

# Two-Way Coupling of Skinning Transformations and Position Based Dynamics

YUHAN WU, The University of Tokyo, Japan  
 NOBUYUKI UMETANI, The University of Tokyo, Japan

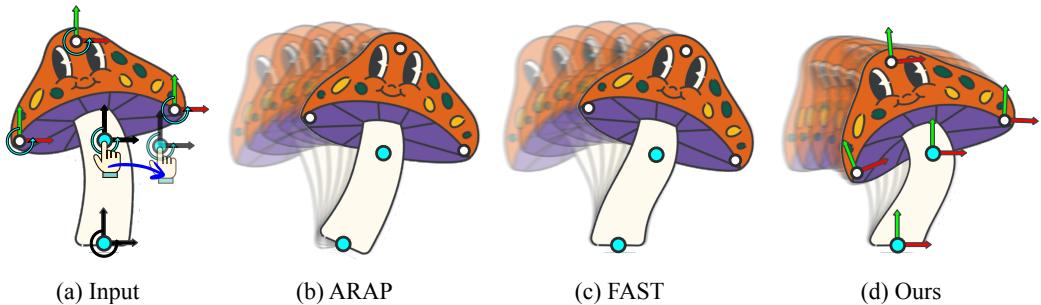


Fig. 1. (a) User inputs animation by dragging and fixing the control points. The degrees of freedom specified by the user inputs are shown in black. (b) As-rigid-as-possible (ARAP) deformation simply rotates the entire character with no deformation. (c) Automatic fast skinning transformation (FAST) can deform characters without dynamic effects. (d) Our method creates a 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 enable digital characters to be animated with minimal user input. Physics simulations can improve the detailed dynamic movement of an 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. This is achieved by introducing new steps into 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 to 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 tedious editing.

CCS Concepts: • Computing methodologies → Physical simulation.

Additional Key Words and Phrases: position based dynamics, complementary dynamics, skinning transformation

---

Authors' addresses: Yuhan Wu, The University of Tokyo, Japan; Nobuyuki Umetani, The University of Tokyo, Japan.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

2577-6193/2023/8-ART1 \$15.00

<https://doi.org/10.1145/3606930>

**ACM Reference Format:**

Yuhan Wu and Nobuyuki Umetani. 2023. Two-Way Coupling of Skinning Transformations and Position Based Dynamics. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 1 (August 2023), 18 pages. <https://doi.org/10.1145/3606930>

## 1 INTRODUCTION

Character animation is a labor-intensive art form; thus, computer graphics researchers have studied 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 handle transformations from limited user inputs [Unzueta et al. 2008; Wu et al. 2004; Zhao and Badler 1994].

However, manually designing animations 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 in addition to an artist-specified input animation. The 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 for postprocessing; 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 both to simultaneously control the motions of the handles. The key challenge is to achieve coupling between dynamic simulation and skinning transformation. While the user manipulates some handles, others are automatically specified to provide dynamic animation effects.

Our method was inspired by *complementary dynamics* [Zhang et al. 2020], which is a framework in which a physical simulation is constrained in a complementary subspace of a skinning deformation space. Whereas their original study focused on adding detailed dynamic deformation to the skinning deformation, we utilized complementary dynamics to bidirectionally couple the skinning deformation and physics simulation, and the missing specifications of the skinning deformation handles were automatically computed through 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 the *complementary constraint*. At each time step, we first solved the dynamics of the simulated material using a background triangular mesh. We then determined the parameters of the handles using an optimization approach similar to inverse kinematics. Finally, based on these parameters, we applied 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 specifications, even for characters with many handles. Once the character's shape and handles are loaded, our algorithm does not require expensive precomputation, and the user can change the handles to manipulate them, allowing intuitive control of the skinning transformation on the fly. By bridging the gap between PBD and skinning transformation, we open new possibilities for interactive dynamic character animation design.

## 2 RELATED WORKS

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 two-way coupling between skinning transformation and position-based dynamics to support interactive dynamic character animation design.

## 2.1 Geometric Deformation

*Skinning.* Skinning deforms a character’s mesh by applying 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]. Deformation of these handles can be achieved through linear blend skinning (LBS) using painted or automatically generated skinning weights [Jacobson et al. 2011]. Various studies have achieved a higher deformation quality [Forstmann and Ohya 2006; Kavan et al. 2007; Le and Lewis 2019]. Our focus is still on 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] determines a quasi-static deformation that minimizes the quadratic elastic energy, while satisfying constraints. Further extensions include the handling of 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, pre-computation is typically required for interactive frame rates. Our approach finds dynamic deformations in real time without pre-computation.

## 2.2 Inverse Kinematics

With a model bound to a skeletal structure, inverse kinematics (IK) enables 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 specifications of rigidly moving vertices [Der et al. 2006]. Although our method optimizes the rigging parameters, it does consider dynamic effects.

