

BlendSim: Simulation on Parametric Blendshapes using Spacetime Projective Dynamics

Yuhan Wu  and Nobuyuki Unemani 

The University of Tokyo, Japan

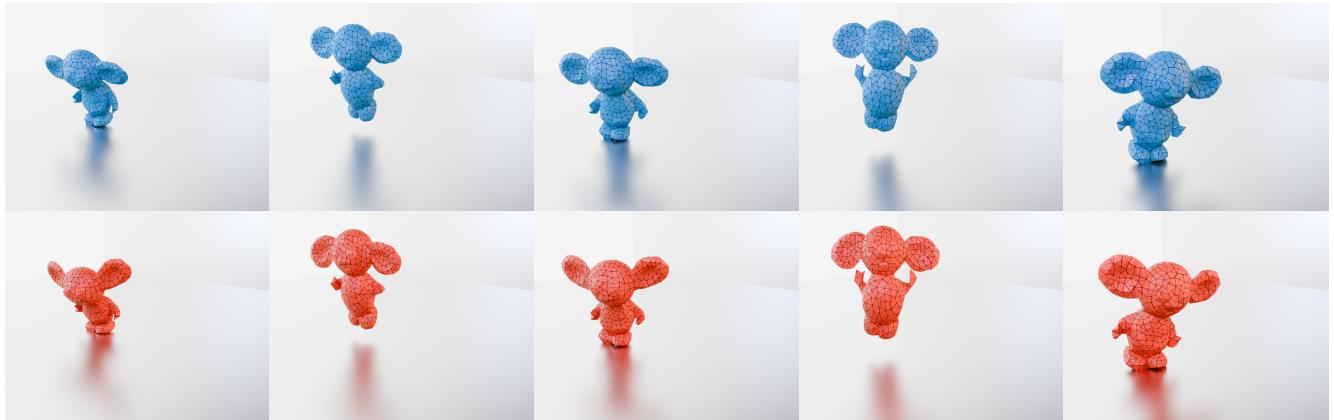


Figure 1: From the five input meshes at keyframes (top), our method outputs animation with dynamic inertia effect (bottom). Simply blending the input meshes at the keyframes using linear or spline interpolation leads to unnatural animation where the character looks stiff. Our method generates more natural blendshape animation by optimizing the meshes at the keyframes and the timings of the keyframes. Our spacetime optimization allows the user to selectively release the constraints to achieve the large secondary motion that deviate from the keyframes (see the ears of the Cheburashka model).

Abstract

We propose BlendSim, a novel framework for editable simulation using spacetime optimization on the lightweight animation representation. Traditional spacetime control methods suffer from a high computational complexity, which limits their use in interactive animation. The proposed approach effectively reduces the dimensionality of the problem by representing the motion trajectories of each vertex using continuous parametric Bézier splines with variable keyframe times. Because this mesh animation representation is continuous and fully differentiable, it can be optimized such that it follows the laws of physics under various constraints. The proposed method also integrates constraints, such as collisions and cyclic motion, making it suitable for real-world applications where seamless looping and physical interactions are required. Leveraging projective dynamics, we further enhance the computational efficiency by decoupling the optimization into local parallelizable and global quadratic steps, enabling a fast and stable simulation. In addition, BlendSim is compatible with modern animation workflows and file formats, such as the glTF, making it practical way for authoring and transferring mesh animation.

CCS Concepts

- Computing methodologies → Physical simulation;

1. Introduction

Keyframe animation provides a simple but effective method for designing the movement of a character by interpolating poses in

sparse discrete keyframes. In contrast, physically based animation typically assumes different representations, where the vertex positions of the meshes are stored for each frame, requiring a vast

amount of storage. Furthermore, once a frame-by-frame animation is computed, manual editing of the outcome is impractical, as modifying a single frame introduces abrupt changes in the animation. Since the simulation is computed and stored frame-by-frame, transferring data is also difficult due to large file sizes, thus integrating such simulations with existing workflows proves challenging.

Several studies on character animation have explored the creation of plausible motions by combining simulation and keyframe-based workflows. Spacetime constraints offer a solution by framing this challenge as an optimization problem, allowing users to define keyframes and physics to govern the in-between motions [WK88]. Given a set of keyframes and their corresponding timings, a physical model of the character can be established, and the trajectory that most closely adheres to physical laws can be determined by solving a minimization problem in the time domain. Subsequent studies have extended spacetime control to deformable objects [BdSP09; HSvTP12; SvTSH14], optimizing the motion of elastic materials between keyframes.

Despite their theoretical potential, spacetime control methods have witnessed limited practical adoption in animation design owing to their high computational demands and memory requirements. Existing spacetime control methods often require discretization of the time domain into small increments, which significantly increases the dimensionality of the problem and, thus, the computational cost. Although previous studies have attempted to make spacetime control more practical, these methods still struggle to provide both editable and real-time solutions, particularly for complex deformable objects. Several approaches require subspace reduction methods to improve performance; however, this process sacrifices the spatial and temporal details of motion that often resemble trivial interpolations between input frames. Additionally, existing techniques struggle with large-scale problems owing to the high dimensionality of the data, that is, hundreds of frames, each containing the entire position of every vertex. This results in either slow convergence or a tendency for the optimizer to become trapped in local minima, leaving much of the potential of spacetime control untapped.

To address these limitations, we propose BlendSim, a novel spacetime control method that significantly reduces the complexity of the optimization process. We tailor the physics simulation to adopt a deformation representation, where the smooth trajectory of each vertex is specified by C^1 continuous parametric splines, such as Bézier splines, in the time domain. By optimizing only the control points of these splines (i.e., the blendshape meshes and their timing), we substantially reduce the number of variables, making it feasible to address spacetime control problems in real time.

Since the proposed approach adopts a spacetime optimization framework, we can control animation by setting constraints. For instance, the proposed method can take shapes at sparse keyframes as inputs, then the method automatically generates natural continuous animation. The proposed method can easily generate cyclic animations, as recently demonstrated by Jia et al. [JWL23]. In the proposed framework, collisions can be also handled by treating them as constraints applied to keyframe shapes.

We achieve efficient spacetime optimization using projective dynamics [BML^{*}23]. In contrast to methods that rely on approxi-

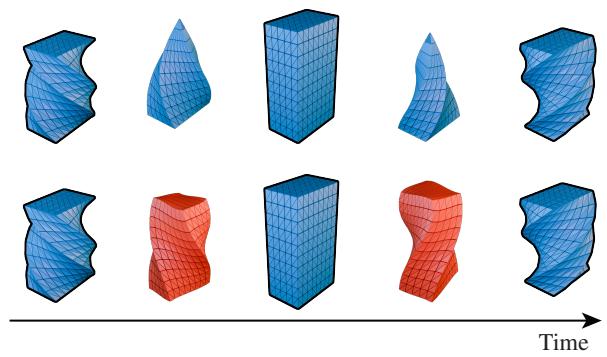


Figure 2: *Comparison against naïve keyframe interpolation.* (Top) the naïve linear interpolation between keyframe meshes (outlined) yields incorrect volume loss. (Bottom) the proposed method robustly computes physically natural dynamic deformation interpolating between these keyframes by adding four more keyframe meshes as the control points of the spline interpolation, which are optimized spatially and temporally.

mated linear elastic forces [BdSP09], projective dynamics decouples energy computations into a local nonlinear projection onto the constraint manifold, followed by a global linear projection across all constraints. The proposed method extends the global step to a quadratic minimization problem over a parametric trajectory. Meanwhile, the local step resolves the nonlinear deformation of the sample points. This is crucial for our method because a quadratic curve alone cannot capture highly nonlinear physical motion. The initial optimization requires several seconds, but the proposed method can reoptimize the animation in real time in response to the edits.

Our contributions are summarized as follows:

- **Spline-based spacetime optimization.** We introduce a novel spacetime control framework that reduces the complexity of animation optimization by representing motion trajectories using continuous parametric splines in the time domain. This approach allows for real-time optimization by significantly reducing the number of variables involved.
- **Keyframe timing optimization.** By treating timing as an optimizable parameter, we offer greater flexibility in generating more dynamic and physically plausible animations.
- **Handling various constraints.** The proposed method incorporates constraints for static collisions, which were often overlooked in traditional spacetime optimization owing to their complexity. Constraints on shapes and velocity tangents can be integrated seamlessly, such as sparse keyframe input and cyclic animation.
- **Compatibility with typical animation workflow.** The proposed method is highly compatible with modern keyframe-based animation workflows. Thus, we facilitate easy integration into current animation pipelines, making the proposed method practical for industry applications and content creation.

