecting deformation gradient to a desired manifold and compute $\mathbf{T}_k^r$ by projecting $\mathbf{T}_k$ to $SE(3)$. In the global step, we solve Eq. 13 with fixed $\mathbf{p}$ and $\mathbf{T}_k^r$ using Eq. 17 to find the middle ground that accommodates the nonlinearity. We summarize our algorithm in Alg. 1. The superscript inside parenthesis indicates the iteration number.

5 PIPELINE OVERVIEW

The pipeline of our method starts with constructing a conforming mesh of the rigid (bone) and deformable (flesh) bodies. We use one .obj file to define the closed surface of the exterior and another .obj file to define the closed surface of the interior skeleton. Our framework takes the two .obj files as input and outputs a .smesh file which tetgen [51] can turn into conforming (Delaunay) tetrahedralized mesh with a different regional attribute for each enclosed region. From this .mesh file, we know exactly which vertices belong to which bone region and which vertices belong to the flesh region. In addition, we created a .joint file specifically in our framework to keep track of the affiliation of all the rigid bones and joints. By combining the .joint and .mesh file, we have a complete model of the soft articulated character. This process of constructing the model of soft articulated character is illustrated in Figure 2. We show the model of a human, a frog and a Thordyke in Figure 3, where the deformable parts and the rigid parts (bone) are colored pink and white respectively. The red dots represent the joint positions. The deformable parts are simulated using a linear co-rotational model in all the examples shown in Section 6.

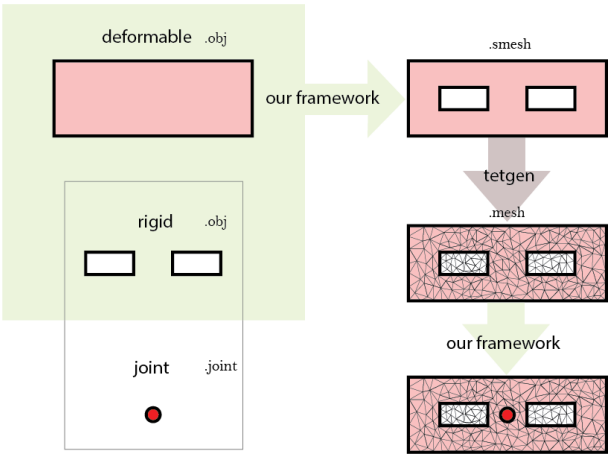


Fig. 2: Schematic workflow of mesh construction

Fig. 3: Visualization of the interior of our models. The deformable and rigid components are colored using pink and white respectively. The joints are illustrated using red dots.

To handle contact, we use the collision method proposed in [52], which computes collision forces based on consistent n-body penetration depth information. We add the collision forces in $\mathbf{f}_{\text{ext}}$ in Eq. 1. We accelerate collision detection with spatial hashing [52] as well.

6 RESULTS

Table 1 summarizes the settings and timings for the examples we used in the paper. All examples are executed on

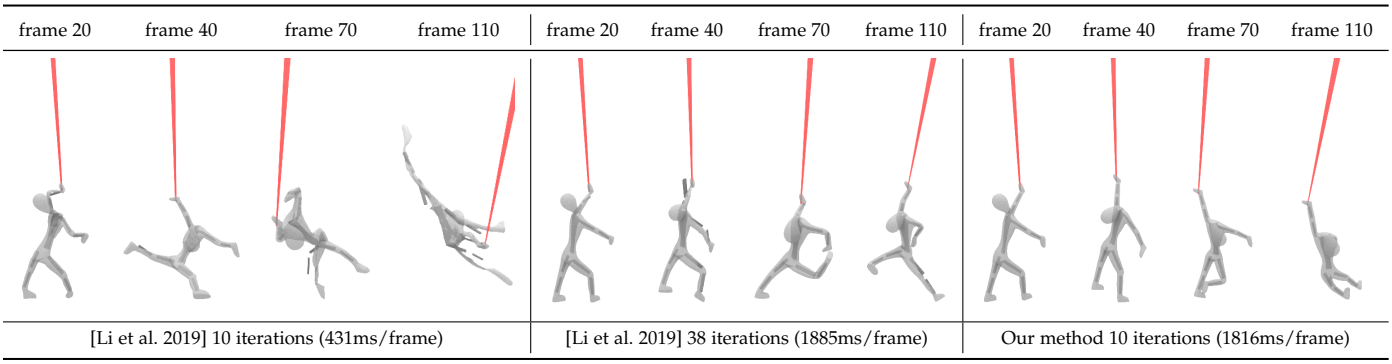


Fig. 4: The artifact of bones sticking outside of the body is sometimes observed in the method proposed in [3] when the stiffness of the material is very low. More iterations in [3] does not always cure the artifact. With our method, this artifact is never observed.

