ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2026
D. Levin and M. Chu
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 8

SSILK : Self-Supervised Integration of Latent Kinematics for Joint-Driven Neural Garments

M. Perepichka[1,2], A. Schoentgen[1], E. Paquette[3], and T. Popa[2]

[1]Ubisoft La Forge, Canada
[2]Concordia University, Canada
[3]École de technologie supérieure, Canada

Figure 1: SSILK generates physically plausible motion even for loose garments, a long-standing challenge for pose-based methods.

Abstract

Garment animation systems are essential for high-fidelity digital characters in various applications such as films and video games. We propose SSILK, a neural network-based method that is explicitly designed to handle loose garments such as capes and robes, garment types that are notoriously hard to simulate via pose-based garment neural simulators. We achieve this through three contributions. First, we formulate our method as a latent-space time integration scheme inspired by Verlet integration, resulting in more temporally stable garment dynamics compared to prior methods. Second, we employ a hybrid proxy-joint and blendshape decoder to represent full SO(3) transformations that can be too challenging to capture using blendshapes alone. Finally, we propose a data-augmentation strategy to improve performance on out-of-distribution scenarios. We demonstrate the effectiveness of our method on various garment and character animations and compare it to the state of the art.

Keywords: cloth simulation, neural networks, physics-based learning, character animation, deep learning

1. Introduction

Character garment animation has been an active field of inquiry in both industry and academia in the last few decades. Fueled by the expansion of digital media domains such as animated films, digital avatars, video games, and the visual effects (VFX) industry, the demand for high fidelity 3D characters has been ever growing, and with it, the demand for systems capable of simulating convincing clothing. Complex pipelines have been designed to simulate character clothing, with fundamental technological decisions driven by a trade-off between quality and compute. Offline applications such as VFX and feature films have extensively utilized physics-based simulation in order to create realistic draped garments. Other applications such as video games or real-time avatar systems, which are much more limited in their compute budget, have largely relied on hierarchical joint-based solutions, with skeletal hierarchies coupled with skinning weights driving the deformation of character clothing. Even today, animated character garments in video games are largely animated by Linear Blend Skinning (LBS) [MTLT88] and corrective blendshapes [LCF00]. Although introduced decades ago, these methods continue to be highly effective owing to their simple formulation and efficient computation.

The computational limitations of consumer hardware have inspired extensive research into achieving realistic cloth animation under strict compute constraints. Subspace-based methods have been proposed in order to project the dynamics of various elastic deformations onto smaller, more wieldy subspaces [PW89, BJ05, HTC*14]. With advances in deep learning, several methods [HDDN19, BME20a, BMTE21, PLPM20, SOC19, PMJ*22] use neural networks to model garment deformations, trained on datasets obtained from either data capture or physics simulation. While these methods can achieve real time performance, they typically rely on a computationally expensive training-data generation phase and may exhibit visual artifacts, partly due to the use of simple MSE-based loss formulations.

Self-supervised pose-based neural cloth simulators [BME20b, SOC22,BME22,CCP24] have been explored to address these shortcomings. This alternative paradigm directly formulates the training objective as the minimization of physical potential energies of the character garment, avoiding the need for prior dataset generation. While these pose-based methods have had success, they suffer from their reliance on corrective blendshapes over skinned character kinematics [LCF00] and from a lack of temporal integration, resulting in poor performance on loose garments and extreme motions that are poorly represented in the training data. Increasing the blendshape count can improve animation quality, but comes at a significant GPU memory cost, which is a critical constraint in real-time applications.

An alternative graph-based neural cloth simulator paradigm has been explored [GBH23, GBB*24, LWK24], utilizing Graph Neural Networks in order to predict vertex accelerations. Trained with similar physics-based loss functions, this class of method captures richer garment dynamics due to its explicit time integration and better generalizes across character garments. However, its computational cost is orders of magnitude higher than that of pose-based methods, making them less suitable for performance-intensive applications. As a result, there remains a clear research gap in achieving realistic support for loose garments in real-time settings.