Unlike conventional IK, which requires chains of articulated bones, Jacobson et al. [2012] presented fast automatic skinning transformations (FAST), which allow disconnected skeletons by solving the ARAP energy in the deformation subspace of the LBS. Quadratic energy optimization was 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 precomputed singular value decomposition. Wang et al. [2015] presented an efficient approach for computing a smooth linear deformation space that can be controlled using control points and frames.

*Rig-space dynamics.* While IK determines the rigging parameters through static optimization, rig-space deformation determines the rigging parameters based on physics-based dynamics. Gilles et al. [2011] simulated dynamic deformation in the deformation space specified by dual quaternion skinning. Subsequently, the approach was generalized and accelerated [Hahn et al. 2012, 2013]. Wang et al. [2015] simulated the physics-based secondary dynamics in a highly controllable deformation space. Our approach is similar to these approaches in that we find the rigging parameters based on physics. However, although these approaches confine deformation to the rig space, our approach allows for simulated deformation outside the rig space using complementary dynamics.

## 2.3 Postprocessing Dynamics

Time-integrated physics simulations can add the 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; James and Pai 2002; Li et al. 2013; McAdams et al. 2011]. Barbić et al. [2009] added dynamics to keyframe animation using

fast space-time optimization. Velocity skinning [Rohmer et al. 2021] adds a cartoon-style dynamic stretching effect to the input quasistatic LBS animation. Data-driven physics is another option when the characters share the same material and topology [Kim et al. 2017]. Willett et al. [2017] presented an interface for dynamic illustrations using rigs animated by a mass-spring system.

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

The seminal work TRACK focused on thin-shell deformation, and complementary dynamics [Zhang et al. 2020] further extended the idea of rigged mesh animation controlled by handles. Complementary dynamics constrained the detailed deformation simulation to a subspace orthogonal to the nonlinear deformation space of a rigged mesh. Although the complementary dynamics required significant runtime computation as it solved the constrained optimization on the fine model, further acceleration was achieved by subspace dynamics using eigenanalysis [Bencheikroun et al. 2023]. However, we used position-based dynamics, both for real-time simulation of various materials and for maintaining orthogonality to the rigging deformation. Furthermore, the unspecified rigging parameters were optimized by fitting the rigged mesh to the simulated deformation.

## 2.4 Projective Dynamics

Projective dynamics [Bouaziz et al. 2014] is a simulation scheme that solves the implicit time integration of linear elastic material by a local-global solver. Liu et al. [Liu et al. 2017] presented a generalization of various nonlinear continuum material models. Brandt et al. [2018] accelerated the projective dynamics by limiting the deformation in the subspace spanned by linear blend skinning.

By employing projective dynamics, Li et al. [2019; 2020] achieved a two-way coupling between rigid and soft bodies in real time. Whereas the rigid and elastic regions were completely separated in their approach, our approach shares these regions through our novel PBD constraint formulation for complementary dynamics.

## 2.5 Position Based Dynamics

PBD simulates deformation for real-time applications by sequentially solving positional constraints on the vertices [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 of the iteration number. Based on XPBD, the sub-stepping technique can reduce numerical damping [Macklin et al. 2019]. A unified framework was proposed by Macklin et al. [2014] in which a rigid body was simulated as a group of connected particles by shape matching. Further work has extended the PBD to handle rigid bodies more efficiently as single entities [Müller et al. 2020; Rumman and Fratarcangeli 2014]. Stable and robust constraint-based formulations were proposed for Neo-Hookean materials [Macklin and Müller 2021]. By simulating the flesh of animated characters using PBD, secondary motion can be simulated at runtime [Iwamoto et al. 2015; Rumman and Fratarcangeli 2014].

## 3 METHOD

*User Interface.* First, we describe the application from the user’s perspective. Initially, the user imports characters to be animated. The user then places deformation handles such as points or cages (see Figure 2). The user can select arbitrary handles to specify their translations or rotations, whereas the others remain unspecified (see Figure 3). The selection was on-the-fly, and the user could freely select and deselect handles, while the simulation was performed in real time.

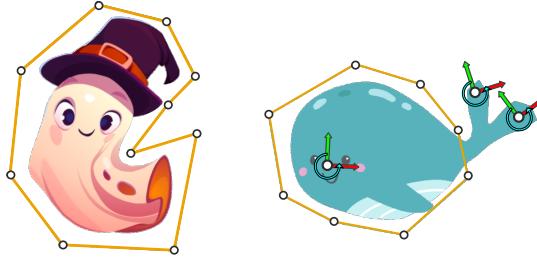


Fig. 2. Cage-based handles are effective in manipulating global shapes, whereas point-based handles are effective in modifying local details. The left figure shows an example of a cage-based deformation. In the figure on the right, cage- and point-based controls are used to combine their advantages.