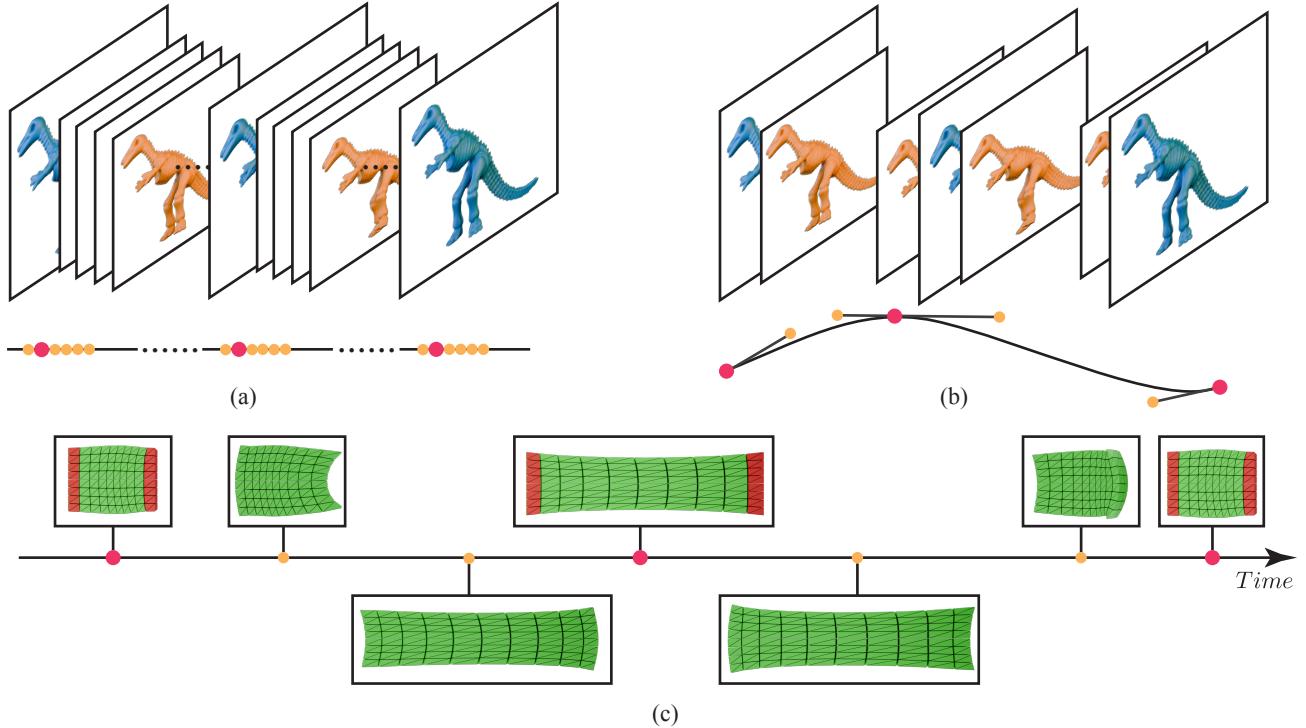


Figure 3: Editability of our blendshape representation against traditional frame-by-frame simulation results. (a) Typical physically-based simulation is computed frame-by-frame, which requires significant storage space. Manually editing one of the frames after simulation would violate animation continuity. (b) The proposed method uses much more compact spline interpolation based on blendshapes. Furthermore, blendshapes and their spline weights are editable in most animation software while maintaining weight continuity. (c) Example of edited blendshape (green) from the input constraints (red).

2. Related Works

2.1. Spacetime Constraints

The concept of spacetime constraints was introduced by Witkin et al. [WK88], where animation was formulated as an optimal control problem. Their groundbreaking work lays the foundation for future animation control and optimization studies. Similar to our method, Cohen [Coh92] developed an interactive spacetime editing system by b-spline based trajectory. It was further applied to various physical systems, mostly human motion optimization [RGBC96; Gle97; PW99; FP03; SHP04; SP05] and rigid bodies [PW99].

Subsequent studies extended spacetime optimization to deformable objects, albeit with varying approaches. For instance, Kass et al. [KA08] proposed Wiggly Splines, which applied spacetime constraints without directly focusing on physical simulations. They used a one-dimensional spring system controlled by complex number coefficients, which artists had to adjust manually to achieve the desired deformations. This method demonstrated the adaptation of spacetime constraints to non-physical character animations, although it relied heavily on user input.

The complexity of simulating elastic materials made spacetime constraints challenging to implement in deformable body simulations until Barbić et al. [BdSP09] introduced subspace simulations. By applying model reduction techniques to keyframe-based data,

they made spacetime optimization more feasible for interactive applications. Hildebrandt et al. [HSvTP12] built on this approach by combining linear model reduction around keyframes with Wiggly Splines, enabling interactive response times for deformable simulations. Li et al. [LHdG*14] expanded spacetime optimization to the inverse design of physical materials, addressing the problem by optimizing the material parameters alongside motion trajectories. The limited expressiveness of subspace models also restricts their range of achievable designs. We address the limitation of these methods in Fig. 4. Furthermore, they need to re-analyze the eigenmodes of keyframes if a keyframe shape is changed. Comparatively, as demonstrated in Fig. 5 our method can re-optimize the animation responsively with user edits.

Despite these advances, the local-minimum problem remains a significant challenge. Thomas and Joe [NM93] proposed a global search algorithm for generating multiple novel trajectories that can be executed in parallel. Schulz et al. [SvTSH14] addressed this issue by introducing sparse controls, which yielded more vibrant dynamics in deformable object animations. Furthermore, Wampler et al. [WP09] proposed a hybrid approach that combines spacetime optimization with a derivative-free finding algorithm that specifically targets locomotion in legged animals.

Cyclical animation is another important aspect of animation cre-

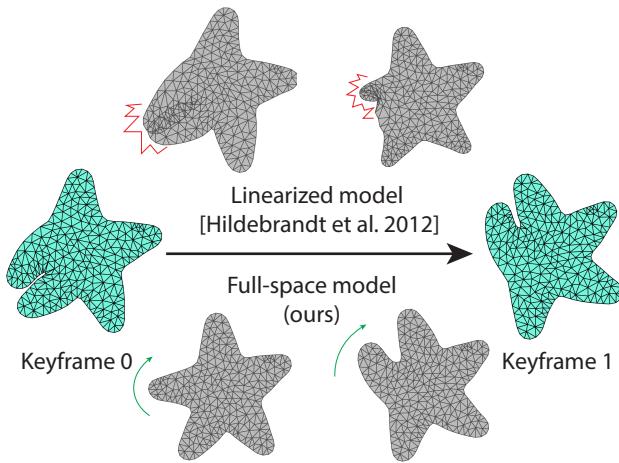


Figure 4: Starfish Waving Arm. Previous spacetime optimization methods often rely on linearized vibration analysis around each keyframe, which require keyframes to remain relatively close; otherwise, large differences can exceed the capacity of the linearized model which cannot generate a reasonable result [HSvTP12]. Although rotation-strain coordinates [LHdG*14] offer improvements, it is still reported to struggle with large rotations by later research [BZC*23]. In contrast, our method can handle significant rotational motions between keyframes.

ation. Gonzalez et al. [GAU10] applied periodic boundary conditions to partial differential equations to create cyclic motion. Similarly, Jia et al. [JWLC23] applied constraints only at the start and end of a motion clip to generate seamless looping trajectories, thereby rendering the cyclic animation highly efficient.

In addition, techniques such as static shape interpolation have been explored for smooth in-between keyframe animations [HTZ*10; ACZW19; VSSH15]. Whereas these methods achieve visually appealing interpolation, they lack dynamic effects and focus purely on geometric transformations without physical realism.

2.2. Complementary and Example-based Animation

Apart from spacetime optimization, a significant body of research focuses on automatically enriching input animation by adding more dynamic and detail. Barbić et al. [BdSP09] developed a keyframe editing system for directable deformable objects. For thin-shell animations, *Tracks* [BMWG07] extended the initial input by complementing it with detailed surface simulations. Zhang et al. [ZBLJ20] further extend this concept with complementary dynamics, a more general framework for adding secondary motion to existing animations. Several studies accelerates complementary dynamics by using position based dynamics [WU23] and using subspace simulation [BZC*23].