TABLE 1: Results on our example models

Example	#Verts		#Elems	#Bones	#Joints	Time			
	Elastic	Total				PD [1]	[3]	IPOPT [49]	Ours
cantilever	1010	1690	7661	2	1	70ms	73ms	8370ms	139ms
bead bracelet	0	1603	4638	12	12	40ms	62ms	3495ms	51ms
nunchuck	0	14073	66748	2	1	712ms	706ms	6074ms	791ms
aerial silk	11617	12683	45888	9	8	431ms	449ms	9681s	1816ms
frog	24615	29466	121630	18	13	1.140s	1.241s	30686s	5.75s
snail	14318	28840	105694	1	0	1.029s	1.027s	580s	1.795s
Thorndyke hanging	29623	37689	137312	59	9	1.319s	2.337s	>10h	16.958s
Thorndyke rolling	29623	37689	137312	59	9	1.317s	1.672s	>10h	17.347s

TABLE 2: Timing breakdown on our example models

Example	Local Step		Global Step				Total
	$\mathbf{p}$	$\mathbf{T}_k^r$	$\mathbf{Q}_f^{-1}\mathbf{Q}_c^\mathsf{T}$	$\mathbf{S}^{-1}$	$\mathbf{Q}_f^{-1}\mathbf{c}_f$	others	
cantilever	9.264ms	0.1ms	4.091ms	0.245ms	0.225ms	0.009ms	13.964ms
bead bracelet	4.927ms	0.155ms	–	0.027ms	–	–	5.109ms
nunchunck	76.955ms	2.164ms	–	0.027ms	–	–	79.145ms
aerial silk	52.364ms	0.191ms	103.627ms	20.145ms	4.218ms	1.127ms	181.673ms
frog	111.627ms	0.591ms	343.882ms	107.791ms	7.873ms	3.382ms	575.145ms
snail	144.955ms	2.5ms	24.536ms	1.136ms	6.1ms	0.236ms	179.464ms
Thorndyke hanging	126.764ms	0.919ms	1175.3ms	370.609ms	10.073ms	12.145ms	1695.81ms
Thorndyke rolling	127.055ms	0.927ms	1213.8ms	370.727ms	10.127ms	12.136ms	1734.772ms

an Intel®Core™ i7-8750H CPU. We experimented different methods and recorded their timings as shown in the "Time" columns in Table 1. PD time denotes the simulation time of our base-line deformable body simulator Projective Dynamics [1] without considering any rigid or joint constraints. All other methods simulate both the deformable and rigid bodies at the same time. [3] time is the timing for our previous method [3], which is slightly slower compared with PD. IPOPT time denotes the solving time for Eq. 6 using an off-the-shelf interior point solver [49]. At last, the run time cost of our method is shown in the last column. To make a fair comparison in performance, we run 10 local/global iterations for all the examples simulated using [1], [3] and our method. Our method has comparable performance with Projective Dynamics in simple cases, like our previous method [3]. In complicated cases with both more deformable elements and more rigid/joint constraints, our method runs an order of magnitude slower than PD, to achieve more accurate and physically-correct results. We show the accuracy of our method both qualitatively and quantitatively later in this section.

To better assess the performance bottleneck of our method, we list the timing breakdown for a single local/global iteration, as shown in Table 2, where we note down the timings to execute line 4 and 5 in Alg. 1 as the local step cost and the timings to assemble the building blocks to compute line 7 in Alg. 1 as the global step cost. As we can see from Table 2, the main extra overhead of our method compared with PD is to compute the matrix $\mathbf{Q}_f^{-1}\mathbf{Q}_c^\mathsf{T}$ in Eq. 17. Although $\mathbf{Q}_f$ is a constant matrix in PD, we still do not want to compute and store its inverse explicitly because the inverse of $\mathbf{Q}_f$ is dense. We instead pre-factorized $\mathbf{Q}_f$ using Cholesky factorization and treated the assembly of $\mathbf{Q}_f^{-1}\mathbf{Q}_c^\mathsf{T}$ as one linear solve with multiple right-hand-side vectors. Since $\mathbf{Q}_c^\mathsf{T}$ is a sparse matrix, we also used MUMPS [53] to accelerate this linear solve. We apply the same trick to solve for the sparse linear solve $\mathbf{S}^{-1}$ after assembling it. We report the time of assembly and solving for $\mathbf{S}$ under the column "$\mathbf{S}^{-1}$" in Table 2. Since the typical size of $\mathbf{S}$ is much smaller compared to $\mathbf{Q}_f$, the solving for $\mathbf{S}$ is also less expensive compared to computing $\mathbf{Q}_f^{-1}\mathbf{Q}_c^\mathsf{T}$. Our method can simulate soft bodies with both interior and exterior rigid parts, whereas the previous fast simulation methods usually focus on only on rigid part or the soft part. Ragdoll physics