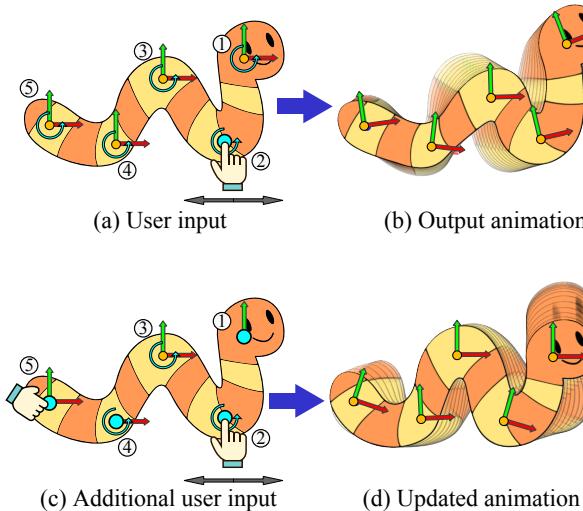


Fig. 3. Examples of on-the-fly user control of the skinning transformation. (a) Worm controlled using five control points. The translation of point ② is specified by user inputs, but it is free to rotate. (b) The user can view the resulting wiggling animation by dragging the control point (c) Then, the user specifies the rotation of the control point ⑤. The vertical position of the control point ④ and its rotation and horizontal positions of the control point ① are fixed. (d) The resulting animation is computed immediately.

*Algorithm Overview.* Figure 4 shows an overview of the proposed algorithm. The system first triangulates [Shewchuk 2002] the input shape and automatically computes the skinning weights for LBS-based deformations [Jacobson et al. 2011]. We used two meshes with the same topology in our method: the *rigged mesh* and *simulation mesh*. 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. Because the user’s control is given to the rigged mesh, but the dynamic deformation is simulated on the simulation mesh, these two meshes must be coupled.

The user can choose to display either the simulation or the rigged mesh. The simulation mesh had more high-frequency details (i.e., distortions) than the rigged mesh. Rigged animation can easily be exported as a sequence of skinning transformation parameters that are convenient for

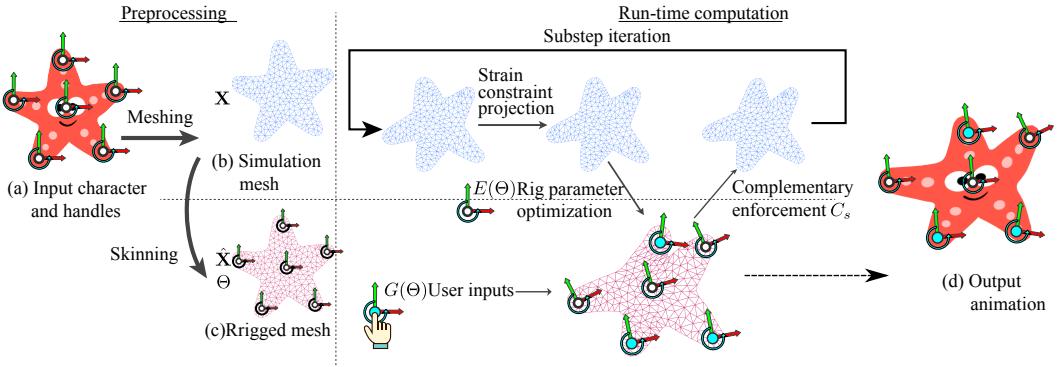


Fig. 4. Overview of the system. (a) Input rigged character. (b) Simulation mesh hidden in background. The deformation of the Neo-Hookean materials was simulated on the mesh, whereas the deformation was constrained by rigging. (c) The rigged mesh displayed on the screen is the final result. User inputs specify some of the rigging parameters. The unspecified parameters were computed based on the simulation mesh deformation. (d) Output animation is based on LBS, but with rich dynamic effects.

further downstream modifications. In our experiments, we show a rigged mesh for 2D characters and a simulation mesh for 3D examples.

Based on the sub-stepping technique [Macklin et al. 2019], we divided each time step into several sub-steps. Each sub-step consisted of three procedures.

- (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 the rigging parameters (Section 3.1).
- (3) Rig to Simulation: Apply complementary constraints to the simulation mesh, enforcing the orthogonality between the skinning deformation and PBD (Section 3.2).

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

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

*Notation.* Our algorithm was generalized to 2D and 3D spaces. Therefore, we refer to the spatial dimensions as  $D \in \{2, 3\}$ . 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. We denote positions of the simulation mesh vertices as  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$ , positions of the rigged mesh vertices as  $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_N]^T \in \mathbb{R}^{N \times D}$ . The rigged mesh  $\hat{\mathbf{X}}$  was rigged using handles. As the rigged mesh is fully controlled by the handles, it can be written as a function of the rigging parameter  $\hat{\mathbf{X}}(\Theta)$ .