Projective dynamics, initially introduced by Bouaziz et al. [BML*23], has been proven effective in supporting example-based animations. It was further extended to handle skeletal char-

acters by Li et al. [LLK19a; LLK20], providing a framework for fast skeletal animation using projective dynamics. Differentiable projective dynamics [DWM*21] further enhance this technique by solving inverse problems in animations involving frictional contact, making it possible to compute physically accurate animations under various constraints. We extend the projective dynamics in a new direction. We present a differentiable formula for spacetime optimization. By representing motion trajectories continuously, we develop a novel local-global solver for optimizing the animations.

2.3. Blendshape Animation

Blendshapes are widely used in animation, particularly in facial animation. Lewis et al. [LHdG*14] provided a comprehensive survey on blendshape facial models, explaining how expressive facial animations are generated through the linear interpolation of blendshape basis shapes. More recently, Egger et al. [EST*20] presented a survey on the evolution of blendshape models in facial animation, particularly focusing on how motion capture data has been incorporated into 3D morphable face models. Deep learning approaches have also been explored. For example, Casas et al. [CO18] demonstrate how neural networks can learn the soft-tissue physics of blendshape models and reproduce dynamic blendshapes based on input skinned meshes, offering a new avenue for blending data-driven models with physics-based simulations.

The intersection of blendshapes and physics-based simulation is most commonly seen in facial animation, with various methods combining the two to improve realism and physical accuracy. Barrielle et al. [BSC16] introduced *Blendforces*, a framework that constrains physics-based simulations using motion-captured facial data. They convert facial animation into a set of force bases, allowing them to combine blendshapes through physics-based techniques. Ichim et al. [IKNP16] proposed a method that computes a set of physics-compatible blendshapes from 3D facial scans, combining the expressiveness of blendshapes with the accuracy of physics-based simulations. Similarly, Kadlecak et al. [KK19] developed a more accurate physics model based on facial mag-

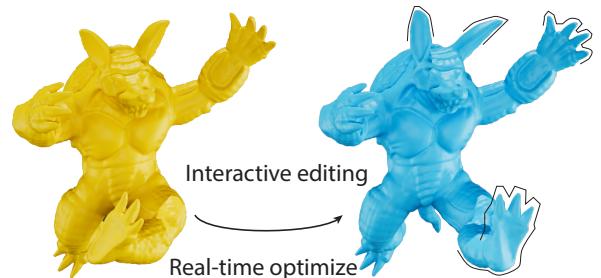


Figure 5: Full space interactive editing of ARMADILLO. Without spatial reduction, users can edit each keyframe shape in its full space. The interpolated animation is re-optimized responsively in real time, even for a character comprising up to 10k elements. Please refer to our accompanying video for a demonstration of this real-time interactive workflow.

netic resonance images, solving an inverse physics problem to build precise facial simulations. While these works focus on facial animations, they highlight the potential of combining blendshapes with physics-based techniques. Our work extends the combination of blendshapes and physics-based simulation to hyperelastic deformable objects.

3. Methods

Input and output Our inputs of our method include a shape represented as a mesh, a set of the mesh's keyframe deformations, and physics simulation parameters (e.g., stiffness, and gravity). Specifically, the user input N number of deformed shapes, which have vertex positions $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N\}$, at corresponding keyframe timing $\{\bar{t}_1, \dots, \bar{t}_N\}$. From the input key frames, our method optimizes the keyframe animation based on the physical plausibility. The output animation is N number of optimized keyframe timings $\{t_1, \dots, t_N\}$ and corresponding meshes with optimized vertex positions $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and the velocity of the vertices $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_N\}$.

The output keyframe deformation *partially* matches the input keyframe deformation. The user can input which vertices in the mesh in which keyframe output deformation follows the input deformation. Specifically, the user inputs a set of masks for each keyframe $\mathcal{M} = \{\mathbf{m}_1, \dots, \mathbf{m}_N\}$. The mask of a vertex takes 1 if the output vertex position matches with that of the input and takes 0 if the output vertex position moves freely during the optimization (i.e., $\mathbf{m}_i \odot \mathbf{x}_i = \mathbf{m}_i \odot \bar{\mathbf{x}}_i$ where \odot stands for the element-wise product). Additionally, the user can input the constraint on the keyframe timing, e.g., for which keyframe the output timing need to be equal to the input timing. Furthermore, the user can specify various constraints on the output animation, such as looping constraints and discrete velocity constraints, which we will discuss in detail in Section 3.5.

3.1. Spline Representation of Mesh Animation

We animate the mesh using the sequence of cubic Bézier splines. We interpolate vertex positions and their velocities at two consecutive keyframes. As illustrated in Figure 6, at the time $t \in [t_i, t_{i+1}]$, the positions of the mesh is computed

$$\begin{aligned} \mathbf{x}(t) = & b_0(\theta)\mathbf{x}_i + b_1(\theta)(\mathbf{x}_i + T_i\mathbf{v}_i) \\ & + b_2(\theta)(\mathbf{x}_{i+1} - T_i\mathbf{v}_{i+1}) + b_3(\theta)\mathbf{x}_{i+1}, \end{aligned} \quad (1)$$

where b_0, b_1, b_2, b_3 are the cubic Bernstein polynomials, $\theta = (t - t_i)/(t_{i+1} - t_i)$ is their parameter, and $T_i = t_{i+1} - t_i$ is the interval between the keyframes. Note that this construction ensures that the velocity of vertices exactly becomes \mathbf{v}_i at the keyframe time t_i . Thus, our formulation of the sequence of Bézier splines has C^1 -continuity ensuring smooth transitions across keyframes. Detailed derivations are provided in Appendix A. Our formulation in (1) can be considered blendshapes, and the output can be directly exported to an animation format supporting blendshape.

We express the relationship between position and control parameters in a more concise matrix representation

$$\mathbf{x}(t) = \mathbf{B}_i(\theta)\mathbf{q}_i, \quad (2)$$

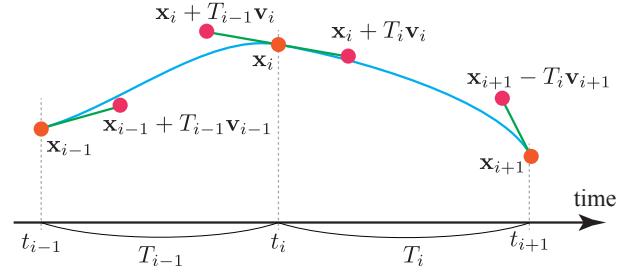


Figure 6: Parameterization of mesh animation using the vertex positions \mathbf{x} and their velocity \mathbf{v} at the keyframe timings. We use cubic Bézier spline interpolation to ensure the C^1 continuity.

where $\mathbf{B}_i = [b_0 + b_1, b_1 T_i, b_2 + b_3, -b_2 T_i]$ is the blendshape weights and $\mathbf{q}_i = [\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_{i+1}, \mathbf{v}_{i+1}]^\top$ are the positions and velocities at both ends of the i -th interval. The first and second time differentiation of vertex positions $\mathbf{x}(t)$ becomes $\dot{\mathbf{x}} = (1/T_i)\mathbf{B}'_i\mathbf{q}_i$ and $\ddot{\mathbf{x}} = (1/T_i^2)\mathbf{B}''_i\mathbf{q}_i$ respectively, where \mathbf{B}'_i and \mathbf{B}''_i is the first and the second derivative of \mathbf{B}_i with respect to θ .

Compatibility with existing animation tools Our representation of the mesh animation can be considered a blendshape animation (see Fig. 3). As a result, any file formats or software that support blendshapes are compatible with our method, and our method can be easily embedded into other animation design frameworks. One of the advantages of blendshape animation is edit-ability. Even if our optimization is not running in real-time, the artists can still manually edit the blendshapes and their spline weights without destroying the continuity of the animation. Another advantage is the compact file size. Because we only keep the vertex deformation information at sparse keyframes (rather than at all the frames with small time steps), the file size is compact, thus suitable for transferring the animation on the Internet.

3.2. Spacetime Optimization

Let us consider the equation of motion of elastic deformation

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{x}} = \mathbf{f}^{\text{in}}(\mathbf{x}) + \mathbf{f}^{\text{ext}}, \quad (3)$$

where the \mathbf{M} is the mass matrix, \mathbf{D} is the damping matrix, \mathbf{f}^{in} is the internal force (i.e., gradient of the elastic potential energy multiplied by -1), and \mathbf{f}^{ext} is the constant external force (e.g., gravity).