alone cannot capture the secondary motion of soft bodies, for example, the belly of the frog shown in Figure 5 where a deformable frog with human-like skeleton inside is tumbled inside a rotating box. Projective Dynamics, does not respect the rigidity of an object, for example, the shell of a snail shown in Figure 6. Of course we can forge the rigid parts in Projective Dynamics by tuning their stiffness to extremely high. However, this leads to a locking behavior as shown in the middle of Figure 6 where the shell of the snail barely rotates. Our method is able to simulate both the soft and rigid components in a unified framework without penalizing the rigid body motion, producing vivid secondary motions for the belly of the frog and and the soft part of the snail, while maintaining the rigid and joint constraints. Please refer to our accompanying video for more details.

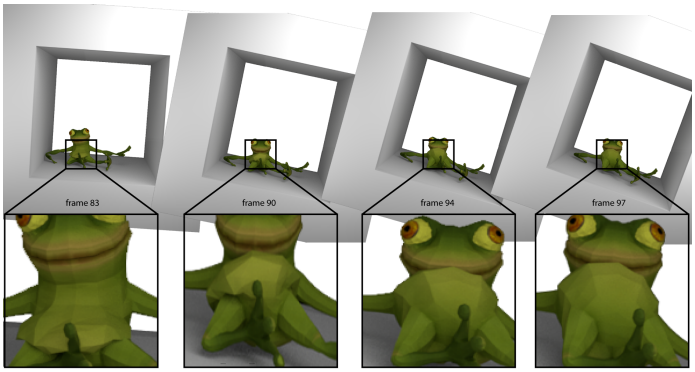


Fig. 5: A frog is tumbled in a rotating box. The legs are bent in a way that the thigh and calf are almost straight and they rotate around the knees, exhibiting similar effects of ragdoll physics. At the same time, the secondary motion (belly shake) from the deformable body is also present in our result.

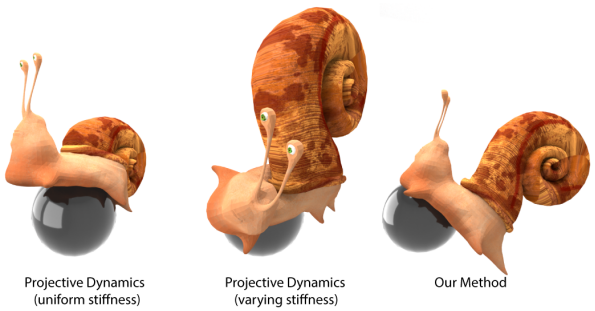


Fig. 6: A falling snail collides with a sphere. Our method (right) keeps the shell rigid while Projective Dynamics with uniform stiffness (left) collapses it and Projective Dynamics with varying stiffness (middle) suffers from a locking artifact.

Our previous method [3] is a good attempt towards a fast and physically accurate soft and rigid body simulator, but it has problems in some cases. In Figure 4, a deformable acrobat with rigid skeleton inside is dragged by a red ribbon. The leftmost sequences show the result of our previous method [3], which didn't take the elastic energy into consideration when solving the joint constraint optimization. In some challenging cases where the stiffness of the deformable body is very small, artifacts such as bones penetrating through the surface or even explosions are sometimes observed. With the same number of local-global iterations, our method, in contrast, does not have this artifact as shown on the right of Figure 4. To make it a fair comparison, we also tried to increase the number of iterations of [3] to match the total time cost of our method, as shown in the middle of Figure 4. This helps mitigating this artifact, but can not remove it completely.

We also tested if our method will fail dramatically when not converged completely using the same setting in Figure 4. We do not observe any bone sticking out of the flesh, when reducing the number of local/global iterations from 20 to 10, 5 and 1. However, we do observe the overall motion of the acrobat is more damped when given only a few local/global iterations compared to a converged solution. To better view the differences of our method running with different iterations, we show a top view of the same acrobat in Figure 7. Lowering the local/global iteration count will not cause any artifacts other than the artificial damping inherited from Projective Dynamics. In fact, under those low-iteration-count settings, we found our rigid and joint constraints even easier to satisfy because of the more artificially damped motion. Therefore, we safely choose 10 local/global iterations to run all our examples, similarly to Projective Dynamics. Also like Projective Dynamics, our method inherits the artificial damping from the implicit Euler time integration scheme. When simulated using larger timesteps, our method will deliver more damped results as shown in Figure 8.