### 3.1 Simulation to Rig

After Step (1), in which the simulation mesh  $\mathbf{X}$  is updated using the PBD, we fitted the rigged mesh  $\hat{\mathbf{X}}$  to the simulation mesh by searching for the best rigging parameter. We measured the distance between the simulation mesh  $\mathbf{X}$  and rigged mesh  $\hat{\mathbf{X}}$  using a quadratic formulation

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

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  $\Theta$  are partially provided by user inputs, and the remaining parameters are unknown variables. User inputs can be written as the constraint  $G(\Theta) = 0$ . Subject to this constraint, the goal of this section is to determine the minimizer of (1).

Let  $\Theta^t$  be the optimized rigging parameter determined in the current sub-step. The constraint optimization can be solved by computing the extreme values of the Lagrangian

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

where  $\lambda$  denotes the Lagrange multiplier.

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

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

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

where  $\Theta^{t-1}$  denotes the rigging parameter from the previous sub-step. In Section refsec: detail, we describe the details of this derivative for a rigid transformation in Section 4. Because the dimension of the rigging parameter  $\Theta$  is typically small (up to 100), the direct solver efficiently solves a linear system of Newton's iterations.

Following the strategy described in [Macklin et al. 2019], we employed a sub-stepping technique. More sub-steps and fewer constraint-projection iterations can reduce the numerical damping at the same computational cost. Because the change of rigging parameter  $\Theta$  is typically small in each sub-step, we only solve one iteration of the linearized constraint optimization in (2).

### 3.2 Rig to Simulation

The previous step optimized the rigged mesh  $\hat{\mathbf{X}}$  by minimizing the quadratic difference from the simulation mesh  $\mathbf{X}$ . Subsequently, we updated the simulation mesh to follow the rigged mesh. However, naively 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 a detailed deformation does not appear directly in the rigged mesh, using a simulation mesh similar to the rigged mesh significantly dampens the simulation as the elastic potential energy decreases.

We must allow the simulation mesh to move freely to some extent; meanwhile, the simulation mesh must remain close to the rigged mesh. To achieve this, we used an approach inspired by complementary dynamics [Zhang et al. 2020]. The complementary dynamics ensure *orthogonality* between the complementary deformation and the LBS subspace. In our problem setting, orthogonality can be formulated as

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

We optimized the simulation mesh  $\mathbf{X}$  by enforcing an orthogonality constraint (??). This constraint was enforced using a constraint-projection approach in the PBD framework for real-time performance. The original study [Zhang et al. 2020] applied this orthogonality constraint to a modified Newton’s method. Thus, dynamic simulations and constraints for all degrees of freedom must be solved simultaneously, leading to computational costs that are too high for real-time applications.

First, we reformulated the orthogonality constraint in (??) into a set of constraint functions for each rigging parameter,  $\theta_s$

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

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

$$E_{complementary} = \frac{1}{2h^2\tilde{\alpha}_c} \sum_{s=1}^S C_s^2, \quad (7)$$

where  $h$  is the time interval for sub-stepping, that is, the time step divided by the number of sub-steps.  $\tilde{\alpha}_c \in \mathbb{R}$  denotes a positive compliance parameter.

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

### 3.3 Algorithm Summary

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

Similar to the PBD, we added a damping coefficient  $k_{damp}$  in the final velocity-update step. If there is no damping, the character vibrates continuously because of energy conservation; however, artists might prefer damped animation. The compliance  $\tilde{\alpha}_c$ , damping  $k_{damping}$ , and material parameters can be adjusted in real time while observing the behavior of a character to achieve the most satisfactory results.

## 4 IMPLEMENTATION DETAIL

*Linear Blend Skinning.* LBS blends a set of rigid transformations with 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  $k \in \{1, \dots, K\}$ . The rigged mesh is controlled by the LBS as

