Two-Way Coupling of Skinning Transformations and Position Based Dynamics

Yuhan Wu
The University of Tokyo

Nobuyuki Umetani
The University of Tokyo

(a) Input (b) ARAP (c) FAST (d) Ours

Figure 1: (a) The user inputs animation by dragging and fixing control points. The degrees of freedom specified by user inputs are colored black. (b) As-rigid-as possible (ARAP) deformation simply rotates the whole character with no deformation. (c) Automatic fast skinning transformations (FAST) can deform the character but without any dynamic effects. (d) Our method creates dynamic rigged animation in real time with a position-based physics simulation running in the background while satisfying the user's specifications.

ABSTRACT

Skinning transformations allow digital characters to be animated with minimal user inputs. Physics simulations can improve the detailed dynamic movement of the animated character; however, such details are typically added in the post-processing stage after the overall animation is specified. We propose a novel interactive framework that unifies skinning transformations and kinematic simulations using position-based dynamics (PBD). Our framework allows an arbitrarily skinned character to be partially manipulated by the user, and a kinematic physics solver automatically complements the behavior of the entire character. We achieve this by introducing new steps in the PBD algorithm, (i) lightweight optimization to identify the skinning transformations, which is similar to inverse kinematics, and (ii) a position-based constraint to restrict the PBD solver in the complementary subspace of the skinning deformation. Our method combines the best of the two methods: the controllability and shape preservation of the skinning transformation and the efficiency, simplicity, and unconditional stability of the PBD solver. Our interface allows novices to create vibrant animations without the need for tedious editing.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

position based dynamics, complementary dynamics, skinning transformation

ACM Reference Format:
Yuhan Wu and Nobuyuki Umetani. 2023. Two-Way Coupling of Skinning Transformations and Position Based Dynamics. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Character animation is a labor-intensive art form; thus, computer graphics researchers have been studying techniques that facilitate this process. Skinning transformation allows the shape of a character to be controlled by manipulating *handles* such as points, cages, and bones [Jacobson et al. 2011]. Advanced techniques can help design a visually plausible pose by specifying the handle transformations from limited user inputs [Unzueta et al. 2008; Wu et al. 2004; Zhao and Badler 1994].

However, manually design animation with *dynamic effects* caused by inertia such as follow-through, slow-in, and slow-out [Thomas and Johnston 1981] requires considerable experience and talent. Many researchers have attempted to produce such dynamic effects automatically. One major approach is running a physics simulation on top of an artist-specified input animation. Fine details

can be added to a subspace without changing the coarse deformation. [Bergou et al. 2007; Iwamoto et al. 2015; Rémillard and Kry 2013; Zhang et al. 2020]. These methods employ physical simulations as post-processing; thus, the simulation does not assist in manipulating handles during the animation design process.

Our method enables seamless cooperation between user inputs and physics simulations, allowing them to both control the motions of the handles simultaneously. The key challenge is to achieve the coupling between dynamic simulation and skinning transformation. While the user manipulates some handles, other handles are automatically specified to give animation dynamic effects.

Our method was inspired by *complementary dynamics* [Zhang et al. 2020], a framework in which a physical simulation is constrained in a complementary subspace of a skinning deformation space. While their original study focused on adding detailed dynamic deformation to the skinning deformation, we utilize complementary dynamics to bidirectionally couple the skinning deformation and physics simulation – missing specifications of the skinning deformation handles are automatically computed through the physics simulation.

To achieve real-time performance, we formulated our method using position based dynamics (PBD) [Müller et al. 2007] by introducing a new constraint called *complementary constraint*. In each time step, we first solve the dynamics of the simulated material on a background triangle mesh. Then, we determine the parameters of handles by an optimization approach similar to the inverse kinematics. Finally, based on the parameters, we apply our complementary constraints to restrict the deformation of the background triangle mesh to the complementary space of the deformation space of the handles.

Our method allows the creation of dynamic animation with minimal user specification, even for a character with many handles. Once the character's shape and its handles are loaded, our algorithm does not require any precomputation, and the user can change the handles to manipulate, allowing intuitive control of skinning transformation on the fly. By bridging the gap between PBD and skinning transformation, we open up new possibilities for interactive dynamic character animation design.

2 RELATED WORKS

2.1 Static Deformation

Studies on the interactive deformation of 2D and 3D shapes are common in computer graphics research. We refer to the comprehensive survey in [Yuan et al. 2021]. Our goal is to achieve the two-way coupling between the skinning transformation and position based dynamics to support interactive dynamic character animation design.

Skinning. Skinning deforms a character's mesh by applying the spatially weighted transformations of the underlying handles, such as control points, cages, and rigid bones [Casti et al. 2019; Ju et al. 2008; Magnenat et al. 1988; Sederberg and Parry 1986]. The deformation with these handles can be achieved by linear blend skinning (LBS) using painted or automatically generated skinning weights [Jacobson et al. 2011]. Various studies have achieved higher deformation quality [Forstmann and Ohya 2006; Kavan et al. 2007; Le and Lewis 2019]. Our focus is still on the LBS because it is the most popular skinning deformation method owing to its simplicity and efficiency.

ARAP Deformation. The as-rigid-as-possible (ARAP) deformation [Igarashi et al. 2005; Sorkine and Alexa 2007] finds a quasi-static deformation that minimizes the quadratic elastic energy while satisfying constraints. Further extensions include handling volumetric models, enhanced smooth rotation, and simplified energy formulas [Chao et al. 2010; Cuno et al. 2007; Levi and Gotsman 2014]. Unlike skinning, the ARAP does not require skinning weights. However, precomputation is typically required for interactive frame rates. Our approach finds dynamic deformation in real time without precomputation.

2.2 Inverse Kinematics

With a model bound to a skeletal structure, inverse kinematics (IK) allows positional control at the end of the kinematic chain by solving the optimal parameters of the joints [Unzueta et al. 2008; Wu et al. 2004; Zhao and Badler 1994]. Shi et al. [2007] presented an IK that minimizes the distortion of the skinned mesh. The deformation of a mesh-based IK was further controlled by providing an example of deformation [Sumner et al. 2005] and the specification of rigidly moving vertices [Der et al. 2006]. While our method also optimizes the rigging parameters, our method takes the dynamic effect into account.

Unlike conventional IK which requires chains of articulated bones, Jacobson et al. presented fast automatic skinning transformations (FAST) [2012] allowing disconnected skeletons by solving the ARAP energy in the deformations subspace of LBS. The quadratic energy optimization is solved using a local-global approach: a local step for solving the rotation of triangular elements, and a global step for identifying skinning transformations. The algorithm is extremely fast with a precomputed singular value decomposition. This paper shares a similar goal; however, our work targeted real-time applications for dynamic effects without precomputation.