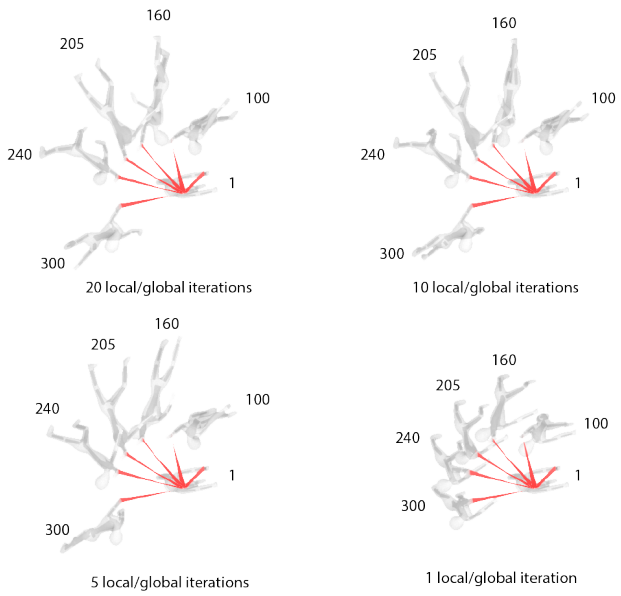


Fig. 7: The top view of aierial silk when using different local/global iteration counts. Frame 1, 100, 160, 205, 240 and 300 are chosen from the simulation and overlapped in the picture. We can see that with only 1 local/global iteration, the simulation is highly damped so that the swing circle is much smaller since the acrobat stays closer to the rest-pose when he loses his velocity fast.

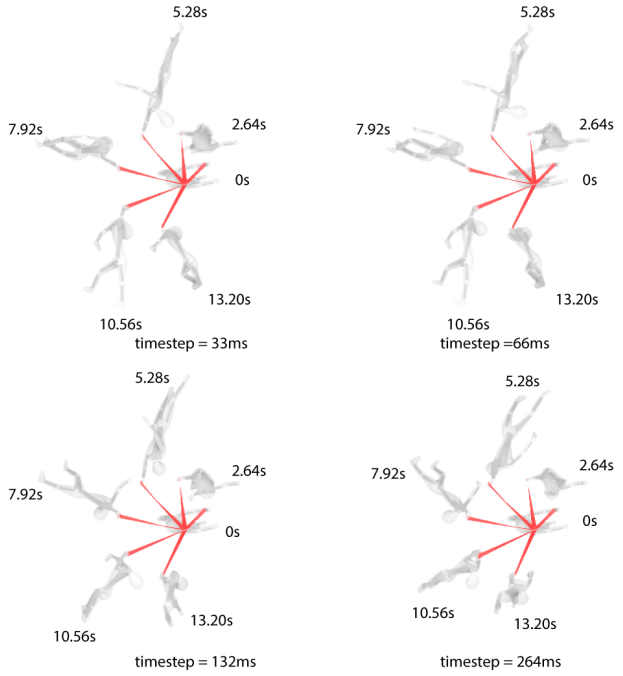


Fig. 8: The top view of aierial silk when using different time steps (all using 20 local/global iterations). 0s, 2.64s, 5.28s, 7.92s, 10.56s and 13.20s indicate the time elapsed from the starting frame. The swing circle gets smaller when the timestep is larger.

In order to validate the accuracy of our method, we tested a deformable cantilever with two rigid "bones" inside it as shown in Figure 9. The ground truth solution was obtained by IPOPT as shown on the left in Figure 9. The middle figure shows the result of our method. We also tried imitating the rigid bodies by setting the stiffness of the white bones extremely large and solving the system using stable constrained dynamics [2] which is able to solve poor-conditioned system with compliant constraints. The result is shown on the right in Figure 9. In this simple example, the result of our method is visually indistinguishable from the result of IPOPT, which can be considered as ground truth. When measured quantitatively by the total energy, our method produces a very similar but not identical energy plot as IPOPT (Figure 11). Our method avoids injecting energy artificially as [3]. However, the computation cost of our method is only 139 ms to produce a frame compared to 8370 ms using IPOPT. The result of stable constrained dynamics differs from the ground truth solution because it only uses a first order operator to approximate the nonlinear solution of the compliant-constrained problems.

Our method empirically converges to the results produced by IPOPT, which we consider as ground truth, as shown in Figure 10. The relative error is defined as:

$$\frac{||\mathbf{x}^{(i)} - \mathbf{x}^*||}{||\mathbf{x}^{(0)} - \mathbf{x}^*||} \tag{18}$$

where $||\cdot||$ is the Euclidean distance, $\mathbf{x}^{(i)}$ is the solution for ith iteration, $\mathbf{x}^*$ is the solution from IPOPT. In the 3 experiments shown in Figure 10, we simulated the first 30 frames all using IPOPT and simulated the 31st frame using

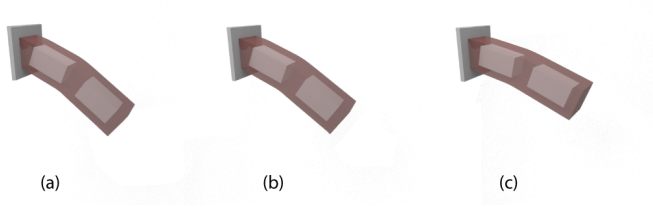


Fig. 9: A cantilever with two "bones" inside. Our method (b) gives visually indistinguishable result as IPOPT (a) while stable constrained dynamics (c) exhibits different movement.