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

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

---

**Algorithm 1:** Our algorithm to couple PBD and rigged mesh.  $\Lambda$  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
     $G \leftarrow \text{UserInputs};$ 
    for  $i = 1$  to #substep do
         $X_{\text{prev}} \leftarrow X;$ 
         $V \leftarrow V + h/m f_{\text{ext}};$ 
         $X \leftarrow X + hV;$ 
         $\Lambda = 0;$ 
         $X, \Lambda \leftarrow \text{Strain}(X, \Lambda);$ 
         $\Theta^t \leftarrow \text{RigOptimization}(X, G);$  // Section 3.1
         $\hat{X} \leftarrow \text{SkinningTransformation}(\Theta^t);$ 
         $X, \Lambda \leftarrow \text{Complementary}(\hat{X}, \Theta^t, \Lambda);$  // Section 3.2
         $V \leftarrow k_{\text{damp}}(X - X_{\text{prev}});$ 
    end
    RenderMesh( $\hat{X}, \Theta$ );
end

```

---

*Complementary Constraints.* The complementary constraints in (6) require differentiation of the LBS (8) deformation with respect to rigid transformations. For simplicity, we consider a special case of three-dimensional deformation (i.e.,  $D = 3$ ). Two-dimensional deformation is a special case in which rotation is limited to that around the z-axis.

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

$$C_{u_k}(x_1, \dots, x_N) = \sum_{i=1}^N m_i w_{ik} (x_i - \hat{x}_i), \quad (9)$$

$$C_{R_k}(x_1, \dots, x_N) = \sum_{i=1}^N m_i w_{ik} [R_k(\bar{x}_i - \bar{u}_k)] \times (x_i - \hat{x}_i), \quad (10)$$

where  $C_{u_k} \in \mathbb{R}^3$  and  $C_{R_k} \in \mathbb{R}^3$  are the constraints corresponding to the  $k$ th rigid transformation (i.e.,  $u_k$  and  $R_k$  respectively). These constraints have three-dimensional values. In the XPBD algorithm, we sequentially enforced constraints for each dimension.

*Rig Optimization Detail.* As described in Section 3.1, we determined the optimal rigging parameter  $\Theta^t$  by minimizing the quadratic energy in (1) under the constraint. For each rigid transformation, we updated the rotation matrix as  $R^t = R(\Delta v)R^{t-1}$  where  $R(\Delta v)$  is the rotation matrix computed from the axis-angle rotation vector  $\Delta v \in \mathbb{R}^3$ . The linearization of this update becomes  $R^t \approx (\mathbb{I} + [\Delta v]_x)R^{t-1}$  where the  $[*]_x \in \mathbb{R}^{3 \times 3}$  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

$$\begin{bmatrix} M_{uu} & M_{Ru}^T & G_u^T \\ M_{Ru} & M_{RR} & G_R^T \\ G_u & G_R & 0 \end{bmatrix} \begin{bmatrix} \Delta U \\ \Delta V \\ \lambda \end{bmatrix} = \begin{bmatrix} C_u \\ C_R \\ 0 \end{bmatrix}, \quad (11)$$

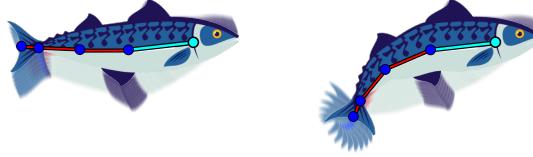


Fig. 5. Animation of the swinging fish. The left figure shows user control of the rotation of the root bone. The figure on the right shows the output animation with rich dynamics over the entire body.

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

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

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

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

Similar to FAST [Jacobson et al. 2012], we only include linear constraints on the rig parameters; therefore, the rigid bone is not discussed because the length constraint is nonlinear. We limited the scope to cases in which only a chain of rigid bones was involved and the root of the chain was fully controlled by user inputs. After optimizing the rigging parameters, we manually fixed the length and orientation of each bone in the chain from the root to the 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 a high resolution purely for visualization (see Fig. 6). While a coarse mesh is sufficient to simulate the dynamics using our algorithm, an unnatural discontinuous deformation is shown on the screen owing to the underlying coarse discretization. Because the input and output are both skinning transformations, we can obtain a fine detailed mesh and show the final result on the mesh, resulting in a smoother deformation.

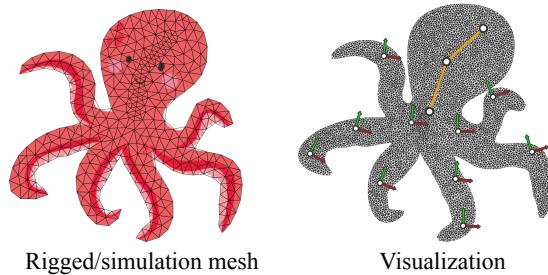


Fig. 6. Example of additional fine skin to improve the visual quality of the output.