2.3 Dynamics as Post Processing

Time-integrated physics simulations can add dynamic effects caused by momentum and inertia to the input animation. For example, given a rigged character animation, the dynamics of the flesh and skin were simulated in [Bender et al. 2013; Capell et al. 2005; Li et al. 2013; McAdams et al. 2011]. Solving the deformation in the subspace defined by an animator's rig reduces the computational cost [Barbič et al. 2009; Hahn et al. 2012, 2013; James and Pai 2002]. Facial animation can also be locally simulated using additional rigs or blended shapes [Angelidis and Singh 2007; Rohmer et al. 2021]. Data-driven physics is also an option when characters share the same type of material and topology [Kim et al. 2017]. Willett et al. [2017] presented an interface for dynamic illustration using the rigs animated by a mass-spring system.

TRACKS [Bergou et al. 2007] computed natural secondary motions on the fine resolution mesh based on the input coarse resolution mesh leveraging Petrov–Galerkin method, resulting in orthogonality between the coarse deformation space and the fine deformation space. Continuous artist-directed dynamic illustration can be completed using example-based deformation [Bai et al. 2016].

While the seminal work TRACK focused on thin shell deformation, the complementary dynamics [Zhang et al. 2020] further extended the idea to the rigged mesh animation controlled by handles. The complementary dynamics constrained the detailed deformation simulation to the subspace orthogonal to the nonlinear deformation space of the rigged mesh. While the complementary dynamics required significant run-time computation as it solved the constrained optimization on the fine model, further acceleration was achieved by the subspace dynamics using eigenanalysis [Benchekroun et al. 2023]. On the other hand, we use position-based dynamics for acceleration and further estimate unspecified rigging parameters from dynamic deformation simulation.

2.4 Position Based Dynamics

PBD simulated the deformation for real-time applications by solving positional constraints on vertices sequentially [Macklin et al. 2014; Müller et al. 2007]. XPBD [Macklin et al. 2016] extended the PBD, such that the stiffness and damping can be specified independently from the iteration number. The substepping technique reduced the numerical damping [Macklin et al. 2019]. Stable and robust constraint-based formulations have been proposed for Neo-Hookean materials [Macklin and Muller 2021] and detailed rigid bodies [Müller et al. 2020]. By simulating the flesh of animated characters by PBD, secondary motion can be simulated at run-time [Iwamoto et al. 2015; Rumman and Fratarcangeli 2014].

By employing projective dynamics, Li et al. [2019] achieved two-way coupling between rigid and soft bodies in real time. While the rigid region and elastic region were completely separated in their approach, our approach shares these regions together through our novel PBD constraint formulation of complementary dynamics.

3 METHOD

User Interface. We first describe the application from the user's perspective. The user first imports a 2D character to be animated. Then, the user places deformation handles such as points or cages (see Figure 2). The user can select arbitrary handles to specify their translations or rotations, while the others remain unspecified (see Figure 3). The selection is on-the-fly – the user can freely select and deselect handles while the simulation is performed in real time.

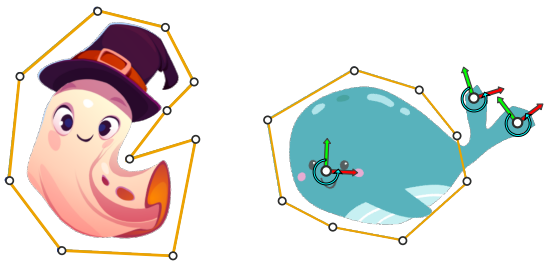

Figure 2: Cage-based handle is effective at manipulating the global shape, while point-based handle is good at modifying the local details. The left figure shows an example of cage-based deformation. In the right figure, cage-based and point-based controls are used together to combine their advantages.

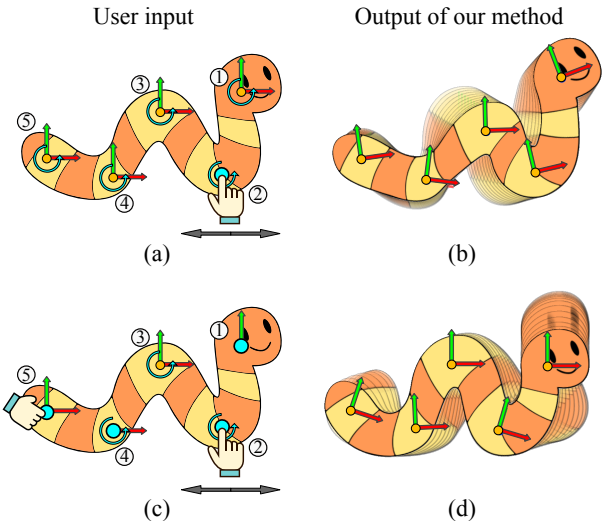

Figure 3: Examples of on-the-fly user control of the skinning transformation. (a) A worm controlled by five control points. Translation of point ② is specified by user inputs, but it is free to rotate. (b) By dragging the control point, the user can see the resulting wiggling animation. (c) Then, the user further specifies the rotation of control point ⑤. The vertical position of control point ④, the rotation and horizontal position of control point ① are fixed. (d) The resulting animation is immediately computed.

Algorithm Overview. Figure 4 illustrates the overview of our algorithm. The system first triangulates [Shewchuk 2002] the input shape and automatically computes the skinning weights for LBS-based deformation [Jacobson et al. 2011]. We use two meshes with the same topology in our method: a *rigged mesh* and a *simulation mesh*. The rigged mesh is shown on the screen, while the simulation mesh is hidden in the background. The deformation of the rigged mesh is fully controlled by skinning transformations (i.e., LBS). On the other hand, the deformation is simulated by PBD on the simulation mesh. Since the user's control is given to the rigged mesh but the dynamic deformation is simulated on the simulation mesh, we need to couple these two meshes.

Although the simulation mesh has more dynamic effects than the rigged mesh, we only display the rigged mesh for two reasons. First, the rigged mesh can work as a low-pass filter. For 2D shapes with detailed textures drawn by the artist, high-frequency deformation in the simulation mesh can only reduce the visual quality. In the rigged mesh, the quality of deformation is guaranteed by continuous weight distribution. Second, the animation of rigged mesh can be easily exported as sequential parameters of skinning transformations, which are convenient for further modification. Since our method is designed to be integrated into an animation production pipeline, it is convenient to input and output skinning transformations.