In this work, we present **SSILK** - Self-Supervised Integration of Latent Kinematics for Joint-Driven Neural Garments. At its core, our method belongs to the family of pose-based neural cloth simulators and achieves real-time performance for assets of complexity comparable to those used in the video game industry. Our method overcomes several shortcomings of existing pose-based neural cloth simulators trained in a self-supervised manner, and makes the following contributions:

- **Explicit Latent Space Verlet-inspired Integration.** Instead of solely relying on recurrent neural networks to learn garment dynamics, we adopt a Verlet-inspired latent integration scheme in order to enforce garment dynamics.
- **Hybrid Proxy-Joint and Blendshape Decoder.** We propose a hybrid proxy-joint and blendshape formulation that provides a more compact and expressive representation for loose garments than blendshapes alone.
- **Novel Data Augmentation.** We propose a series of data-augmentation steps to improve performance on complex motions such as cartwheels, handstands, or backflips, where garment behavior deviates significantly from skeletal motion.

2. Related Work

Cloth simulation is a well-established research field with a rich body of prior work. We first review classical cloth simulation and subspace methods, and then discuss learning-based approaches, ranging from supervised to unsupervised techniques.

2.1. Cloth Simulation

Traditional garment simulation has been heavily researched in computer graphics [TPBF87], relying primarily on continuum mechanics and numerical integration. Foundational approaches utilize the finite element method or mass-spring systems, often requiring implicit integration schemes to maintain stability under large time steps [BW23]. To avoid the computational overhead of solving large linear systems, Position-Based Dynamics (PBD) [MHHR07] and its variant XPBD [MMC16] have become the industry standard for real-time applications. While PBD offers unconditional stability, real time framerates on high density meshes for applications such as videogames remain cost prohibitive due to the iterative nature of the solvers and the bottleneck of collision detection. More recently, Vertex Block Descent (VBD) [CLYY24] and AVBD [GDY25] further improved performance via local vertex updates. Nevertheless, high-fidelity real time cloth simulation in real time settings remains an open problem in the field.

An unavoidable issue with traditional cloth simulation methods is that computational cost scales with vertex count, often nonlinearly. Large meshes require expensive implicit iterative solvers, with collision detections phases typically scaling as $\mathcal{O}(n \log n)$ in the number of vertices. Reducing the size of the computational space was first explored by Pentland et al. [PW89]. Later, Barbivc et al. [BJ05] extended this to PCA basis computed on datasets of precomputed cloth simulation data.

2.2. Supervised Methods

With the rise of deep learning, the idea of utilizing neural networks for predicting garment dynamics gained popularity. Holden et al. [HDDN19] propose utilizing neural networks to predict garment dynamics on linear subspaces generated via PCA. Bertiche et al. [BME20a] propose to utilize conditional variational autoencoders to learn non-linear garment deformation latent spaces. Bertiche et al. [BMTE21] propose utilizing graph neural networks coupled with Multi-Layer Perceptrons (MLP) in-order to predict quasi-static garment states as a function of a character's pose. Patel et al. [PLPM20] explicitly decompose garment deformations into high-frequency and low-frequency detail and predict it as a function of pose, shape, and style. Santesteban et al. [SOC19] utilize recurrent neural networks to predict garment deformation as a function of the body shape and dynamics. Pan et al. [PMJ*22] propose utilizing virtual joints to predict garment deformations from simulated data. Jin et al. [JOG*24] explore a similar route by augmenting rope chain simulations with neural networks. While capable of achieving high quality garment deformations, these methods suffer from

a few fundamental flaws. They rely on large amounts of ground truth training cloth simulation data which must be precomputed before neural network training; any change in garment dynamics not only requires neural network retraining, but also re-generation of the target simulation data. The reliance on ground truth simulation data also limits generalization, with typically poor performance in out-of-distribution scenarios. Additionally, as the training objective is typically a simple mean squared error loss on garment vertices, these methods may exhibit visual artifacts related to garment–body interaction and often rely on post-processing steps to further improve quality. In contrast, the method we introduce is fully unsupervised and trained using physics-based objectives, eliminating the need for precomputed simulation data. A concurrent work, PhySkin [SCG*26], also adopts an unsupervised formulation utilizing proxy joints. However, it is restricted to static pose reconstruction and does not model temporal dynamics. Conversely, our approach explicitly captures dynamic garment behavior over time. Furthermore, we combine proxy joints with learned blendshapes in a hybrid representation, enabling the modeling of both large-scale motion and fine-scale deformations.