The spacetime optimization (e.g., [BSG12]) often minimizes the $L2$ norm of the residual force integrated in the time domain

$$W = \sum_{i=1}^{N-1} \int_{t_i}^{t_{i+1}} \frac{1}{2} \|\mathbf{f}^{\text{res}}(t)\|_{\mathbf{M}^{-1}}^2 dt, \quad (4)$$

where the $\mathbf{f}^{\text{res}} = \mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}\dot{\mathbf{x}} - \mathbf{f}^{\text{in}}(\mathbf{x}) - \mathbf{f}^{\text{ext}}$ is the residual force, which is the unbalanced force and thus a measure for physical implausibility.

Unfortunately, the time integration in (4) cannot be *analytically* evaluated. Hence, we *numerically* compute it by sampling K number of θ uniformly inside $[0, 1]$ for all the intervals between the

keyframes

$$W \simeq \sum_{i=1}^{N-1} \frac{T_i}{2K} \sum_{k=1}^K \|\mathbf{f}_{ik}^{\text{res}}\|_{\mathbf{M}^{-1}}^2, \quad (5)$$

where $\mathbf{f}_{ik}^{\text{res}}$ is the residual force in the i -th interval's k -th sampling point $\mathbf{f}_{ik}^{\text{res}} = \mathbf{f}^{\text{res}}(t_i + T_i \theta_k)$.

The Jacobian of residual force with respect to \mathbf{q}_j is given by

$$\frac{\partial \mathbf{f}_{ik}^{\text{res}}}{\partial \mathbf{q}_i} = \frac{1}{T_i^2} \mathbf{MB}_{ik}'' + \frac{1}{T_i} \mathbf{DB}'_{ik} - \frac{\partial \mathbf{f}^{\text{in}}}{\partial \mathbf{x}} \mathbf{B}_{ik}. \quad (6)$$

Using the gradient of the $\mathbf{f}_{ik}^{\text{res}}$, we can optimize the vertex positions at all the keyframes \mathcal{X} and their velocities \mathcal{V} . For example, since (5) is quadratic, we would use the Levenberg–Marquardt method that iteratively minimizes the objective function. However, this is very costly due to the nonlinear nature of internal force \mathbf{f}^{in} with respect to the vertex positions \mathbf{x} . The $\partial \mathbf{f}^{\text{in}} / \partial \mathbf{x}$ in (6) changes its value every iteration, so as the coefficient matrix.

3.3. Spacetime Projective Dynamics

We efficiently minimize the time-integrated norm of residual forces in (5) using the projective dynamics [BML*23] formulation, which introduces the rotation for each mesh element as auxiliary variables and interleaves the local and global optimization steps. The global step solves the deformation through quadratic optimization with a constant coefficient matrix, whereas the local step efficiently optimizes the rotations in a fully parallel way. While the original projective dynamics solves the deformation of a *single time step* on-by-one, our spacetime projective dynamics simultaneously optimize the *entire animation*. Our spacetime projective dynamics still benefit from the quadratic optimization with a constant coefficient matrix in the global steps and the parallel nature of the local steps.

As shown in [LLK19b], the internal forces in the projective dynamics can be written using the two constant matrices \mathbf{L} and \mathbf{J} as

$$\mathbf{f}^{\text{in}}(\mathbf{x}) = -\mathbf{Lx}(t) + \mathbf{Jp}(t), \quad (7)$$

where $\mathbf{p}(t)$ is a set of projected matrices for all the elements in the mesh obtained by projecting deformation gradients onto a constrained manifold in the local step. Using the formulation in (7), we can write the force residual at the k -th sampling point in the i -th interval as

$$\mathbf{f}_{ik}^{\text{res}} = \mathbf{G}_{ik}\mathbf{q}_i - \mathbf{Jp}_{ik} - \mathbf{f}^{\text{ext}}, \quad (8)$$

$$\mathbf{G}_{ik} = \frac{1}{T_i^2} \mathbf{MB}_{ik}'' + \frac{1}{T_i} \mathbf{KB}'_{ik} + \mathbf{LB}_{ik}, \quad (9)$$

where \mathbf{p}_{ik} is the set of rotations at the k -th sampling point in the i -th interval. Note that the Jacobian of residual force \mathbf{G}_{ik} now becomes constant, which is in constant to the non-constant Jacobian in (6). Our formulation enjoys significant acceleration by storing the Jacobian \mathbf{G}_{ik} for reuse.

The spacetime optimization now becomes minimization of

$$W \simeq \sum_{i=1}^{N-1} \frac{T_i}{2K} \sum_{k=1}^K \left\| \mathbf{G}_{ik}\mathbf{q}_i - \mathbf{Jp}_{ik} - \mathbf{f}^{\text{ext}} \right\|_{\mathbf{M}^{-1}}^2. \quad (10)$$

In the global step, we optimizes the positions and velocities of all



Figure 7: Gravity and collision handling. In this SPOT experiment, Starting from blue input keyframes and an initial guess of collision position, our method iteratively optimizes the in-between animation. The falling trajectory is parabolic due to the effect of gravity. Our optimization refines the time and deformation when the object collides with the floor.

the keyframes $\{\mathcal{Q}, \mathcal{V}\}$ while fixing the rotations at all the sampling time \mathbf{p}_{ik} . This results in solving quadratic optimization where its coefficient matrix is constant – we achieve significant acceleration by computing its Cholesky decomposition and store the decomposition for reuse. In the local step, we optimizes the rotations at all the sampling time \mathbf{p}_{ik} independently while fixing the positions and velocities of all the vertex at all the keyframes. This is also computed efficiently by taking advantage of the parallel computation – all the rotations \mathbf{p}_{ik} in each element in each time step can be computed in parallel.

Optimization detail We use the input vertex positions of the keyframes as initialization of the optimization $\mathcal{X} = \hat{\mathcal{X}}$, whereas the velocity is initialized as zero. The user's constraint on the vertex position at the keyframes is satisfied by setting the hard constraint in the optimization, i.e., we separate free and fixed vertex positions and only solve the optimization only for the positions of free vertices. In the global step, we regulate the update (i.e., damp the update to prevent jumping far from the input in one step) by adding scaled identity matrix to the coefficient matrix of quadratic system solver. This regularization ensures the stability of the Cholesky decomposition and steady convergence of the local-global solver. Our experiments suggest the regularization actually accelerate the convergence of the spacetime optimization (see Section 4). Note that in the original projective dynamics, the mass matrix plays the role of such regularization in the quadratic solver.

3.4. Keyframe Timing Optimization

In contrast to the frame-by-frame simulation, our spline-based interpolation between keyframes allows the continuous optimization of the keyframe timing because the keyframe is not a discrete step but rather a continuous variable. In our Bézier spline formulation, changing interval T_i does not affect the sampled residual forces in intervals other than T_i . Taking advantage of this locality, we can optimize T_i just by minimizing the time integral of the residual force

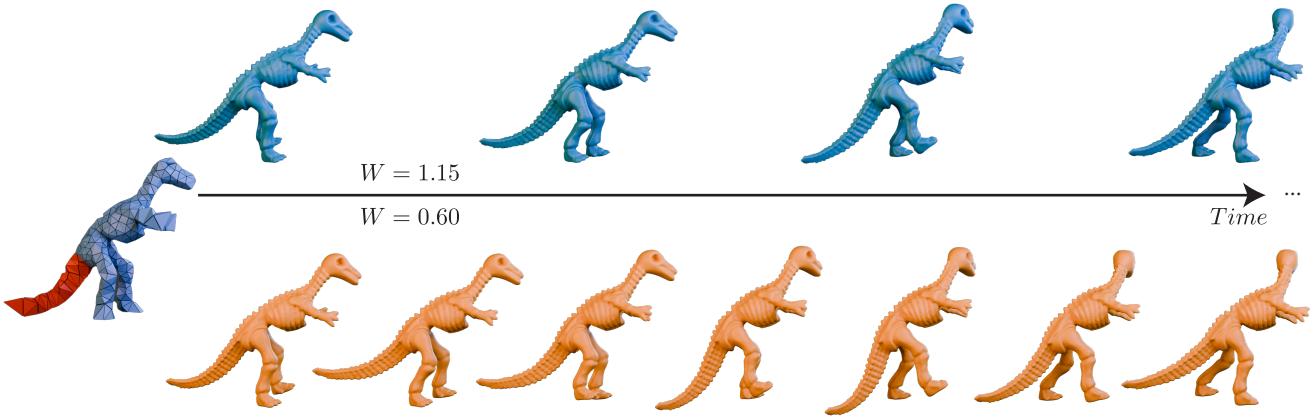


Figure 8: Walking dinosaur animation. In the top row, we show the keyframe mesh input from [BdSP09] for this DINO example. The tetrahedral mesh for embedding the fine mesh is shown on the leftmost. Only the tail part of this dinosaur model (marked in red) can move freely during the optimization; other parts are fixed to the input keyframes. The bottom row shows deformation interpolated from our parametric blendshape animation where the tail has natural vibration. Please refer to our supplemental video for the animation.