Based on the substepping technique [Macklin et al. 2019], we divide each time step into several substeps. Each substep consists of three procedures:

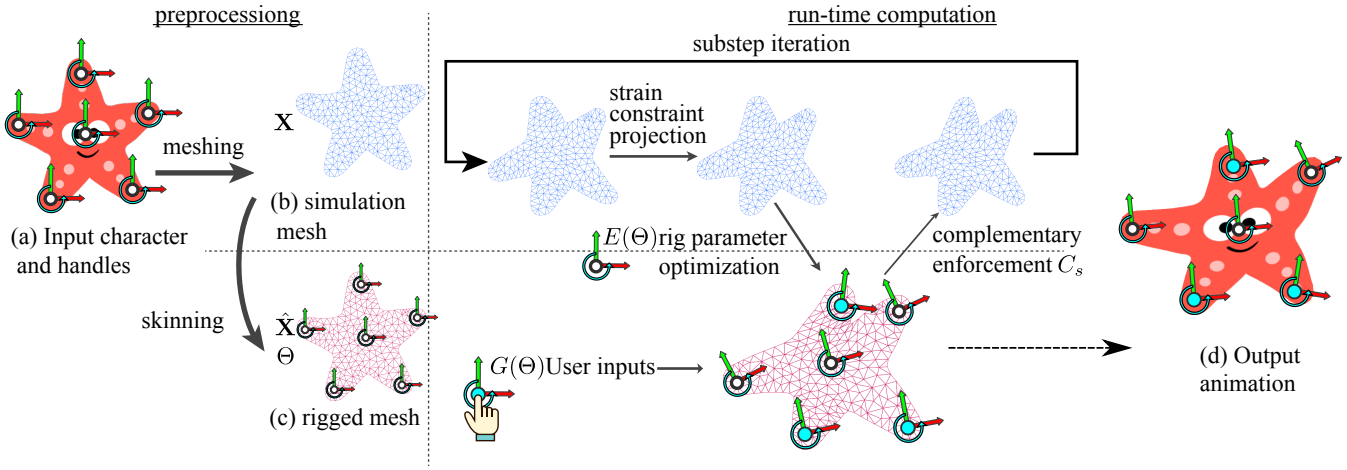

Figure 4: Overview of the system. (a) Input rigged character. (b) The simulation mesh hidden in the background. The deformation of Neo-Hookean materials is simulated on the mesh while the deformation is constrained from the rigging. (c) The rigged mesh shown on the screen as the final result. User inputs specify part of the rigging parameters. The unspecified parameters are computed based on the simulation mesh deformation. (d) The output animation is based on LBS, but with rich dynamic effects.

(1) **Simulation**: simulating the elastic deformation of the hidden mesh by constraint projections of the PBD.
(2) **Simulation to Rig**: fitting the rigged mesh to the simulation mesh by optimizing rigging parameters (Section 3.1).
(3) **Rig to Simulation**: applying complementary constraints to the simulation mesh, enforcing the orthogonality between the skinning deformation and PBD (Section 3.2).

Step (1) is the same as the standard simulation framework in XPBD [Macklin et al. 2016]. For brevity, we still refer to our simulation method as PBD. We use strain-based constraint for Neo-Hookean materials in [Macklin and Muller 2021] to reduce the mesh dependency. For the completeness of the paper, the algorithm of this part is briefly explained in Appendix A.

Note that in this simulation step, no boundary condition from the user inputs is enforced on the simulation mesh. The following two steps achieve the two-way coupling between the simulation mesh and the rig. Step (2) propagates the updates in the simulation mesh to the rig, and then step (3) updates the simulation mesh based on the rig.

Notation. In this paper, we only showcase experiments on 2D characters, but our algorithm generalizes to 3D space. Thus, we refer to the space dimension as $D \in \{2, 3\}$. We denote positions of the simulation mesh vertices as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N\times D}$, positions of the rigged mesh vertices as $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N]^T \in \mathbb{R}^{N\times D}$. The rigged mesh $\hat{\mathbf{X}}$ is rigged by the handles. We denote all the parameters of the handles in a vector $\Theta \in \mathbb{R}^S$, where S is the total degrees of freedom in the rigging parameter. As the rigged mesh is fully controlled by the handles, the rigged mesh can be written as a function of the rigging parameter $\hat{\mathbf{X}}(\Theta)$.

3.1 Simulation to Rig

After step (1), where the simulation mesh $\mathbf{X}$ is updated by PBD, we fit the rigged mesh $\hat{\mathbf{X}}$ to the simulation mesh by searching for the best rigging parameter. We measure the distance between the simulation mesh $\mathbf{X}$ and the rigged mesh $\hat{\mathbf{X}}$ with a quadratic formulation

$$E(\Theta) = \frac{1}{2}\left\langle \mathbf{X} - \hat{\mathbf{X}}(\Theta), \mathbf{M}\left\{\mathbf{X} - \hat{\mathbf{X}}(\Theta)\right\}\right\rangle_F, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the diagonal mass matrix, and $\langle *, * \rangle_F$ is the Frobenius inner product. The parameters of the handles Θ are partially given by user inputs, and the remaining parameters are unknown variables. The user inputs can be written as a constraint $G(\Theta) = 0$. Subject to this constraint, the goal of this section is to find the minimizer of (1).

Let Θ^t be the optimized rigging parameter to be found in the current substep. the constraint optimization can be solved by computing the extreme value of the Lagrangian

$$\mathcal{L}(\Theta^t, \lambda) = E(\Theta^t) + \lambda G(\Theta^t), \tag{2}$$

where the λ is the Lagrange multiplier.

We apply a single iteration of Newton's method to the Lagrangian (2) to estimate the optimal rig parameter Θ^t. The first-order Taylor expansion of the rigged mesh and the user input constraints are

$$\hat{\mathbf{X}}\left(\Theta^t\right) = \hat{\mathbf{X}}\left(\Theta^{t-1}\right) + \left.\frac{\partial \hat{\mathbf{X}}}{\partial \Theta}\right|_{\Theta^{t-1}} (\Theta^t - \Theta^{t-1}), \tag{3}$$

$$G(\Theta^{t-1}) + \left.\frac{\partial G}{\partial \Theta}\right|_{\Theta^{t-1}} (\Theta^t - \Theta^{t-1}) = 0, \tag{4}$$

where Θ^{t-1} is the rigging parameter from the previous substep. We describe the details of this derivative for the rigid transformation in Section 4. As the dimension of the rigging parameter Θ is typically small up to 100, the direct solver efficiently solves the linear system of Newton's iteration.

Following the strategy described in [Macklin et al. 2019], we employ the substepping technique. More substeps and fewer constraint-projection iterations can reduce the numerical damping at the same computational cost. Because the change of rigging parameter Θ is

typically small in each substep, we only solve one iteration of the linearized constraint optimization in (2).

3.2 Rig to Simulation

The previous step optimizes the rigged mesh $\hat{\mathbf{X}}$ by minimizing the quadratic difference from the simulation mesh $\mathbf{X}$. We then need to update the simulation mesh to follow the rigged mesh. However, naïvely optimizing simulation mesh by minimizing the quadratic difference in (1) simply matches the simulation mesh exactly to the rigged mesh, completely removing the detailed deformation that cannot be represented by the skinning. Although such detailed deformation does not directly appear in the rigged mesh, using the simulation mesh the same as the rigged mesh significantly dampens the simulation as the elastic potential energy decreases.