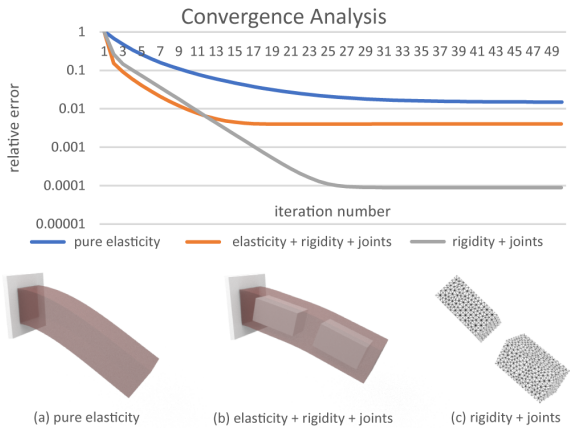


Fig. 10: Our method is shown to converge empirically. The horizontal axis is the number of local/global iterations we run for our method and the log-scaled vertical axis is the relative error our method produces compared to the ground truth solution simulated using IPOPT. The blue graph corresponding to the bottom left configure (a) simply treats the entire cantilever as a uniform deformable bar and the gray graph corresponding to the bottom right configure (c) is simulating the two bones and one joint in configure (b) without any deformable elements.

our method to produce the plots of relative error. Note that when simulating a pure deformable body as shown in the blue graph of Figure 10, our method degenerates to vanilla Projective Dynamics. In most of our test cases we experience similar convergence behavior with Projective Dynamics which reduces the error fast at the beginning and quickly slows down afterwards.

We tested our method on an interesting model called Thorndyke, as shown in Figure 12. In this example, the character is pinned at one point on the horn and swings under gravity. The Thorndyke model is very complicated: the horns, teeth, and nails are all rigid bodies in additional to possessing an internal rigid skeleton. Our method exhibits realistic simulation of both the internal and external rigid bodies. The method proposed in [3] cannot handle it very well and the simulation under the same configuration exploded in the third frame. The computation time in Table 1 of [3] is the average of only the first two frames.

Figure 13 shows that our method can exhibit richer and more realistic animation than ragdoll physics in the classic falling-down-stairs scene. The snout of Thorndyke is squashed when it touches the stair. In the meantime, the

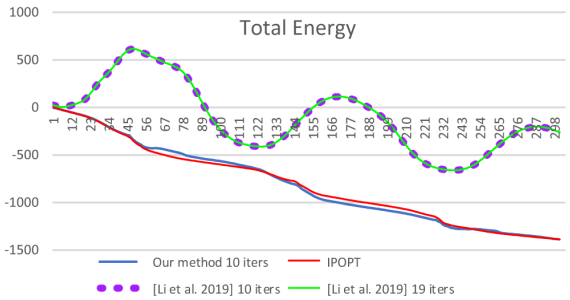


Fig. 11: The total energy of the cantilever example in Figure 9, generated by different methods. The purple line shows the total energy with the method of [3] using 10 iterations (70ms) and the green line shows the energy with the method of [3] with 19 iterations (142ms).The blue line shows the total energy of our method using 10 iterations (139ms). The red line shows the total energy by using IPOPT. Given more iterations to the method of [3], the energy artifact is not fixed. The total energy is a sum of kinetic energy, gravitational potential energy and elastic potential energy.

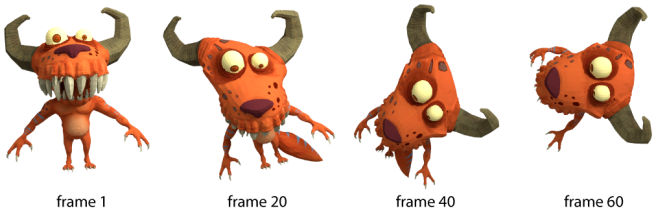


Fig. 12: Our method can be used to simulate external rigid parts like the horns, teeth and nails in Thorndyke.

horns, teeth, nails and the skeleton inside move as rigid bodies, adding realism to the animation. Our method is able to handle the collisions of the Thorndyke both on its soft part and its exterior rigid components.