between two keyframes $t \in [t_i, t_j]$. The time integral becomes

$$W_i = \int_{t_i}^{t_{i+1}} \frac{1}{2} \|\mathbf{f}^{\text{res}}(t)\|_{\mathbf{M}^{-1}}^2 dt \simeq \frac{T_i}{2K} \sum_{k=1}^K \|\mathbf{f}_{ik}^{\text{res}}\|_{\mathbf{M}^{-1}}^2. \quad (11)$$

The first-order derivative becomes

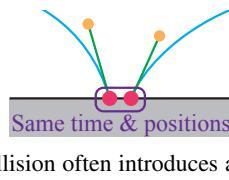
$$\frac{\partial W_i}{\partial T_i} = \frac{1}{2K} \sum_{k=1}^K \left\{ \|\mathbf{f}_{ik}^{\text{res}}\|_{\mathbf{M}^{-1}}^2 + 2T_i \frac{\partial \mathbf{f}_{ik}^{\text{res}}}{\partial T_i} \mathbf{M}^{-1} \mathbf{f}_{ik}^{\text{res}} \right\}. \quad (12)$$

Computing the second-order derivative is straightforward – we can derive the exact formula including the second-order derivative of residual force with respect to the interval $\partial^2 \mathbf{F}_{\text{res}} / \partial T_i^2$. Details on these derivatives are in Appendix A.

Note that, the change of time interval T_i , affects the sampled Jacobian in the i -th interval \mathbf{G}_{ik} in (9). In the actual optimization process, we use alternating optimization strategy. First, we iterate several steps for optimizing poses and velocities $\{\mathcal{X}, \mathcal{V}\}$, then we optimize the timing of keyframes and redo any precomputation that is affected by the updates. Finally, we go back to the step of trajectory optimization. The keyframe timing optimization does significantly change the overall animation as it is parameterized continuously with the Bézier spline. One keyframe timing optimization is typically enough for the convergence.

3.5. Various Constraints

Collision handling Our cubic Bézier parameterization smoothly interpolates vertex positions and their velocities of a mesh at keyframes (see Figure 7). Our interpolation is C_1 continuous, i.e., the velocity changes continuously. Meanwhile, the collision often introduces a discontinuous change in the velocities, which is difficult to handle as it uses our interpolation scheme. As the inset figure illustrates, we use two adjacent keyframes that share the same timing



and the same vertex positions to handle the velocity's discontinuity. These constraints (i.e., zero interval and same vertex positions) are specified by the user and applied to the optimizations as hard constraints. The user roughly inputs the keyframe timing at a collision event, and our keyframe timing optimization automatically adjusts the keyframe timing to the time the collision occurs.

Since our method differs from step-by-step simulation, we cannot use penalty-based or position-based collision handling techniques which requires small time steps [MHHR07]. Thus, we incorporate KKT conditions into our optimization framework to ensure a physically plausible shape at the colliding keyframe. During each iteration, we identify colliding vertices and update the active constraints accordingly. We also adaptively increase the number of sampled frames around the collision event to capture more accurate deformation. In the BALL example (Fig. 12), we set the material to be soft for more pronounced collision effects and sample 20 intermediate points between adjacent keyframes to achieve a plausible colliding animation.

Cyclic animation The ability to seamlessly loop animation is essential for achieving natural-looking motion that can be repeated without visible discontinuities. Last year, Jia et al. [JWLC23] explored methods to achieve these seamless loops by spacetime optimization. However, their method comes at the cost of significant computation time, as the entire trajectory, which is computed frame-by-frame, needs to be optimized to ensure smooth transitions at the boundaries of the loop. In contrast, our method allows for a much more efficient solution by introducing the blendshape representation. We put a simple constraint that the first and the last keyframes share the same positions and velocities for all the vertices. These constraints ensure continuity in both position and velocity, creating a seamless loop with significantly reduced computation cost. Furthermore, our ability to optimize the keyframe

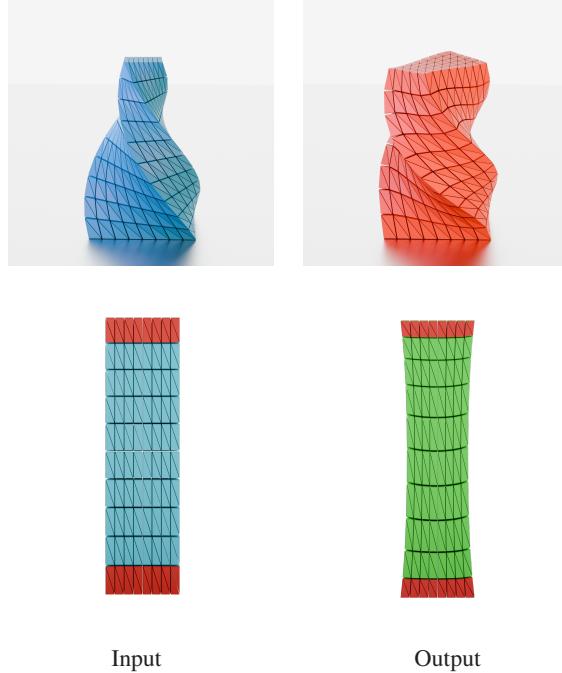


Figure 9: We optimize the animation of regular shapes in 3D (top, TWIST) and 2D (bottom, STRETCH) with cyclic constraints. The left column shows one of the in-between frames in the input animation, which is the geometric interpolation of keyframes. The right columns shows our result by fixing the top and bottom vertices and letting our method automatically complement the animation of other vertices. (Top) the volume shrinking in input animation is refined in the output. (Bottom) the shape of the stretched input keyframe is also optimized due to volume preservation.

timing is crucial in representing vibration (e.g., a pendulum), as the animation duration automatically adjusts to the vibration period.

4. Results

We demonstrate that our algorithm produces natural physical animations across various scenarios. Our implementation is written in C++ and tested on an Intel(R) Core(TM) i7-14700KF CPU with 32GB of memory. All physics models use projective dynamics with strain and volume constraints [BML*23]. We utilize the Eigen3 library for all linear solvers, and OpenMP for parallel computing. **Upon acceptance, we will publish our code on GitHub.**

Experimental setup This section presents simulations of several example scenes to evaluate our method. These scenes demonstrate various situations, such as viewing input keyframes, trajectory optimization, interactive re-optimization after keyframe edits, keyframe timing optimization, and data exportation. Table 1 summarizes scene statistics and performance metrics. The convergence criterion is met when the change in the spacetime integral W is less than 1% of its initial value. We set the damping matrix as $\mathbf{D} = 0.1\mathbf{M}$ (i.e., Rayleigh damping) for all tests. We note that the simulation

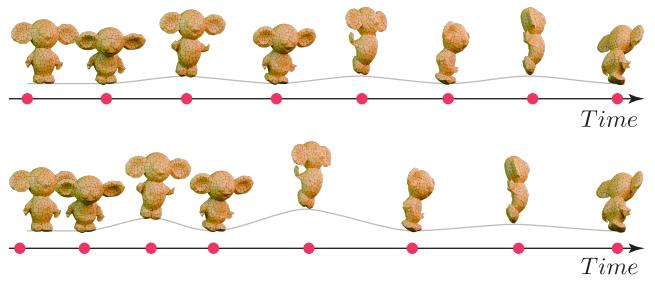


Figure 10: Timing optimization. The top row shows the original keyframes for CHEB, where all keyframes are distributed with equal time intervals. In the bottom row, we optimize the keyframe timing while constraining the total duration of the animation to remain the same. As shown in the bottom row, our method automatically adjusts the timing of the jumping keyframes, extending their intervals to create a more natural-looking jump (i.e., higher jump results in longer airborne time).

results were not very sensitive to the damping force. For the input mesh animations (DINO and ARMADILLO), we first generate a decimated mesh and use TetGen [Han15] to create a tetrahedral simulation body. We then reconstruct the full mesh animation from the tetrahedra-based motion via barycentric coordinates.