We need to allow the simulation mesh to move freely to some extent, meanwhile the simulation mesh needs to stay close to the rigged mesh, To achieve this goal, we use the approach inspired by the complementary dynamics [Zhang et al. 2020]. The complementary dynamics ensures the *orthogonality* between the complementary deformation and the LBS subspace. In our problem setting, the orthogonality can be formulated as

$$\left\langle \frac{\partial \hat{\mathbf{X}}(\Theta)}{\partial \Theta}\Big|_{\Theta^t}^{T}, \mathbf{M}(\mathbf{X}-\hat{\mathbf{X}}) \right\rangle_F = 0. \tag{5}$$

We optimize the simulation mesh $\mathbf{X}$ by enforcing the orthogonality constraint (5). This constraint is enforced using the constraint projection approach in the PBD framework for real-time performance. Note that the original work [Zhang et al. 2020] applies this orthogonality constraint to a modified Newton's method. Thus, dynamics simulation and constraints for all degrees of freedom need to be solved together, leading to computational costs too expensive for real-time applications.

We first reformulate the orthogonality constraint in (5) to a set of constraint functions for each rigging parameter θ_s

$$C_s(\mathbf{x}_1,\ldots,\mathbf{x}_N) = \sum_{i=1}^{N} m_i \left(\frac{\partial \hat{\mathbf{x}}_i}{\partial \theta_s}\right)^T (\mathbf{x}_i - \hat{\mathbf{x}}_i), \tag{6}$$

where $s \in \{1,\ldots,S\}$ and m_i is the mass of i-th vertex. The original PBD satisfies each constraint $\mathbf{C}_s = 0$ sequentially by changing simulation mesh $\mathbf{X}$. XPBD extends the method to handle variable compliance of constraints by setting up a quadratic energy [Macklin et al. 2016]. For the complementary constraint, the quadratic energy *complementary energy* becomes

$$E_{complementary} = \frac{1}{2h^2\tilde{\alpha}_c}\sum_{s=1}^{S}\mathbf{C}_s^2, \tag{7}$$

where h is the time interval for substepping, i.e., the time step divided by the number of the substeps. $\tilde{\alpha}_c \in \mathbb{R}$ is the positive compliance parameter.

The original method in complementary dynamics is equivalent to zero compliance $\tilde{\alpha}_c = 0$. Our PBD-based technique allows the user to adjust $\tilde{\alpha}_c$. Setting the complementary energy with lower compliance can let physics simulation more strictly adhere to user inputs, while setting the compliance higher can improve the smoothness, fluency, and exaggeration of output animation.

3.3 Algorithm Summery

Algorithm 1 shows the resulting complete workflow of our method. We take advantage of the property of the PBD algorithm, where constraints are separately satisfied in a Guass-Sidel manner. This property allows us to first solve elastic strain-based constraints and then the complementary constraints separately after the rigging parameters are determined.

Same as PBD, we add a damping coefficient k_{damp} in the final velocity update step. If there is no damping, the character vibrates continuously as the result of energy conservation, but the artists might prefer the damped animation. The compliance $\tilde{\alpha}_c$, damping k_{damping}, and material parameters can be adjusted in real time while observing the character's behavior to achieve the most satisfying results.

Algorithm 1: Our algorithm to couple PBD and rigged mesh. Λ stands for the Lagrange multipliers for the XPBD. Strain() and Complementary() stand for the constraint projection for strain-based PBD and our complementary constraint enforcement.

```
while animation do
    G ← UserInputs;
    for i = 1 to #substep do
        X_prev ← X
        V ← V + h/m f_ext
        X ← X + hV
        Λ = 0
        X, Λ ← Strain(X, Λ)
        Θ^t ← RigOptimization(X, G)                ▷ Section 3.1
        X̂ ← SkinningTransformation(Θ^t)
        X, Λ ← Complementary(X̂, Θ^t, Λ)            ▷ Section 3.2
        V ← k_damp (X − X_prev)
    end
    RenderMesh(X̂, Θ)
end
```

4 IMPLEMENTATION DETAIL

Linear Blend Skinning. The LBS blends the set of rigid transformations with the weights defined on the vertices. Let $\mathbf{R}_k \in \mathbb{R}^{D\times D}$ be the rotation matrix and $\mathbf{u}_k \in \mathbb{R}^D$ be the translation vector of the rigid transformation where $k \in \{1,\ldots,K\}$. The rigged mesh is controlled by LBS as

$$\hat{\mathbf{x}}_i = \sum_{k=1}^{K} w_{ik}\left[\mathbf{R}_k(\bar{\mathbf{x}}_i - \bar{\mathbf{u}}_k) + \mathbf{u}_k\right], \quad i = 1,\ldots,N, \tag{8}$$

where $\bar{\mathbf{x}}_i \in \mathbb{R}^D$ is a point in the initial mesh, $\bar{\mathbf{u}}_k$ is the translation in the initial configuration, and the skinning weights w_{ik} denotes the influence of the k-th transformation on the rigged mesh's position $\hat{\mathbf{x}}_i$.

Complementary Constraints. The complementary constraints in (6) requires the differentiation of LBS (8) deformation with respect to rigid transformations. For simplicity of notation, we consider the special case of three-dimensional deformation (i.e., $D = 3$). Note

that, the two-dimensional deformation is a special case where the rotation is limited to that of around the z-axis.

For the rigid transformation, the complementary constraints on the simulation mesh in (6) can be written as

$$\mathbf{C}_{u_k}(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} m_i w_{ik} (\mathbf{x}_i - \hat{\mathbf{x}}_i), \tag{9}$$

$$\mathbf{C}_{R_k}(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} m_i w_{ik} \left[\mathbf{R}_k(\bar{\mathbf{x}}_i - \bar{\mathbf{u}}_k)\right] \times (\mathbf{x}_i - \hat{\mathbf{x}}_i), \tag{10}$$

where $\mathbf{C}_{u_k} \in \mathbb{R}^3$ and $\mathbf{C}_{R_k} \in \mathbb{R}^3$ are the constraints corresponding to the k-th rigid transformation (i.e., $\mathbf{u}_k$ and $\mathbf{R}_k$) respectively. Note that these constraints have three-dimensional values. In the XPBD algorithm, we enforce the constraints sequentially for each dimension.