2.3. Self-Supervised Methods

To address issues with supervised methods, Bertiche et al. [BME20b] propose to train a self-supervised neural network which predicts garment positions as a function of the character's pose. The novelty of this method is its training procedure: self-supervised with no ground-truth simulation data. Instead, cloth energy potentials are expressed as loss functions that are directly minimized in the training process. Follow-up works such as SNUG [SOC22] and NCS [BME22] have extended this approach to support dynamic garment deformations, with the expression of cloth inertia as an additional energy to minimize. A strong selling point for these methods is their ease of integration into existing game pipelines, leveraging blendshapes [BMTE21] and linear blend skinning [MTLT88], which are heavily supported in most modern game engines. However, due to their reliance on skinning with respect to the character skeleton, these methods often fail to generalize to loose garments such as capes where the cloth motion does not closely follow skeletal kinematics. Chen et al. [CCP24] tackle the issue of skeletal dependence by utilizing gaussian radial basis functions based skinning, but the method still suffers from the fundamental limitation of skinning loose garments to a rigid character joint hierarchy. Recently, Li et al. [LLL*25] proposed a self-supervised approach to latent integration, first learning an autoencoder on a generated simulation dataset, and subsequently learning a neural latent integrator on the autoencoder's subspace. However, the approach still relies on the generation of ground-truth simulation data and is evaluated on dynamic sequences that do not directly target draped character garments.

Grigorev et al. [GBH23] propose an alternative approach with HOOD, which employs a Graph Neural Network to predict vertex accelerations as a function of the current vertex states. Subsequent methods such as ContourCraft [GBB*24] and SENC [LWK24] have added various improvements to these methods, such as self-collision resolution. While Graph Neural Network (GNN) based methods typically offer superior performance compared to traditional position-based approaches, they have several shortfalls that make them unsuitable for interactive applications. They require the creation of dynamic garment-body collision edges on a per-frame basis. It relies on a per-vertex latent embedding of size 128, which results in relatively high memory consumption; by comparison, NCS [BME22] uses a total latent dimensionality of 512. Finally, this method relies on several message passing steps per frame to propagate information throughout their hierarchical graph. These factors result in performance metrics that are unsuitable for real time applications.

3. Methodology

The goal of this work is to develop a neural simulator capable of generating temporally consistent sequences of draped garment vertex positions for animated characters in real time. Our approach is designed to address key limitations of existing pose-based neural cloth simulators, in particular their inability to preserve independent garment momentum, model garment behavior that depends on past motion, and robustly generalize to long motion sequences and loose garments. Practical deployment considerations, including GPU memory constraints in game engines, inform the design of our simplified decoder architecture.

Prior pose-based approaches typically parametrize garment dynamics directly as a function of character pose and pose derivatives, often via recurrent architectures. While highly effective for relatively tight garments, such formulations implicitly tie garment motion to skeletal kinematics. As a result, dynamic states collapse when pose derivatives vanish, preventing the representation of independent momentum, locking behaviors, or history-dependent configurations. These limitations make pose-based methods brittle in out-of-distribution scenarios and poorly suited for garments whose dynamics diverge significantly from the underlying skeleton.

To overcome these limitations, we propose a latent neural simulator that explicitly integrates garment dynamics within a compact latent space, thereby decoupling garment motion from instantaneous character kinematics. Our method, illustrated in Figure 2, consists of three main components: (i) a latent initializer that predicts consistent initial garment states from static pose information (Section 3.1); (ii) a latent residual stepper that advances the garment state forward in time via explicit integration (Section 3.2); and (iii) a neural decoder that reconstructs world-space garment geometry from latent states (Section 3.3).

This formulation enables stable long-horizon rollouts and produces physically plausible garment dynamics, as illustrated in Figure 9. For certain highly dynamic loose-garment scenarios, additional decoder expressiveness can be beneficial. Section 3.4 presents proxy joints as a decoder-level augmentation that builds on the existing latent dynamics formulation.