Under extreme cases where all vertices are set as rigid vertices, our method degenerates to a rigid body simulator. In Figure 14, a nunchuck consisting of two rigid blocks($2 \times 2 \times 4$ each) and a joint connecting them is pinned at one point and is falling under gravity. The rightmost figure shows the result by using IBDS [54] [55], which simulates multi-body system of rigid bodies, particles, many different joint types and collisions with dynamic and static friction. The timestep is 0.1 second in all nunchuck experiments. The color of the figures shows the heat map of difference between our method and IBDS. Our method behaves differently with IBDS because we use backward Euler as our time integrator since Projective Dynamics, where our method is based on, is derived from backward Euler, whereas IBDS uses an explicit one. In order to remove the difference caused by the time integrator, we sub-step both our method and IBDS by 1, 10 and 100, as shown in Figure 14. When we remove the effects of different integrators by sub-stepping, our method converges to the results obtained by the state-of-the-art rigid body simulator perfectly.

Our method naturally handles closed loops which are considered challenging cases that require special treatment

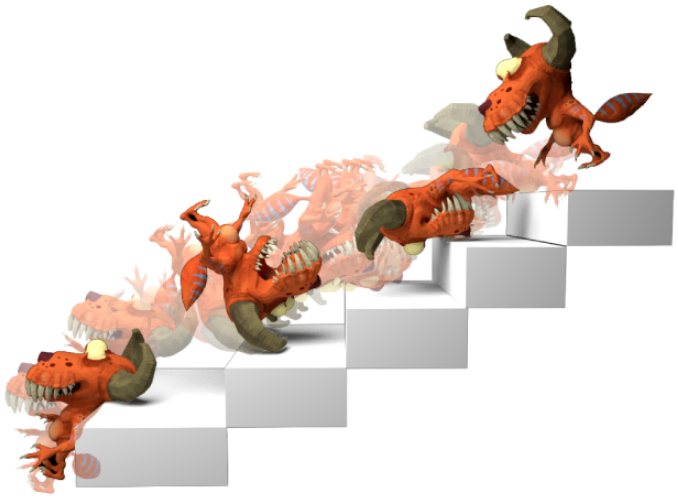


Fig. 13: Thorndyke rolls down the stairs.

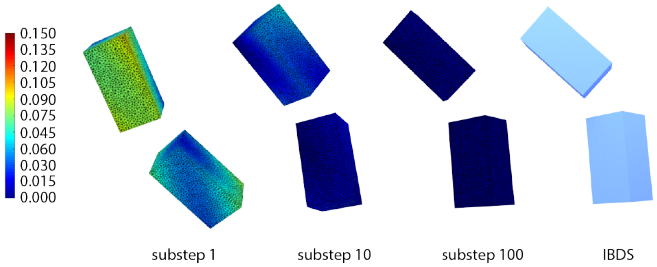


Fig. 14: The 820th frame of a simulation by droping the nunchuck while pinned at a point. The color is the heat map of the difference between our method and IBDS. Our method converges to rigid body dynamics if a sufficiently small timestep is used.

[24]. Figure 15 shows a bead bracelet which consists of 12 beads as rigid bodies and 12 joint constraints connecting the 12 beads in a loop. Our method simulates the bracelet easily without transforming the loop into a tree and adding additional forces to maintain the loop structure.

7 Limitation and Future Work

Our method inherits the artifacts from Projective Dynamics like the numerical damping due to insufficient convergence. It is possible to accelerate our method using some advanced methods designed for Projective Dynamics, such as [13] and [20]. The ball joint constraints prevent bones from separating, but each bone can still rotate arbitrarily far, which is not true for most biological joints. An interesting avenue for future work would involve adding the support of joint limits. Other types of joint constraints: prismatic, universal, cylindrical, screw or spline, which we don't support now, can provide the ability to animate more complex and interesting characters, therefore, are of great research value in the future as well. In all our examples, the characters are passive, i.e., not actuated. It would be possible to extend our method with artist directed control to our physics-based system. Spacetime constraints [27] offer control over the animation

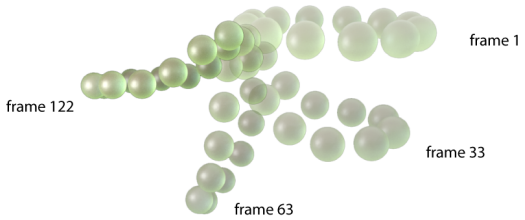


Fig. 15: A bead bracelet falling down under gravity while one bead is held by attachment constraint.

and add physical realism with desired secondary motion effects. State-of-the-art character control using spacetime constraints supports only articulated bodies [27] or purely deformable bodies [56]. In the future it would be interesting to extend spacetime constraints to characters with soft flesh and rigid skeletons.

8 Conclusions

We introduced a new monolithic method to simulate articulated soft characters based on constrained dynamics. Our method is accelerated using similar local/global strategies in Projective Dynamics and is simple to implement; in particular, no additional code is required to explicitly handle the coupling between rigid and deformable bodies. Our method is suitable for simulating characters with deformable bodies, rigid bones and joints, which is often required for realistic creatures. We believe our new articulated/deformable simulating method will find use in computer games or training simulators.