Cyclic twisting and stretching Our first experiment focuses on a simple twisting and stretching volume, as illustrated in Fig. 9. This TWIST example demonstrates the robustness of our method in generating physically plausible animation while correcting non-physical aspects of the keyframe shapes. The initial and final keyframes share the same positions and velocities to achieve a precisely periodic animation. Interestingly, the optimization converges faster as the cyclic animation reduces the number of variables.

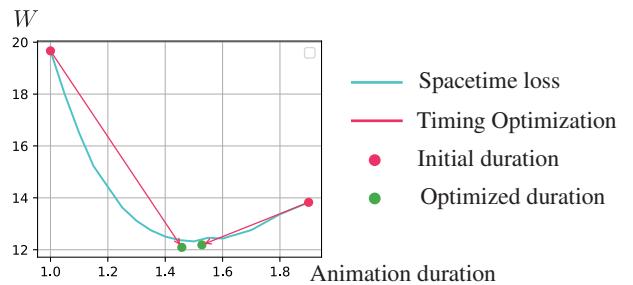


Figure 11: Convergence of the timing optimization. For the cyclic animation in Fig. 9, the blue line shows the time integration of squared residual force after keyframe position and velocity optimization for different animation duration. Starting from two different initial timing setups (red points), we optimize the keyframe positions and velocities first and then optimize the total time duration. They both converge to a similar minimum (green points).

Walking dinosaur We further compare our method with previous work on the walking dinosaur model [BdSP09]. We reduce the complexity of simulation by embedding the deformation in the

| Solver | | | | | | w/o regularization | | w/ regularization | | |
|-----------|-------|-------|------|---------------|---------------------|--------------------|-----------|-------------------|-----------|-------------------------|
| Name | Verts | Tets | Keys | Sample points | Precompute time (s) | Iter time (s) | Iter nums | Iter time (s) | Iter nums | Timing optimziation (s) |
| STRETCH | 594 | 2400 | 3 | 14 | 0.18 | 0.012 | 4 | 0.009 | 4 | 0.023 |
| TWIST | 594 | 2400 | 5 | 40 | 0.699 | 0.042 | 10 | 0.035 | 9 | 0.016 |
| DINO | 439 | 1192 | 10 | 90 | 0.91 | 0.045 | 1 | 0.030 | 3 | 0.023 |
| CHEB | 922 | 3125 | 10 | 90 | 3.771 | 0.92 | 1 | 0.080 | 5 | 0.205 |
| ARMADILLO | 2598 | 10024 | 5 | 40 | 3.073 | 0.121 | 4 | 0.100 | 5 | 0.246 |
| BALL | 316 | 1070 | 3 | 40 | 0.412 | - | - | 0.112 | 10 | 0.009 |
| SPOT | 823 | 2288 | 3 | 24 | 1.043 | - | - | 0.263 | 9 | 0.012 |

Table 1: Model statistics and performance. Verts refers to the number of vertices. Tets refers to the number of tetrahedral elements, and Keys refers to the number of keyframes. We test performance with or without the regularization term, as discussed in Section 3.3. We report the CPU time for one optimization iteration for both trajectory and timing. Here, “-” means failure of the convergence.

coarse resolution mesh. We first employ the nested-cage method [SVJ15] to refine the mesh to a specific resolution, then generate tetrahedral mesh for simulation. Similar to their approach, our tetrahedral model includes approximately 1,000 vertices. The barycentric coordinates of original mesh vertices are computed for embedding deformation, such that we output can animate the original mesh. As demonstrated in performance Table 1, Although there are 10 keyframes, and we aim at optimizing the position and tangents on these keyframes all at once, all the precomputation and iteration can be computed efficiently. Since the input animation already contains dense keyframes, a single iteration of the local-global solver yields satisfactory results. To showcase the advantages of our method, we release the constraint on the dinosaur’s tail, as shown in Fig. 8. Manually animating the swinging tail of the dinosaur that naturally goes with other body part would challenging for the novice artist. Meanwhile, our method adds vibrant secondary motion effects to the tail automatically. When the user edits any part of the keyframe, it takes only another iteration to re-optimize the entire animation, ensuring smooth integration of user modifications.

Jumping Cheburashka. We apply our method to a more complex model, CHEBURASHKA, as in our teaser (Figure 1). The animation is retargeted from human motion using Pinocchio [BP07], so the initial keyframe has rigid ears. Thus, we release the constraints on the ears in each keyframe to deform the ears. The resulting animation smooths the deformable dynamics between keyframes and adds secondary motion to the ears, creating a richer and more dynamic effect.

Keyframe timing optimization. As shown in Fig. 10, when users introduce extreme deformations into a keyframe, the time intervals before and after this keyframe are automatically adjusted to be longer, ensuring that the transition is smoother and eliminating abrupt deformations. To test the stability of timing optimization in Section 3.4, we set up an experiment shown in Fig. 11. We first record the spacetime energy W in (4) for various animation duration. The spacetime energy W is minimized when the cyclic motion fits the period of elastic vibration, thus minimizing the amount of residual forces. Then, starting from different initial guess animation durations, we tested how quickly it reached the minimum spacetime

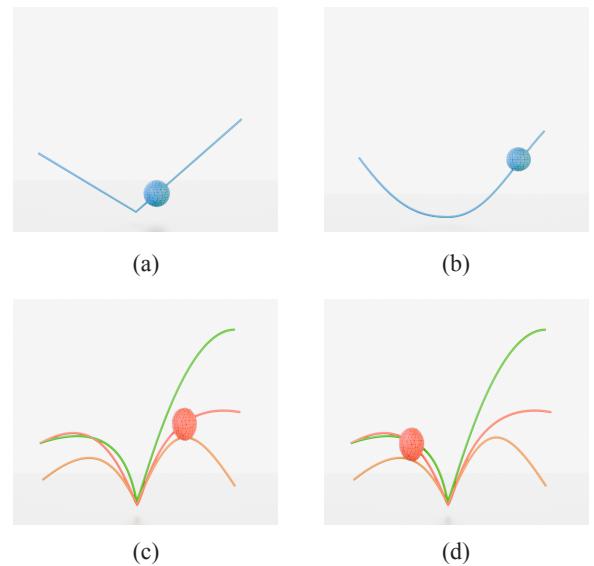


Figure 12: Real-time optimization while changing input keyframe deformation. (a) The input keyframes of BALL specify initial, colliding, and final positions. (b) Without a discontinuity, our interpolation produces a smooth but nonphysical result. (c, d) By introducing a discontinuous path for collision, users can interactively edit the initial and final positions while viewing the optimized in-between animation in real time. Our optimization adjusts the deformation in the colliding keyframe and the velocities and timings of all the keyframes.

energy W . We found that one single Newton’s method results in the duration of animation that is close to the minimum of W .

Collision In Fig. 12, we demonstrate a bouncing ball animation optimized as two trajectories before and after the collision. Users can interactively replace the starting and ending keyframes, and our method efficiently re-optimizes the bouncing motion. For collision handling, we need to track vertices that violate contact conditions and set up constraints for them in each iteration. For the exam-

ples that involve collision, Regularization is necessary to control the magnitude of updates. Fortunately, visually appealing animations can be achieved after only 2-4 iterations. Further iterations reduce the spacetime energy but do not result in noticeable visual improvements. Fig. 7 showcases a more complex model with contact. Even with a moderate-sized tetrahedral model up to 10k elements, our method optimizes the trajectory while refining the collision keyframes.

Regularization In each global step in our spacetime projective dynamics, we often regularize the update by damping the solution of a quadratic linear system. It depends on the examples whether this regularization is effective or not. As shown in Table 1, the DINO and CHEB examples converge faster without regularization, while the other models converge faster with regularization. We observe that regularization is effective if the input keyframes animation is far from the physically plausible deformation. For example, in TWIST animation in Figure 9, the deformations between the keyframes before the optimization are physically implausible as they have severe volume loss. In such a case, relaxation using regularization accelerates the convergence, as shown in Fig. 13.