Rig Optimization Detail. As described in Section 3.1, we find the optimal rigging parameter Θ^t by minimizing the quadratic energy in (1) under the constraint. For each rigid transformation, we update the rotation matrix as $\mathbf{R}^t = \mathbf{R}(\Delta\mathbf{v})\mathbf{R}^{t-1}$ where $\mathbf{R}(\Delta\mathbf{v})$ is the rotation matrix computed from the axis-angle rotation vector $\Delta\mathbf{v} \in \mathbb{R}^3$. The linearization of this update becomes $\mathbf{R}^t \simeq (\mathbb{I} + [\Delta\mathbf{v}]_\times)\mathbf{R}^{t-1}$ where the $[*]_\times \in \mathbb{R}^{3\times3}$ stands for the anti-symmetric matrix equivalent to the cross product and $\mathbb{I}$ is the identity matrix in three-dimension. The linear system resulting from Newton's method becomes

$$\begin{bmatrix} \mathbf{M}_{uu} & \mathbf{M}_{Ru}^T & \mathbf{G}_{\mathbf{u}}^T \\ \mathbf{M}_{Ru} & \mathbf{M}_{RR} & \mathbf{G}_{\mathbf{R}}^T \\ \mathbf{G}_{\mathbf{u}} & \mathbf{G}_{\mathbf{R}} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{U} \\ \Delta\mathbf{V} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{C}_u \\ \mathbf{C}_R \\ 0 \end{bmatrix}, \tag{11}$$

where the right-hand side vectors corresponding to the k-th rigid transformations are the same as the $\mathbf{C}_{u_k}$ and $\mathbf{C}_{R_k}$ in (9), and the $\mathbb{R}^{3\times3}$ matrix entries corresponding to the k-th and l-th rigid transformations are

$$(\mathbf{M}_{uu})_{kl} = \sum_{i=1}^{N} m_i w_{ik} w_{il} \mathbb{I}, \tag{12}$$

$$(\mathbf{M}_{Ru})_{kl} = \sum_{i=1}^{N} m_i w_{ik} w_{il} [\mathbf{R}_k(\bar{\mathbf{x}}_i - \bar{\mathbf{u}}_k)]_\times, \tag{13}$$

$$(\mathbf{M}_{RR})_{kl} = \sum_{i=1}^{N} m_i w_{ik} w_{il} [\mathbf{R}_k(\bar{\mathbf{x}}_i - \bar{\mathbf{u}}_k)]_\times [\mathbf{R}_l(\bar{\mathbf{x}}_i - \bar{\mathbf{u}}_l)]_\times, \tag{14}$$

Same as FAST [Jacobson et al. 2012], we only include linear constraints on rig parameters, so rigid bone is out of discussion since the length constraint is nonlinear. We limit the scope to cases when only a chain of rigid bones is involved, and the root of the chain is fully controlled by user inputs. After the optimization of rigging parameters, we manually fix the length and orientation of each bone in the chain from root to tail. Fig. 5 shows an example of a swinging fish but controlling only the rotation of the root bone.

In the final step of mesh rendering, we can prepare a rigged mesh with high resolution purely for visualization (see Fig. 6). While a coarse mesh is enough to simulate the dynamics by our algorithm, unnatural discontinuous deformation will be shown on the screen due to the underlying coarse discretization. Since the input and output are both skinning transformations, we can rig a fine detailed mesh and show the final result on the mesh, resulting in more smooth deformation.

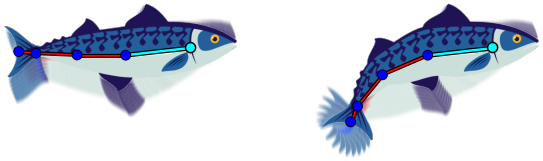

Figure 5: Animation of a swinging fish. The left figure is the user control on only the rotation of root bone. The right figure is the output animation with rich dynamics over the whole body.

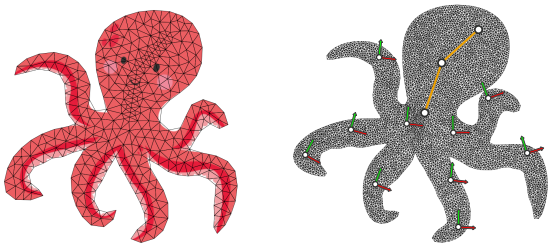

Figure 6: An example of additional fine skin to improve the visual quality of the output

5 RESULTS

Experimental Setup. We implemented our method in *Python* and *Taichi* [Hu et al. 2019]. The linear system in rig optimization was solved using the Python library *SciPy*. For performance measurements, We ran all our examples on a Core-i9-11900F CPU at 2.50 GHz. The performance was evaluated on a single thread, with the auto-parallelization of Taichi disabled. We bought the character images in this paper with the right where no attribution is required. Following the substepping strategy [Macklin et al. 2019], we set $\Delta t = 1/60\ s$ per frame, dividing each frame into 15 substeps. In each substep, we performed one iteration for the constraint projection for the elastic strain-based constraints and the complementary constraints.

Comparison. We conducted a comparison against ARAP [Sorkine and Alexa 2007] and FAST [Jacobson et al. 2012] using the same characters and user inputs (see Figure 1 and Figure 8). For the energy formulation in both methods, we selected the surface-based spokes and rims energy because deformation is more natural than other alternatives [Chao et al. 2010]. In each example, we manually fixed or dragged some of the handles, and observed the movements of the rest of the handles and the overall character animation. In ARAP, we fixed the movement of the vertices where the user specified the movement of the handles. Figure 8 illustrates how our method generates dynamic effects resulting from inertia in response to user inputs.

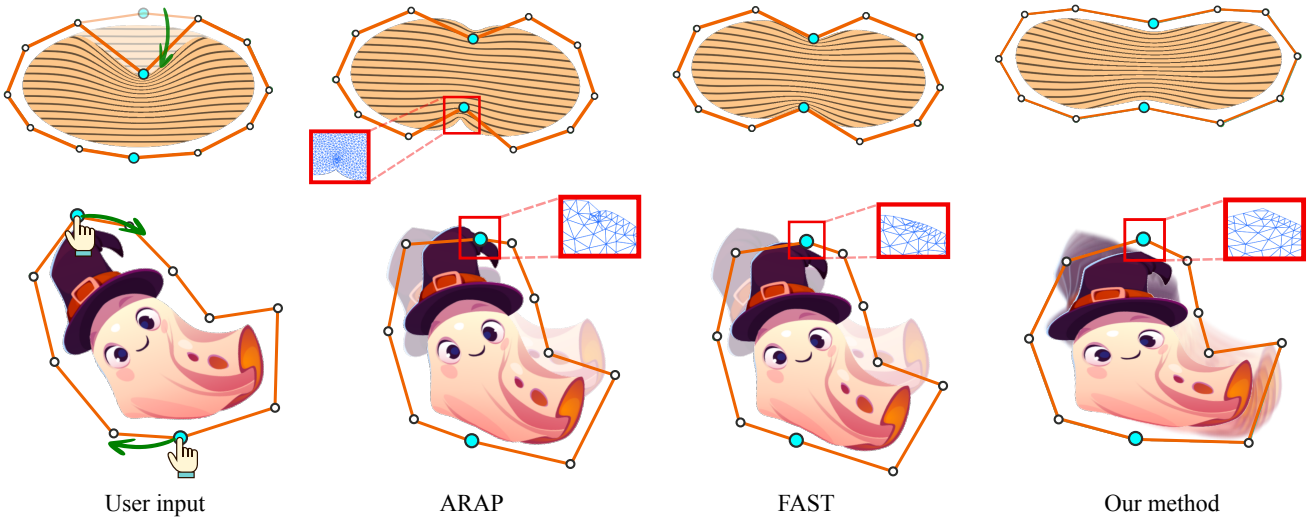

Figure 7: Manipulating cage-based characters with different methods. ARAP is prone to failure and causes mesh flipping in local regions near the handle. While FAST can generate smooth local deformation around the manipulated handles, the overall shape remains rigid, lacking dynamic effects. In contrast, our method allows the movements of specified handles to propagate throughout the entire spatial domain, resulting in lively and natural dynamic effects.