All components are trained jointly in a fully self-supervised manner using physics-inspired energy losses (Sections 3.5 and 3.6). Implementation and training details are provided in Section 3.7.

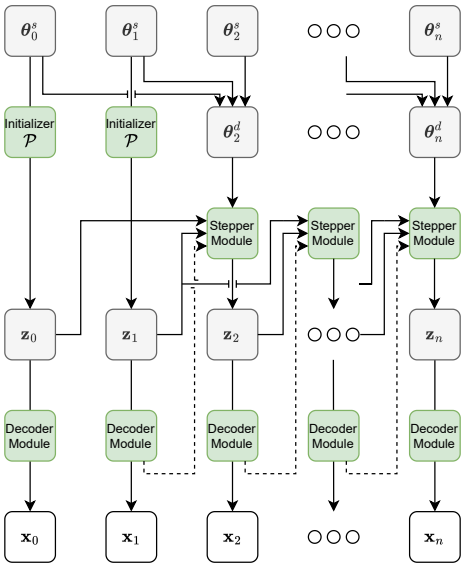


Figure 2: Overview of the SSILK pipeline. Taking the first two character poses as input, we initialize latent states $\mathbf{z}_0$ and $\mathbf{z}_1$. Each latent state is independently decoded to generate both $\mathbf{x}_t$ and the proxy animation feedback to the stepper module. Latents are integrated using the stepper module.

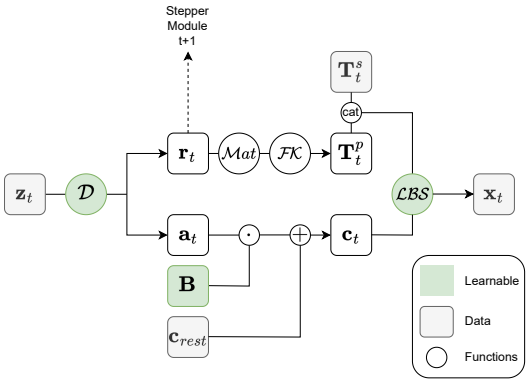


Figure 3: Our decoder module architecture. Taking as input a given latent state $\mathbf{z}_t$, our neural network $\mathcal{D}$ outputs blendshape activations $\mathbf{a}_t$ as well as joint rotations $\mathbf{r}_t$. We reconstruct canonical space offsets by multiplying $\mathbf{a}_t$ with a learned blendshape matrix $\mathbf{B}$ and adding the rest garment positions $\mathbf{c}_{rest}$. We convert the 6D rotation to transformation matrices and perform forward kinematics to yield proxy joint transformation matrices $\mathbf{T}_t^p$. These are then concatenated with the character's skeletal transformation matrices $\mathbf{T}_t^s$. Finally, we perform linear blend skinning to yield the world space garment coordinates $\mathbf{x}_t$

3.1. Latent Initialization

Our simulator relies on the skeletal kinematics to initialize the garment state at the start of each rollout. At timestep t, we construct a static pose descriptor θ_t^s following prior work [BME22]. This descriptor consists of flattened local joint rotations encoded using a continuous 6D representation, together with unposed joint-local gravity directions. It captures sufficient geometric context to characterize the quasi-static garment configuration associated with a given body pose. A neural latent initializer $\mathcal{P}$ maps this static pose descriptor to a latent garment state:

$$\mathbf{z}_t = \mathcal{P}(\theta_t^s). \tag{1}$$

For each animation sequence, the initializer is applied to the first two poses, producing initial latent states $\mathbf{z}_0$ and $\mathbf{z}_1$, which serve as starting points for subsequent dynamic rollouts. Initializing latents to zero leads to unstable optimization and large spurious velocities. In contrast, the learned initializer acts as a pose-conditioned quasi-static solver, placing the garment in a physically plausible region of the latent space and enabling stable convergence under self-supervised physics-based losses. These initialized states define the garment configuration at rollout start; all subsequent motion is generated dynamically by the latent integration scheme described next.

3.2. Latent Integration

Latent Space Representation. We represent the garment state at each timestep using a compact latent vector $\mathbf{z}_t$, which implicitly encodes both configuration and momentum. Unlike prior pose-based formulations that tightly couple garment dynamics to skeletal motion, our latent representation evolves independently of pose derivatives. This allows the simulator to preserve garment momentum when the character becomes stationary and to represent path-dependent behaviors such as folding, locking, or delayed relaxation.