File exportation Our animation representation is supported by an existing animation file format that supports blendshape animation. We use the OpenUSD library to export our animations as weighted blendshape sequences, which can be imported directly into popular tools such as Blender and Maya. We also use Blender to convert the file format to glTF. Our output can be viewed using a typical glTF viewer. The supplemental material includes several examples of glTF and USD files. Our representation is lightweight and suitable for transferring files online. As shown in Table 2, compared to the animation format that records simulation results frame-by-frame, we significantly reduce the file size and computational load, making it easier to transfer and edit animations without sacrificing much quality.

5. Limitation and Future Work

Although we optimize up to 10 keyframes at once to show the capability of our method, it is not necessary in practical usage. A direct

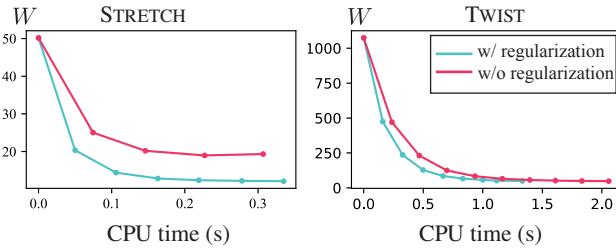


Figure 13: We test our method with and without the regularization to the quadratic system solve (i.e., identity matrix added to the coefficient matrix to damp the updates). We record the convergence of the spacetime integral with respect to CPU time. The left figure is the cyclic STRETCH animation, and the right figure is the cyclic TWIST animation, as introduced in Fig. 9.

performance improvement would be to leverage *spacetime window* [Coh92], which only selects several keyframes at a time to optimize in each step. Another improvement is to mix our method with spatial subspace method [BdSP09]. However, in our experiment, simply use the rigging weights from linear blend skinning leads to unstable optimization results. Tests on more detailed spatial subspace constructions are necessary.

Our method begins as a keyframe interpolation approach and evolves into a system that generates additional blendshapes, ultimately converting the input into weighted blendshape animations. Given that blendshape animation is widely used in modern animation workflows, a promising extension of our method would be to support blendshape animations as input. This could involve complementing the blendshape space with optimized new shapes.

In this paper, we have only focused on solid deformable objects. However, theoretically, our method could be extended to any material that supports local-global solvers. For instance, blendshape animation is also common in cloth animation. Given that projective dynamics has been extended to cloth simulation, it is feasible to extend our approach to generate cloth-specific blendshapes, enhancing its controlability and portability.

6. Conclusion

We present BlendSim, a novel framework that transitions the foundation of physics-based simulations from discrete frame-by-frame methods to continuous interpolation trajectories using blendshapes. Our approach establishes a parametric connection between blendshapes and physical realism, enabling editable and transferable simulations. By compressing the simulation in the time domain through predefined cubic spline interpolation, we maintain high spatial detail while significantly reducing computational complexity.

BlendSim can be seamlessly integrated into modern animation workflows based on keyframes and its compact blendshape representation is fully compatible with industry-standard formats such as USD and

| Name | File size (.usdc, MB) | | Animation time (s) |
|-----------|-----------------------|----------|--------------------|
| | Full-frame (60 FPS) | BlendSim | |
| STRETCH | 32.8 | 2.4 | 1.58 |
| TWIST | 41.5 | 4.5 | 2.0 |
| DINO | 64.3 | 9.7 | 3.15 |
| CHEB | 72.9 | 12.6 | 2.7 |
| BALL | 5.7 | 1.1 | 0.6 |
| SPOT | 20.1 | 2.3 | 1.0 |
| ARMADILLO | 45.6 | 8.4 | 2.4 |

Table 2: We export our outputs in USD file format for both 60 FPS frame-by-frame animation and to our blendshape animation. The file size of frame-by-frame animation can be significantly large when the frames of animation increase. Meanwhile, our method only needs to store blendshapes and weight animation curves; our file size only depends on the number of keyframes.

glTF. Through the use of projective dynamics, we achieve a balance between computational efficiency and accurate physics-based motion. Our framework supports a wide range of constraints, including collisions, cyclic motions, and dynamic keyframe adjustments, making it applicable to both traditional character animation and more complex deformable object simulations. Through this work, we aim to bridge the gap between high-fidelity physics simulations and interactive animation design, offering a new level of control and efficiency to animators and simulation artists.

References

- [ACZW19] AHARON, IDO, CHEN, RENJIE, ZORIN, DENIS, and WEBER, OFIR. “Bounded distortion tetrahedral metric interpolation”. *ACM Transactions on Graphics (TOG)* 38.6 (2019), 1–17 4.
- [BdSP09] BARBIĆ, JERNEJ, da SILVA, MARCO, and POPOVIĆ, JOVAN. “Deformable object animation using reduced optimal control”. *ACM SIGGRAPH 2009 papers*. 2009, 1–9 2–4, 7, 8, 10.
- [BML*23] BOUAZIZ, SOFIEN, MARTIN, SEBASTIAN, LIU, TIENTIAN, et al. “Projective dynamics: Fusing constraint projections for fast simulation”. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, 787–797 2, 4, 6, 8.
- [BMWG07] BERGOU, MIKLÓS, MATHUR, SAURABH, WARDETZKY, MAX, and GRINSPUN, EITAN. “Tracks: toward direactable thin shells”. *ACM Transactions on Graphics (TOG)* 26.3 (2007), 50–es 4.
- [BP07] BARAN, ILYA and POPOVIĆ, JOVAN. “Automatic rigging and animation of 3D characters”. *ACM Trans. Graph.* 26.3 (July 2007), 72–es. ISSN: 0730-0301. DOI: [10.1145/1276377.1276467](https://doi.org/10.1145/1276377.1276467). URL: <https://doi.org/10.1145/1276377.1276467>.
- [BSC16] BARRIELLE, VINCENT, STOIBER, NICOLAS, and CAGNIART, CÉDRIC. “Blendforces: A dynamic framework for facial animation”. *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, 341–352 4.
- [BSG12] BARBIĆ, JERNEJ, SIN, FUNSHING, and GRINSPUN, EITAN. “Interactive editing of deformable simulations”. *ACM Transactions on Graphics (TOG)* 31.4 (2012), 1–8 5.
- [BZC*23] BENCHEKOUN, OTMAN, ZHANG, JIAYI ERIS, CHAUDHURI, SIDDHARTH, et al. “Fast Complementary Dynamics via Skinning Eigenmodes”. *arXiv preprint arXiv:2303.11886* (2023) 4.
- [CO18] CASAS, DAN and OTADUY, MIGUEL A. “Learning nonlinear soft-tissue dynamics for interactive avatars”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), 1–15 4.
- [Coh92] COHEN, MICHAEL F. “Interactive spacetime control for animation”. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, 293–302 3, 10.
- [DWM*21] DU, TAO, WU, KUI, MA, PINGCHUAN, et al. “Diffppd: Differentiable projective dynamics”. *ACM Transactions on Graphics (TOG)* 41.2 (2021), 1–21 4.
- [EST*20] EGGER, BERNHARD, SMITH, WILLIAM AP, TEWARI, AYUSH, et al. “3d morphable face models—past, present, and future”. *ACM Transactions on Graphics (ToG)* 39.5 (2020), 1–38 4.
- [FP03] FANG, ANTHONY C and POLLARD, NANCY S. “Efficient synthesis of physically valid human motion”. *Acm transactions on graphics (tog)* 22.3 (2003), 417–426 3.
- [GAU10] GONZALEZ CASTRO, GABRIELA, ATHANASOPOULOS, MICHAEL, and UGAIL, HASSAN. “Cyclic animation using partial differential equations”. *The Visual Computer* 26 (2010), 325–338 4.
- [Gle97] GLEICHER, MICHAEL. “Motion editing with spacetime constraints”. *Proceedings of the 1997 symposium on Interactive 3D graphics*. 1997, 139–ff 3.
- [Han15] HANG, SI. “TetGen, a Delaunay-based quality tetrahedral mesh generator”. *ACM Trans. Math. Softw* 41.2 (2015), 11 8.
- [HSvTP12] HILDEBRANDT, KLAUS, SCHULZ, CHRISTIAN, von TYCOWICZ, CHRISTOPH, and POLTHIER, KONRAD. “Interactive spacetime control of deformable objects”. *ACM transactions on graphics (TOG)* 31.4 (2012), 1–8 2–4.
- [HTZ*10] HUANG, JIN, TONG, YIYING, ZHOU, KUN, et al. “Interactive shape interpolation through controllable dynamic deformation”. *IEEE Transactions on Visualization and Computer Graphics* 17.7 (2010), 983–992 4.
- [IKNP16] ICHIM, ALEXANDRU EUGEN, KAVAN, LADISLAV, NIMIER-David, MERLIN ELÉAZAR, and PAULY, MARK. “Building and animating user-specific volumetric face rigs”. *SCA’16: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. The Eurographics Association. 2016 4.
- [JWLc23] JIA, SHIYANG, WANG, STEPHANIE, LI, TZU-MAO, and CHERN, ALBERT. “Physical Cyclic Animations”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3 (2023), 1–18 2, 4, 7.
- [KA08] KASS, MICHAEL and ANDERSON, JOHN. “Animating oscillatory motion with overlap: wiggly splines”. *ACM SIGGRAPH 2008 papers*. 2008, 1–8 3.
- [KK19] KADLEČEK, PETR and KAVAN, LADISLAV. “Building accurate physics-based face models from data”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2.2 (2019), 1–16 4.
- [LHDG*14] LI, SIWANG, HUANG, JIN, de GOES, FERNANDO, et al. “Space-time editing of elastic motion through material optimization and reduction”. *ACM Transactions on Graphics (TOG)* 33.4 (2014), 1–10 3, 4.
- [LLK19a] LI, JING, LIU, TIENTIAN, and KAVAN, LADISLAV. “Fast simulation of deformable characters with articulated skeletons in projective dynamics”. *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450366779. DOI: [10.1145/3309486.3340249](https://doi.org/10.1145/3309486.3340249). URL: <https://doi.org/10.1145/3309486.3340249>.
- [LLK19b] LI, JING, LIU, TIENTIAN, and KAVAN, LADISLAV. “Fast simulation of deformable characters with articulated skeletons in projective dynamics”. *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450366779. DOI: [10.1145/3309486.3340249](https://doi.org/10.1145/3309486.3340249).
- [LLK20] LI, JING, LIU, TIENTIAN, and KAVAN, LADISLAV. “Soft articulated characters in projective dynamics”. *IEEE Transactions on Visualization and Computer Graphics* 28.2 (2020), 1385–1396 4.
- [MHHR07] MÜLLER, MATTHIAS, HEIDELBERGER, BRUNO, HENNIX, MARCUS, and RATCLIFF, JOHN. “Position based dynamics”. *Journal of Visual Communication and Image Representation* 18.2 (2007), 109–118 7.
- [NM93] NGO, J THOMAS and MARKS, JOE. “Spacetime constraints revisited”. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993, 343–350 3.
- [PW99] POPOVIĆ, ZORAN and WITKIN, ANDREW. “Physically based motion transformation”. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999, 11–20 3.
- [RGBC96] ROSE, CHARLES, GUENTER, BRIAN, BODENHEIMER, BOBBY, and COHEN, MICHAEL F. “Efficient generation of motion transitions using spacetime constraints”. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, 147–154 3.
- [SHP04] SAFONOVA, ALLA, HODGINS, JESSICA K, and POLLARD, NANCY S. “Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces”. *ACM Transactions on Graphics (ToG)* 23.3 (2004), 514–521 3.
- [SP05] SULEJMANPAŠIĆ, ADNAN and POPOVIĆ, JOVAN. “Adaptation of performed ballistic motion”. *ACM Transactions on Graphics (TOG)* 24.1 (2005), 165–179 3.