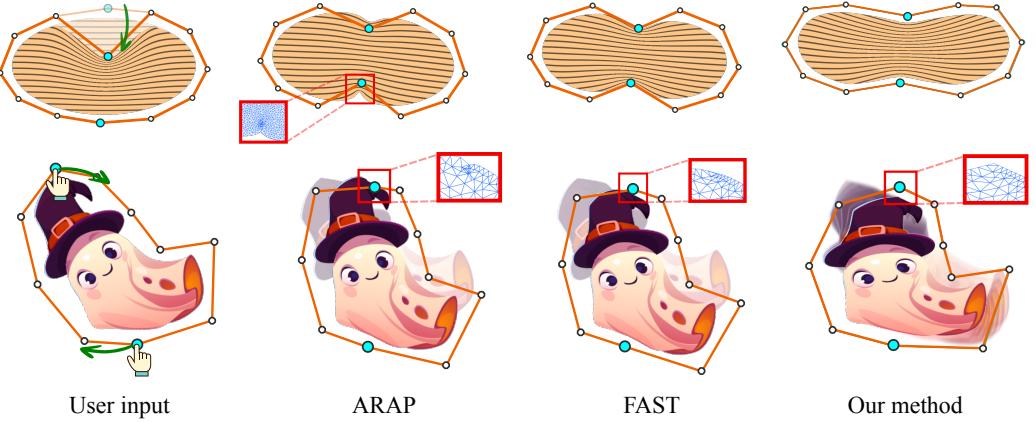


Fig. 7. Manipulation of cage-based characteristics using different methods. The ARAP is prone to failure and causes mesh flipping in local regions near the handle. Although FAST can generate smooth local deformations around manipulated handles, the overall shape remains rigid and lacks dynamic effects. By contrast, our method allows the movements of specified handles to propagate throughout the entire spatial domain, resulting in lively and natural dynamic effects.

## 5 RESULTS

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

*Comparison.* We conducted a comparison with ARAP [Sorkine and Alexa 2007] and FAST [Jacobson et al. 2012] using the same characters and user inputs(see Figure 1 and 8). For energy

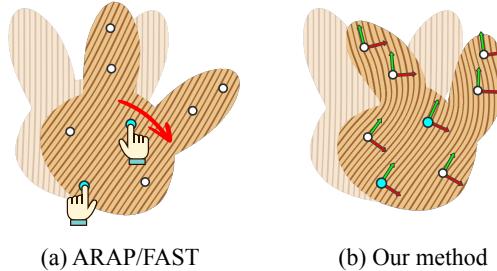


Fig. 8. Rotating the bunny-head shape by dragging two control points. Both the ARAP and FAST methods treat the shape as a rigid object during rotation. By contrast, our method introduces detailed motion to all free control points across the entire shape, resulting in more nuanced and dynamic deformations.

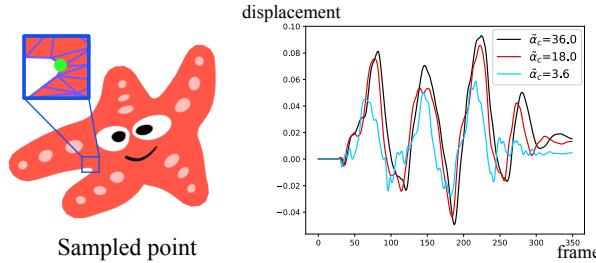


Fig. 9. We randomly selected a point (left) and recorded its movement along the horizontal axis (right) when the rigged mesh was manipulated using a rough user input. When the compliance coefficient  $\tilde{\alpha}_c$  is low, which indicates that the constraints are stiff, the selected point moves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

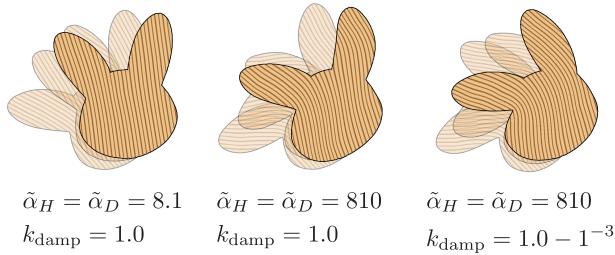


Fig. 10. Same character model and user input as Fig. 8. The animation becomes different by changing elastic compliance and damping coefficients.

formulation in both methods, we selected surface-based spokes and rim energy because deformation is more natural than in other alternatives [Chao et al. 2010]. In each example, we manually fixed or dragged some of the handles and observed the movements of the remaining handles and the overall character animation. In the ARAP, we fixed the movement of the vertices where the user specified the movement of the handles. Figure 8 illustrates how the proposed method generates dynamic effects resulting from inertia in response to user inputs.

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 dynamic effect and smoothness can be easily controlled by adjusting the interactive material parameters. The teaser in Figure 1 illustrates a more complex comparison with rotational constraints. Please see the supplemental video for animations of these comparisons.

*Effects of Parameters.* Tweaking the damping and compliance parameters for a specific character is crucial for a high-quality output. As no pre-computation is required, the user can interactively adjust these parameters to search for the best dynamic output effects.

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

The hydrostatic compliance  $\tilde{\alpha}_H$  and deviatoric constraints  $\tilde{\alpha}_D$  are the material parameters of the Neo-Hookean material. As illustrated in Fig. 10, when the compliance is extremely low, the model

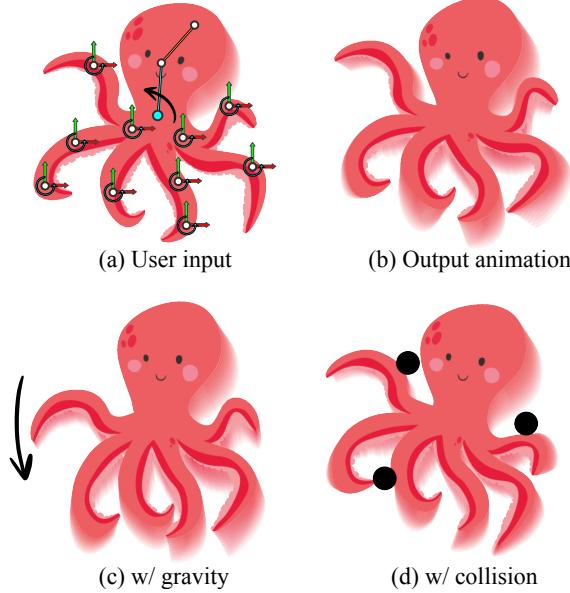


Fig. 11. (a) Octopus with numerous control points on its tentacles, with the central root bone fully controlled by the users. (b) Resulting animation. (c) Application of gravity as an external force. (d) Output animation with collision handling.

behaves similarly to a rigid body. However, with high compliance, the model was simulated as a soft elastic body. While 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 response to user inputs. In our experiment, adjusting the compliance and damping parameters was crucial for achieving a balance between realistic dynamics and aesthetic appeal.

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