Acknowledgments

We thank Eftychios Sifakis for many inspiring discussions. We also thank Yasmin Down for Thorndyke modelling, and Henry Rietra for snail modelling, Dimitar Dinev for proofreading. This work was supported in part by National Key R&D Program of China [grant number 2018YB1403900] and the National Science Foundation under Grant Numbers IIS-1617172, IIS-1622360 and IIS-1764071. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge the support of Activision, Adobe, and hardware donation from NVIDIA Corporation.

References

[1] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: fusing constraint projections for fast simulation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 154, 2014.

[2] M. Tournier, M. Nesme, B. Gilles, and F. Faure, "Stable constrained dynamics," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 132, 2015.

[3] J. Li, T. Liu, and L. Kavan, "Fast simulation of deformable characters with articulated skeletons in projective dynamics," in *Symposium on Computer Animation*, 2019.

[4] Y. Chen, Q. hong Zhu, A. E. Kaufman, and S. Muraki, "Physically-based animation of volumetric objects," in *CA*, 1998.

[5] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *ACM Siggraph Computer Graphics*, vol. 21, no. 4, pp. 205–214, 1987.

[6] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr, "Dynamic real-time deformations using space & time adaptive sampling," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 31–36.

[7] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler, "Stable real-time deformations," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2002, pp. 49–54.

[8] J. Teran, S. Blemker, V. Hing, and R. Fedkiw, "Finite volume methods for the simulation of skeletal muscle," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2003, pp. 68–74.

[9] D. L. James and D. K. Pai, "Artdefo: accurate real time deformable objects," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 65–72.

[10] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[11] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*. ACM, 2016, pp. 49–54.

[12] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly, "Shape-up: Shaping discrete geometry with projections," in *Computer Graphics Forum*, vol. 31, no. 5. Wiley Online Library, 2012, pp. 1657–1667.

[13] T. Liu, S. Bouaziz, and L. Kavan, "Quasi-newton methods for real-time simulation of hyperelastic materials," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 23, 2017.

[14] R. Narain, M. Overby, and G. E. Brown, "ADMM $\supseteq$ projective dynamics: Fast simulation of general constitutive models," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '16. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2016, pp. 21–28. [Online]. Available: http://dl.acm.org/citation.cfm?id=2982818.2982822

[15] H. Wang, "A chebyshev semi-iterative approach for accelerating projective and position-based dynamics," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 246, 2015.

[16] M. Fratarcangeli, V. Tibaldo, and F. Pellacini, "Vivace: A practical gauss-seidel method for stable soft body dynamics," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 214, 2016.

[17] C. Brandt, E. Eisemann, and K. Hildebrandt, "Hyper-reduced projective dynamics," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 80:1–80:13, 2018.

[18] T. Kugelstadt, D. Koschier, and J. Bender, "Fast corotated fem using operator splitting," *Computer Graphics Forum*, vol. 37, pp. 149–160, 12 2018.

[19] C. Soler, T. Martin, and O. Sorkine-Hornung, "Cosserat rods with projective dynamics," *Computer Graphics Forum (proceedings of SCA issue)*, vol. 37, no. 8, 2018.

[20] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, and L. Liu, "Anderson acceleration for geometry optimization and physics simulation," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 42, 2018.

[21] R. Featherstone, *Robot Dynamics Algorithm*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

[22] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 137–146.

[23] R. L. Weinstein, "Simulation and control of articulated rigid bodies," Ph.D. dissertation, STANFORD UNIVERSITY, 2007.

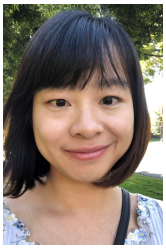
[24] J. Bender, K. Erleben, and J. Trinkle, "Interactive simulation of rigid body dynamics in computer graphics," in *Computer Graphics Forum*, vol. 33, no. 1. Wiley Online Library, 2014, pp. 246–270.

[25] W. W. Armstrong and M. W. Green, "The dynamics of articulated rigid bodies for purposes of animation," *The visual computer*, vol. 1, no. 4, pp. 231–240, 1985.