- [SVJ15] SACHT, LEONARDO, VOUGA, ETIENNE, and JACOBSON, ALEC. “Nested cages”. *ACM Transactions on Graphics (TOG)* 34.6 (2015), 1–14 9.
- [SvTSH14] SCHULZ, CHRISTIAN, von TYCOWICZ, CHRISTOPH, SEIDEL, HANS-PETER, and HILDEBRANDT, KLAUS. “Animating deformable objects using sparse spacetime constraints”. *ACM Transactions on Graphics (TOG)* 33.4 (2014), 1–10 2, 3.
- [VSSH15] VON-TYCOWICZ, CHRISTOPH, SCHULZ, CHRISTIAN, SEIDEL, HANS-PETER, and HILDEBRANDT, KLAUS. “Real-time nonlinear shape interpolation”. *ACM Transactions on Graphics (TOG)* 34.3 (2015), 1–10 4.
- [WK88] WITKIN, ANDREW and KASS, MICHAEL. “Spacetime Constraints”. *SIGGRAPH Comput. Graph.* 22.4 (June 1988), 159–168. ISSN: 0097-8930. DOI: [10.1145/378456.378507](https://doi.org/10.1145/378456.378507). URL: <https://doi.org/10.1145/378456.378507>, 2, 3.
- [WP09] WAMPLER, KEVIN and POPOVIĆ, ZORAN. “Optimal gait and form for animal locomotion”. *ACM Transactions on Graphics (TOG)* 28.3 (2009), 1–8 3.
- [WU23] WU, YUHAN and UMETANI, NOBUYUKI. “Two-Way Coupling of Skinning Transformations and Position Based Dynamics”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3 (2023), 1–18 4.
- [ZBLJ20] ZHANG, JIAYI ERIS, BANG, SEUNGBAE, LEVIN, DAVID I.W., and JACOBSON, ALEC. “Complementary Dynamics”. *ACM Transactions on Graphics* (2020) 4.

Appendix A:

First we define the derivative of position, velocity and acceleration wrt. to time respectively. Suppose $t \in [\mathbf{t}_i, \mathbf{t}_{i+1}]$ then

$$\frac{\partial \mathbf{x}(t)}{\partial T_i} = b_1(\theta) \mathbf{v}_i - b_2(\theta) \mathbf{v}_{i+1}, \quad (13)$$

$$\frac{\partial \dot{\mathbf{x}}(t)}{\partial T_i} = -\frac{1}{T_i^2} \mathbf{B}'_i \mathbf{q}_i + \frac{1}{T_i} (b'_1(\theta) \mathbf{v}_i - b'_2(\theta) \mathbf{v}_{i+1}), \quad (14)$$

$$\frac{\partial \ddot{\mathbf{x}}(t)}{\partial T_i} = -\frac{2}{T_i^3} \mathbf{B}''_i \mathbf{q}_i + \frac{1}{T_i^2} (b''_1(\theta) \mathbf{v}_i - b''_2(\theta) \mathbf{v}_{i+1}), \quad (15)$$

$$\frac{\partial^2 \mathbf{x}(t)}{\partial T_i^2} = 0 \quad (16)$$

$$\frac{\partial^2 \dot{\mathbf{x}}(t)}{\partial T_i^2} = \frac{2}{T_i^3} \mathbf{B}'_i \mathbf{q}_i - \frac{2}{T_i^2} (b'_1(\theta) \mathbf{v}_i - b'_2(\theta) \mathbf{v}_{i+1}), \quad (17)$$

$$\frac{\partial^2 \ddot{\mathbf{x}}(t)}{\partial T_i^2} = \frac{6}{T_i^4} \mathbf{B}''_i \mathbf{q}_i - \frac{4}{T_i^3} (b''_1(\theta) \mathbf{v}_i - b''_2(\theta) \mathbf{v}_{i+1}). \quad (18)$$

Although it seems complicated, each derivative is computed by interpolation of current vertex positions and velocities. Substitute these derivatives into \mathbf{f}^{res} in (4), we obtain

$$\frac{\partial \mathbf{f}^{\text{res}}(t)}{\partial \mathbf{T}_i} = \mathbf{M} \frac{\partial \ddot{\mathbf{x}}(t)}{\partial T_i} + \mathbf{D} \frac{\partial \dot{\mathbf{x}}(t)}{\partial T_i} - \frac{\partial \mathbf{f}^{\text{in}}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(t)}{\partial T_i}, \quad (19)$$

$$\frac{\partial^2 \mathbf{f}^{\text{res}}(t)}{\partial \mathbf{T}_i^2} = \mathbf{M} \frac{\partial^2 \ddot{\mathbf{x}}(t)}{\partial T_i^2} + \mathbf{D} \frac{\partial^2 \dot{\mathbf{x}}(t)}{\partial T_i^2} - \frac{\partial \mathbf{f}^{\text{in}}}{\partial \mathbf{x}} \frac{\partial^2 \mathbf{x}(t)}{\partial T_i^2}. \quad (20)$$