*Comparison with Rig-space Dynamics.* Simulating physics-based dynamic deformation, where the deformation is limited to a rigged deformation sub-space, has more damping than our method. This is because the sub-space does not contain high-frequency deformations in the fine mesh, whereas our method allows high-frequency deformations in the simulation mesh, which is complementary to the rigged deformation subspace. Figure 12 compares the displacement of one vertex throughout the same dynamic simulation solved using the proposed method and the sub-space method in [Brandt et al. 2018]. Our method exhibits more detailed high-frequency dynamics.

*Performance.* Table 1 presents the timing and frame rates for all examples included in this study. The computational time was divided into two parts. The first part,  $t_{\text{pbd}}$ , represents the time cost when simulating a character as a free elastic material using PBD, whereas the other  $t_{\text{rig}}$  accounts for

Model	Input model			Runtime(ms)		FPS
	N	Triangles	K	$t_{\text{pbd}}$	$t_{\text{rig}}$	
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:  $t_{\text{pbd}}$  corresponds to the time cost of the steps that exist in the original PBD algorithm, and  $t_{\text{rig}}$  represents the time cost of the new steps we introduced, including rig parameter optimization, linear blend skinning, and solving complementary constraints. FPS indicates the frame rate of the application.

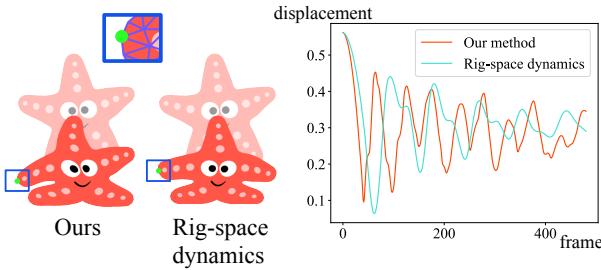


Fig. 12. We compared our method (left) with a rig-space dynamic simulation (middle) by sampling the movement of a vertex along the horizontal axis during the simulation of the starfish falling onto the ground without damping (right). The proposed method includes additional high-frequency dynamic details.

the time cost of the new steps we have introduced. The efficiency of our method is demonstrated by comparing it with the time cost of the original PBD algorithm. As expected, our approach offers intuitive character control and detailed interactive physics-based simulations.

*Material Models.* The proposed method supports the use of various materials. Fig. 13 illustrates the twisting deformation of a cube-shaped object with two types of materials: the Neo-Hookean material in [Macklin and Muller 2021] on a tetrahedral mesh and the combination of edge length, bending, and volume-preserving constraints in [Müller et al. 2007] on a triangle mesh. For each material, excessive stretching was observed near the control points when hard constraints were enforced directly on these points. In contrast, our method results in smooth deformations while satisfying the constraints. Note that Fig. 13 shows the simulation mesh used to visualize fine surface details.

*Further Acceleration by Precomputation.* The time cost of the PBD scales linearly with the model size. However, as indicated in Table 1, our method incurs an increasing time cost when handling a larger number of elements. This performance limitation is primarily attributed to the rig-optimization step discussed in Section 4, which involves updating and solving a linear system. To address this limitation, we draw inspiration from shape matching [Müller et al. 2005], where

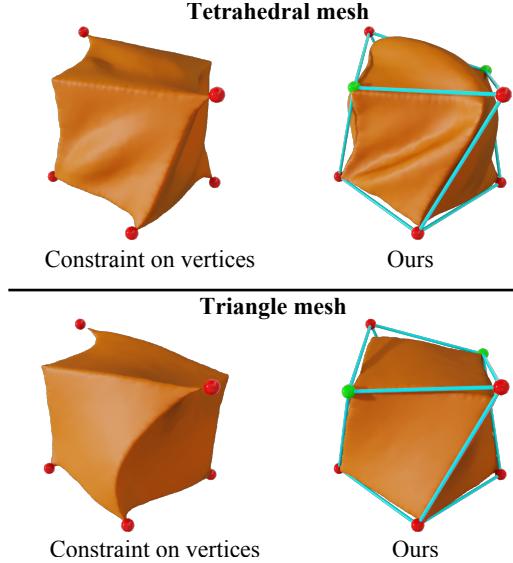


Fig. 13. Deformation of a 3D cube model with a Neo-Hookean material on a tetrahedral mesh (top) and a combination of edge length, bending, and volume-preserving constraints on a triangle mesh (bottom). Left: Applying hard constraints to the vertices. Right: Our method uses a cage-based deformation as the rigged mesh.

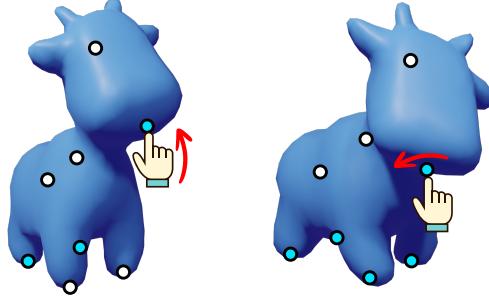


Fig. 14. A character was simulated on a mesh with 2k tetrahedra and eight control points. PBD simulation of this character as Neo-Hookean material costs 3.1 ms per frame. By precomputing the inverse matrix for rig optimization, our method requires only 5.4 ms in each frame, adding only a small cost to the original PBD simulation.

we first compute the optimal affine transformations, and then extract the translation and rotation components of the affine transformation as rigging parameters. As demonstrated in [Jacobson et al. 2012], affine transformations can be optimized by solving a constant linear system. To optimize the performance during user interaction, we can precompute the inverse of the coefficient matrix in (11) when the rigging structure is updated, and the inverse can be reused for subsequent updates. Fig. 14 showcases our experiment on a moderately-sized 3D model. Using this acceleration technique, the time cost of our method remains the same as that of position-based dynamics.