[26] R. Weinstein, J. Teran, and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 3, pp. 365–374, May 2006. [Online]. Available: https://doi.org/10.1109/TVCG.2006.48

[27] A. Witkin and M. Kass, "Spacetime constraints," in *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '88. New York, NY, USA: ACM, 1988, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/54852.378507

[28] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast, "Dynamic response for motion capture animation," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 697–701.
[29] D. Baraff and A. Witkin, "Partitioned dynamics," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1997.
[30] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins, "Combining active and passive simulations for secondary motion," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp. 86–96, 2000.
[31] J. Jansson and J. S. Vergeest, "Combining deformable-and rigid-body mechanics simulation," *The Visual Computer*, vol. 19, no. 5, pp. 280–290, 2003.
[32] J. Lenoir and S. Fonteneau, "Mixing deformable and rigid-body mechanics simulation," in *Proceedings Computer Graphics International, 2004.* IEEE, 2004, pp. 327–334.
[33] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw, "Hybrid simulation of deformable solids," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2007, pp. 81–90.
[34] T. Shinar, "Simulation of coupled rigid and deformable solids and multiphase fluids," Ph.D. dissertation, Stanford, CA, USA, 2008, aAI3313660.
[35] J. E. Lloyd, I. Stavness, and S. Fels, "Artisynth: A fast interactive biomechanical modeling toolkit combining multibody and finite element simulation," in *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, pp. 355–394.
[36] A. A. Shabana, *Dynamics of Multibody Systems*, 3rd ed. Cambridge University Press, 2005.
[37] L. Liu, K. Yin, B. Wang, and B. Guo, "Simulation and control of skeleton-driven soft body characters," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 215, 2013.
[38] S. Capell, M. Burkhart, B. Curless, T. Duchamp, and Z. Popović, "Physically based rigging for deformable characters," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2005, pp. 301–310.
[39] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 37, 2011.
[40] F. Hahn, S. Martin, B. Thomaszewski, R. Sumner, S. Coros, and M. Gross, "Rig-space physics," *ACM transactions on graphics (TOG)*, vol. 31, no. 4, p. 72, 2012.
[41] L. Kavan and O. Sorkine, "Elasticity-inspired deformers for character articulation," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 196, 2012.
[42] N. Galoppo, M. A. Otaduy, S. Tekin, M. Gross, and M. C. Lin, "Soft articulated characters with fast contact handling," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 243–253.
[43] S. Andrews, M. Teichmann, and P. G. Kry, "Geometric Stiffness for Real-time Constrained Multibody Dynamics," *Computer Graphics Forum*, vol. 36, no. 2, 2017.
[44] J. Kim and N. S. Pollard, "Fast simulation of skeleton-driven deformable body characters," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 5, p. 121, 2011.
[45] M. Verschoor, D. Lobo, and M. A. Otaduy, "Soft-hand simulation for smooth and robust natural interaction," 2018. [Online]. Available: http://gmrv.es/Publications/2018/VLO18
[46] Y. Wang, N. J. Weidner, M. A. Baxter, Y. Hwang, D. M. Kaufman, and S. Sueda, "REDMAX: Efficient & flexible approach for articulated dynamics," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 104:1–104:10, July 2019.
[47] C. Lanczos, *The Variational Principles of Mechanics*, ser. Dover Books On Physics. Dover Publications, 1986. [Online]. Available: https://books.google.com/books?id=ZWoYYr8wk2IC
[48] J. Li, T. Liu, and L. Kavan, "Laplacian damping for projective dynamics," in *VRIPHYS2018: 14th Workshop on Virtual Reality Interaction and Physical Simulation*, 2018.
[49] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.
[50] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, vol. 3, 2010.
[51] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 2, p. 11, 2015.
[52] B. H. Heidelberger, *Consistent collision and self-collision handling for deformable objects*. Eidgenössische Technische Hochschule [ETH] Zürich, 2007.
[53] P. R. Amestoy, J.-Y. L'Excellent, and G. Moreau, "On exploiting sparsity of multiple right-hand sides in sparse direct solvers," *SIAM Journal on Scientific Computing*, vol. 41, no. 1, pp. A269–A291, 2019.
[54] J. Bender and A. Schmitt, "Constraint-based collision and contact handling using impulses," in *Proceedings of the 19th international conference on computer animation and social agents*, Geneva (Switzerland), Jul. 2006, pp. 3–11.
[55] ——, "Fast dynamic simulation of multi-body systems using impulses," in *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, Madrid (Spain), Nov. 2006, pp. 81–90.
[56] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, "Animating deformable objects using sparse spacetime constraints," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 109, 2014.



Jing Li is currently a Ph.D. student at the University of Utah, working on physics-based simulation with professor Ladislav Kavan. She was previously a software engineer at MModal and received her master of Science from Carnegie Mellon University.

Tiantian Liu is currently a researcher at Microsoft Research Asia. His research focus is mainly on physically based simulation and fast optimization methods. He previously received his Ph.D. and M.Sc.Eng. degree from the University of Pennsylvania, and B.Eng. degree from Zhejiang University.

Ladislav Kavan is an associate professor of computer science at the University of Utah. Prior to joining Utah, he was an assistant professor at the University of Pennsylvania and research scientist at Disney Interactive Studios. Ladislav's research focuses on interactive computer graphics, physics-based animation, and geometry processing. His goal is to combine computer graphics with biomechanics and medicine. Ladislav is a member of the ACM SIGGRAPH community.