Figure 7 compares the deformation qualities of different methods using cage-based deformation. Our approach not only introduces dynamic motion but also produces globally smooth deformations. The amount of dynamic effect and smoothness are easily controllable through the interactive material parameters adjustment. The teaser in Figure 1 illustrates a more complex comparison with rotational constraints. Please see our supplemental video for the animations of these comparisons.

Effects of Parameters. Tweaking damping and compliance parameters for a specific character is crucial for high-quality output. Since no precomputation is required, the user can interactively adjust these parameters searching for the best output dynamic effects.

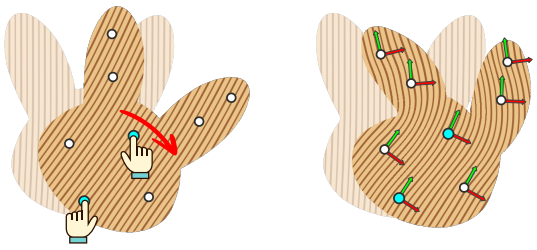

Figure 8: Rotating a bunny-head shape by dragging two control points. Both ARAP and FAST methods treat the shape as a rigid object during rotation. In contrast, our method introduces detailed motion to all free control points across the entire shape, resulting in more nuanced and dynamic deformations.

As described in Section 3.2, we can control the smoothness of the output animation by adjusting the compliance of the complementary constraints. This compliance parameter can serve as a low-pass filter in response to the rough user's input. Figure 9 compares the displacement of one vertex with different compliance values of complementary constraints. The larger compliance greatly smooths the motion curve filtering out rough high-frequency signals in the user inputs.

The material parameters of the Neo-Hookean material (i.e., hydrostatic compliance $\tilde{\alpha}_H$ and deviatoric constraints $\tilde{\alpha}_D$), are explained in Appendix A. As illustrated in Fig. 10, when compliance is extremely low, the model behaves similarly to a rigid body. On the

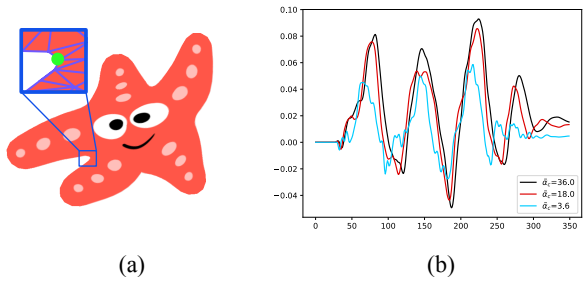

Figure 9: (a) We randomly selected a point and recorded its movement along the horizontal axis when the rigged mesh was manipulated by rough user inputs. (b) When the compliance coefficient $\tilde{\alpha}_c$ is low, indicating that the constraints are stiff, the movement of the chosen point moves rapidly and exhibits considerable oscillation. By setting $\tilde{\alpha}_c$ higher, the motion of the chosen point becomes smoother.

	Input model			Runtime(ms)		FPS
Model	N	Triangles	K	t_{pbd}	t_{rig}	
Ghost(cage-based, Fig. 7)	430	748	10	1.8	5.2	125
Starfish(Fig. 9)	273	445	6	1.9	5.9	113
Worm(Fig. 3)	839	1536	5	2.4	6.4	101
Ellipse(cage-based, Fig. 7)	1934	3727	12	3.2	6.6	91
Mushroom(Fig. 1)	1905	3653	5	3.2	8.1	82
Octopus(Fig. 11)	392	640	12	2.0	9.4	78
Fish(Fig. 5)	1040	1932	4	3.2	13.0	72
Bunny-head(Fig. 8)	2241	4269	8	3.1	14.4	55

Table 1: N represents the number of vertices, and K denotes the number of handles. We categorize the time cost into two parts; as outlined in Algorithm 1, t_{pbd} corresponds to the time cost of the steps that exist in the original PBD algorithm and t_{rig} represents the time cost of the new steps we have introduced, including rig parameter optimization, linear blend skinning, and solving complementary constraints. FPS indicates the frame rate of the application.

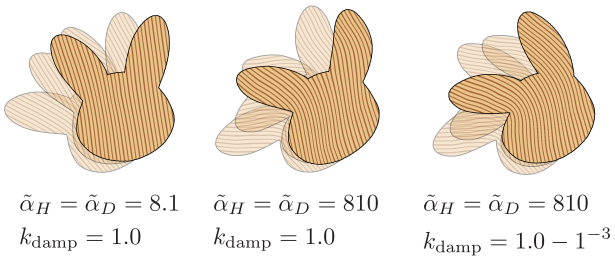

Figure 10: Same character model and user input as Fig. 8. The animation becomes different by changing elastic compliance and damping coefficients.

other hand, with high compliance, the model is simulated as a soft elastic body. While the animation becomes more dynamic and such vibrant movements are physically realistic, they may not be visually attractive for character animation. By adjusting the damping coefficient, the model has a follow-through effect in responding to user inputs. In our experiment, adjusting compliance and damping parameters is crucial for the balance between realistic dynamics and aesthetic appeal.

Physics-related Phenomena. As the motion is simulated using PBD in the background, our approach can directly handle physics-related phenomena such as external forces and collisions. In Figure 11, we added control points to each tentacle to deform the irregular shape. We can dynamically adjust the amount of gravity, causing the tentacles to rise and fall. By modeling collision as non-penetration constraints on the PBD, the tentacles can respond naturally to collisions. Please see the supplemental video for the animation result.

Performance. Table 1 presents the timing and frame rates for all examples included in this paper. We separate the computation time into two parts. The first part, t_{pbd}, represents the time cost when simulating the character as a free elastic material using PBD, while the other t_{rig} accounts for the time cost of the new steps we have introduced. The efficiency of our method is evident by comparing it against the time cost of the original PBD algorithm.