## 6 LIMITATIONS AND FUTURE WORK

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

Our rig parameter optimization uses a single iteration of Newton's method, in which the rotational transformations are linearized. Using the sub-stepping technique, we did not encounter any issues resulting from the first-order approximation. However, the approximation may not quickly converge in extreme situations where the current rigging parameters are very different from the optimal parameters.

In this study, we focused on rigid transformations (i.e., each transformation includes rotation and translation). One interesting extension is to allow richer transformations such as affine or projective transformations. Because affine transformation can represent anisotropic scaling, the resulting rigged mesh can squash and stretch; thus, it may greatly widen the expressive capability of character animation.

We ran all the simulations on the CPU using a single thread. Further modifications are required to compute large-scale 3D models on a GPU. Because the constraints are solved in a Gauss-Sidel manner, they must be split into independent sets using techniques such as red-black ordering.

## 7 CONCLUSION

Linear blend skinning and PBD are popular methods owing to their simplicity and efficiency. Using two-way coupling, we 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. This method is straightforward to implement without the need for expensive pre-computation or complicated parameter settings. The interactive nature of our framework allows users to fine-tune the material properties, integrate external forces, and manage collisions. Compared with rig-space dynamics, the simulation mesh in our method preserves high-frequency dynamics and fine deformation details. While the physics simulation operates in the background, the rigged mesh inputs, and outputs the skinning transformations. This offers various benefits such as efficient storage, smooth deformations, and compatibility with a wide variety of animation software and game engines.

## ACKNOWLEDGEMENT

We thank the anonymous reviewers for their insightful comments and suggestions. We also thank Yuki Koyama for the discussion in the early stage of this research project.

## REFERENCES

- 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 Barbic, 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.

- Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM transactions on graphics (TOG)* 33, 4 (2014), 1–11.
- Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.
- Steve Capell, Matthew Burkhardt, 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.
- Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K. Pai. 2011. Frame-Based Elastic Models. *ACM Trans. Graph.* 30, 2, Article 15 (apr 2011), 12 pages. <https://doi.org/10.1145/1944846.1944855>
- 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 Frédéric 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.
- Jing Li, Tiantian Liu, and Ladislav Kavan. 2020. Soft articulated characters in projective dynamics. *IEEE Transactions on Visualization and Computer Graphics* 28, 2 (2020), 1385–1396.
- Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. *AcM Transactions on Graphics (TOG)* 36, 3 (2017), 1–16.
- 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.
- 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, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)* 24, 3 (2005), 471–478.
- 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 Rémyard 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, Yiyi 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 Suárez. 2008. Full-body performance animation with sequential inverse kinematics. *Graphical models* 70, 5 (2008), 87–104.
- Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.
- 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.