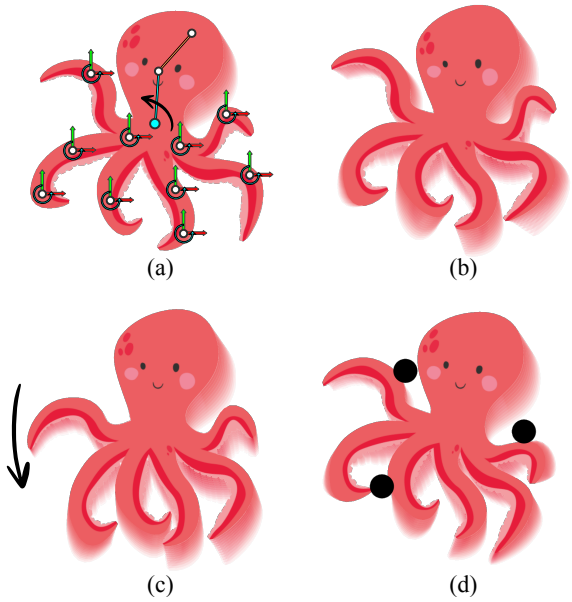

Figure 11: (a) An octopus with numerous control points on its tentacles, with the central root bone fully controlled by users. (b) The resulting animation. (c) Applying gravity as an external force. (d) Output animation with collision handling.

As anticipated, our approach offers intuitive character control and detailed physics-based simulation interactively.

6 LIMITATIONS AND FUTURE WORK

Although our algorithm has achieved real-time performance, there is still room for further optimization. For example, in the current implementation, we iterate through all vertices for each handle. However, leveraging the locality of the rigging weight, a handle can be related to only a small subset of vertices to reduce unnecessary computation.

Our rig parameter optimization uses a single iteration of Newton's method where the rotational transformations are linearized. With the substepping technique, we did not encounter any issues

resulting from the first-order approximation. However, the approximation might not quickly converge in some extreme situations where the current rigging parameters are very different from the optimal one.

In this paper, our focus is rigid transformations (i.e., each transformation has rotation and translation). One interesting extension is allowing richer transformations such as the affine transformation or the projective transformation. Since the affine transformation can represent anisotropic scaling, the resulting rigged mesh can squash and stretch, and thus it may greatly widen the expressive capability of the character animation.

Our algorithm is designed in 3D space, so we believe that our method can be easily applied to 3D characters given the rigged mesh and tetrahedral discretization. However, it is still left to be evaluated. In this paper, we run all the simulations on the CPU using a single thread. For computing large-scale 3D models on GPU, further modification may be necessary. Since the constraints are solved in a Gauss-Sidel manner, we need to split constraints into independent sets with techniques such as red-black ordering.

7 CONCLUSION

Linear blend skinning and PBD are both popular methods owing to their simplicity and efficiency. Through their two-way coupling, we have developed a user-friendly interface that streamlines the character animation design process. Leveraging the advantages of both methods, our proposed approach facilitates real-time manipulation of rigged characters with automatically complemented dynamic effects. The method is straightforward to implement, without the need for precomputation or complicated parameter settings. The interactive nature of our framework allows users to fine-tune material properties, integrate external forces, and manage collisions. While physics simulation operates in the background, our method inputs and outputs rigged meshes. This offers various benefits such as efficient storage, smooth deformations, and compatibility with a wide variety of animation software and game engines.

REFERENCES

Alexis Angelidis and Karan Singh. 2007. Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 129–140.

Yunfei Bai, Danny M Kaufman, C Karen Liu, and Jovan Popović. 2016. Artist-directed dynamics for 2D animation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. In *ACM SIGGRAPH 2009 papers*. 1–9.

Otman Benchekroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. 2023. Fast Complementary Dynamics via Skinning Eigenmodes. *arXiv preprint arXiv:2303.11886* (2023).

J Bender, J Dequidt, C Duriez, and G Zachmann. 2013. Physically-based character skinning. *Virtual Reality Interactions and Physical Simulations (VRIPhys) nov* (2013).

Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. Tracks: toward directable thin shells. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 50–es.

Steve Capell, Matthew Burkhart, Brian Curless, Tom Duchamp, and Zoran Popović. 2005. Physically based rigging for deformable characters. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 301–310.

Sara Casti, Marco Livesu, Nicolas Mellado, Nadine Abu Rumman, Riccardo Scateni, Loïc Barthe, and Enrico Puppo. 2019. Skeleton based cage generation guided by harmonic fields. *Computers & Graphics* 81 (2019), 140–151.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)* 29, 4 (2010), 1–6.

Alvaro Cuno, Claudio Esperança, Antonio Oliveira, and Paulo Roma Cavalcanti. 2007. 3D as-rigid-as-possible deformations using MLS. In *Proceedings of the 27th computer graphics international conference*. Citeseer, 115–122.

Kevin G Der, Robert W Sumner, and Jovan Popović. 2006. Inverse kinematics for reduced deformable models. *ACM Transactions on graphics (TOG)* 25, 3 (2006), 1174–1179.

Sven Forstmann and Jun Ohya. 2006. Fast Skeletal Animation by skinned Arc-Spline based Deformation.. In *Eurographics (Short Presentations)*. 1–4.

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space physics. *ACM transactions on graphics (TOG)* 31, 4 (2012), 1–8.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W Sumner, and Markus Gross. 2013. Efficient simulation of secondary motion in rig-space. In *Proceedings of the 12th ACM SIGGRAPH/eurographics symposium on computer animation*. 165–171.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141.

Naoya Iwamoto, Hubert PH Shum, Longzhi Yang, and Shigeo Morishima. 2015. Multi-layer Lattice Model for Real-Time Dynamic Character Deformation. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 99–109.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78.

Doug L James and Dinesh K Pai. 2002. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 582–585.

Tao Ju, Qian-Yi Zhou, Michiel Van De Panne, Daniel Cohen-Or, and Ulrich Neumann. 2008. Reusable skinning templates using cage-based deformations. *ACM Transactions on Graphics (TOG)* 27, 5 (2008), 1–10.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 39–46.

Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. 2017. Data-driven physics for human soft tissue animation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Binh Huy Le and JP Lewis. 2019. Direct delta mush skinning and variants. *ACM Trans. Graph.* 38, 4 (2019), 113–1.

Zohar Levi and Craig Gotsman. 2014. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics* 21, 2 (2014), 264–277.

Duo Li, Shinjiro Sueda, Debanga R Neog, and Dinesh K Pai. 2013. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Jing Li, Tiantian Liu, and Ladislav Kavan. 2019. Fast simulation of deformable characters with articulated skeletons in projective dynamics. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–10.

Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–7.

Miles Macklin and Matthias Muller. 2021. A constraint-based formulation of stable neo-hookean materials. In *Proceedings of the 14th ACM SIGGRAPH Conference on Motion, Interaction and Games*. 1–7.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*. 49–54.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Miles Macklin, Kier Storey, Michelle Lu, Pierre Terdiman, Nuttapong Chentanez, Stefan Jeschke, and Matthias Müller. 2019. Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–7.

Thalmann Magnenat, Richard Laperrière, and Daniel Thalmann. 1988. *Joint-dependent local deformations for hand animation and object grasping*. Technical Report. Canadian Inf. Process. Soc.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. In *ACM SIGGRAPH 2011 papers*. 1–8.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. In *ACM SIGGRAPH 2011 papers*. 1–12.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Matthias Müller, Miles Macklin, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Detailed rigid body simulation with extended position based dynamics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 101–112.

Olivier Rémillard and Paul G Kry. 2013. Embedded thin shells for wrinkle simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–8.
Damien Rohmer, Marco Tarini, Niranjan Kalyanasundaram, Faezeh Moshfeghifar, Marie-Paule Cani, and Victor Zordan. 2021. Velocity Skinning for Real-time Stylized Skeletal Animation. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 549–561.
Nadine Abu Rumman and Marco Fratarcangeli. 2014. Position based skinning of skeleton-driven deformable characters. In *Proceedings of the 30th Spring Conference on Computer Graphics*. 83–90.
Thomas W Sederberg and Scott R Parry. 1986. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 151–160.
Jonathan Richard Shewchuk. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry* 22, 1-3 (2002), 21–74.
Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. In *ACM SIGGRAPH 2007 papers*. 81–es.
Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.
Robert W Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. 2005. Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)* 24, 3 (2005), 488–495.
F Thomas and O Johnston. 1981. Disney animation: the illusion of life. Disney Editions. *Life, Hyperion, New York, NY, USA* (1981).
Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. 2008. Full-body performance animation with sequential inverse kinematics. *Graphical models* 70, 5 (2008), 87–104.
Nora S Willett, Wilmot Li, Jovan Popovic, Floraine Berthouzoz, and Adam Finkelstein. 2017. Secondary motion for performed 2d animation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 97–108.
Xiaomao Wu, Lizhuang Ma, Zhihua Chen, and Yan Gao. 2004. A 12-DOF analytic inverse kinematics solver for human motion control. *Journal of Information & Computational Science* 1, 1 (2004), 137–141.
Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. 2021. A revisit of shape editing techniques: From the geometric to the neural viewpoint. *Journal of Computer Science and Technology* 36, 3 (2021), 520–554.
Jiayi Eris Zhang, Seungbae Bang, David IW Levin, and Alec Jacobson. 2020. Complementary dynamics. *arXiv preprint arXiv:2009.02462* (2020).
Jianmin Zhao and Norman I Badler. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)* 13, 4 (1994), 313–336.

A DYNAMICS SIMULATION

Macklin and Müller [2021] showed that the Neo-Hooean material can be simulated via the XPBD framework. Here we recap the related details to make this paper self-contained. The implicit Euler method is widely used for dynamics simulation. In each time step, the converged solution is equal to that of a formulation known as variational implicit Euler method [Liu et al. 2013; Martin et al. 2011] that finds the positions of the vertices as the minimizer of

$$\mathbf{x}^{(t+1)} = \underset{\mathbf{x}}{\operatorname{argmax}} \quad \frac{1}{2}(\mathbf{x}-\mathbf{p})^T\mathbf{M}(\mathbf{x}-\mathbf{p}) + \mathbf{E}(\mathbf{x})h^2 \tag{15}$$

$$\mathbf{p} = \mathbf{x}^{(t)} + h\mathbf{v}^{(t)} + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}, \tag{16}$$

where h is the interval of the time step.

In XPBD [Macklin et al. 2016], the energy potential $\mathbf{E}(\mathbf{x})$ is represented by the set of constraints $\mathbf{C} = [C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_r(\mathbf{x})]^T$ as

$$\mathbf{E}(\mathbf{x}) = \frac{1}{2}\mathbf{C}(\mathbf{x})^T\mathbf{K}\mathbf{C}(\mathbf{x}), \tag{17}$$

where $\mathbf{K}$ is the diagonal stiffness matrix for each constraint. In actual implementation, a compliance coefficient is defined corresponding to inverse stiffness $\tilde{\alpha} = \frac{h^2}{k}$. For hard constraints, we can consider it as an energy potential with infinite stiffness, such that $\tilde{\alpha} = 0$. The Neo-Hookean model can be simulated by solving two constraints

$$\mathbf{C}_H(\mathbf{F}) = \det(\mathbf{F}) - 1 \tag{18}$$

$$\mathbf{C}_D(\mathbf{F}) = \sqrt{\operatorname{tr}(\mathbf{F}^T\mathbf{F})} \tag{19}$$

$\mathbf{F}$ is the deformation gradient tensor. The hydrostatic constraint $\mathbf{C}_H$ measures the compression of the volume, and the deviatoric constraint $\mathbf{C}_D$ measures the distortion. By substituting these two energies into the energy potential formula of XPBD, we can see its equivalency to the energy Neo-Hookean model

$$\Phi_{\text{Neo}} = \frac{\lambda}{2}(\det(\mathbf{F}) - 1)^2 + \frac{\mu}{2}(\operatorname{tr}(\mathbf{F}^T\mathbf{F}) - 3) \tag{20}$$

λ and μ are the Lamé parameters. The energy in Eq. 20 is an energy density. In each volume element we can assume that $\mathbf{F}$ is constant. The compliance coefficient of a volume element will be related to the Lamé parameters and its volume V_e

$$\tilde{\alpha}_H = \frac{1}{\lambda V_e h^2} \tag{21}$$

$$\tilde{\alpha}_D = \frac{1}{\mu V_e h^2} \tag{22}$$

Now the free mesh is ready to be simulated as an elastic body. Given a set of constraints C, The speciality of XPBD is to solve constraints separately and iteratively in a Gauss-Sidel manner. The part related to solving constraints in XPBD is shown in algorithm 2.

Algorithm 2: Solve constraints

for *constraints c* **do**
  $\Delta\lambda = \frac{-C(\mathbf{x}) - \tilde{\alpha}\lambda}{\sum_{i=1}^{n} w_i |\nabla_{\mathbf{x}_i} C(\mathbf{x})|^2 + \tilde{\alpha}}$;
  $\Delta\mathbf{x} = \mathbf{M}^{-1}\nabla C(\mathbf{x})^T \Delta\lambda$;
  $\lambda \leftarrow \lambda + \Delta\lambda$;
  $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$;
end

Apart from the Neo-Hookean constraints and all the other common constraints like outer collision, the simulation should also be restricted by skinning transformations. Otherwise, the user inputs cannot take effects on the simulation results. This part is related to complementary dynamics [Zhang et al. 2020]. For clarification, we mark these constraints as $\mathbf{C}_c(\phi)$, and all the other constraints as $\mathbf{C}_e$. Since XPBD is a Gauss-Sidel solver, in each time step we can first solve $\mathbf{C}_e$, and leave $\mathbf{C}_c(\phi)$ later. In this section we explains the terms related to elastic simulation in $\mathbf{C}_e$, but $\mathbf{C}_c(\phi)$ is specified by skinning transformations. Recall that in our method, some degrees of freedom of skinning transformations should be automatically identified by physics simulation. Thus we need an additional step to specify $\mathbf{C}_c(\phi)$ before solving it.