Recent GNN-based simulators address this issue by predicting per-vertex accelerations and performing explicit integration in full vertex space. Our approach can be viewed as a reduced-order analogue of this idea, performing explicit time integration directly in a learned garment subspace.

Explicit Integration in Latent Space. To advance garment dynamics forward in time, we adopt a Verlet-inspired integration scheme applied directly in latent space, following the subspace simulation formulation of Holden et al. [HDDN19]:

$$\bar{\mathbf{z}}_t = \alpha z_{t-1} + \beta(z_{t-1} - z_{t-2}), \tag{2}$$

where α and β are learnable integration parameter vectors. This extrapolated state is then refined using a neural residual stepper $\mathcal{S}$, implemented as an MLP:

$$\mathbf{z}_t = \bar{\mathbf{z}}_t + \mathcal{S}([\bar{\mathbf{z}}_t, \mathbf{z}_{t-1}, \theta_t^d]). \tag{3}$$

The dynamic pose descriptor θ_t^d concatenates static pose descriptors from the current and two previous timesteps, together with global root rotations and translations. Conditioning the residual on pose history provides essential motion context while preventing drift during long rollouts and enabling garment dynamics.

3.3. Decoder

The neural decoder reconstructs world-space garment geometry from latent states. Given a latent state $\mathbf{z}_t$, the decoder predicts blendshape coefficients:

$$\mathbf{a}_t = \mathcal{D}(\mathbf{z}_t). \tag{4}$$

These coefficients parameterize a linear combination of learned blendshape basis vectors $\mathbf{B}$, producing a deformed garment mesh in canonical rest space:

$$\mathbf{c}_t = \mathbf{B}\mathbf{a}_t + \mathbf{c}_{rest}. \tag{5}$$

The canonical garment mesh is then transformed into world space using standard LBS, driven by the character's skeletal transformations:

$$\mathbf{x}_t = \mathcal{LBS}(\mathbf{c}_t, \mathbf{W}^g, \mathbf{T}_t^s), \tag{6}$$

where $\mathbf{W}^g$ denotes garment skinning weights and $\mathbf{T}_t^s$ corresponds to the character joint transformation matrices.

Figure 3 illustrates this decoder architecture. Skinning initializes the garment in a reasonable pose-dependent configuration, which is crucial for convergence given the highly non-convex nature of physics-based self-supervised loss landscapes. Moreover, the decoder predicts only corrective offsets rather than full vertex trajectories, simplifying learning and ensuring compatibility with traditional animation pipelines.

However, this representation assumes that garment motion remains close to a skinned pose and can be expressed through relatively small corrective offsets, an assumption that can break down for loose garments exhibiting large, rigid-body-like motion. We address this limitation in the following section.

3.4. Proxy Joints

The decoder formulation described above relies solely on corrective blendshapes applied on top of skeletal skinning. While effective for tight-fitting garments, blendshapes are less efficient at representing rigid transformations. Consequently, large-scale garment motion must be approximated using linear vertex displacements, placing unnecessary burden on the latent space and destabilizing long-horizon dynamics. Increasing the blendshape count helps mitigate these issues, but this comes at the cost of having to store additional blendshapes in GPU memory.

In some existing real-time animation pipelines and supervised learning frameworks [PMJ*22], this limitation has been addressed using proxy joints, an auxiliary joint hierarchy embedded in the garment, which act as auxiliary deformation handles for semi-rigid motion such as muscle bulges or cloth-like appendages. Inspired by this practice, we extend our decoder with proxy joints as illustrated in Figure 4.

We subdivide garment vertices into three classes: pinned, tight and loose vertices (see Figure 4). Pinned vertices strictly follow the output of linear blend skinning from the character and have no corrective blendshapes. Tight vertices are also only skinned to the character but will have corrective blendshapes. Finally loose vertices are vertices that will be deformed using the proxy joint kinematics coupled with blendshapes. This classification is currently performed manually for each garment type using vertex coloring; however, this process could be automated, which we leave for future work.

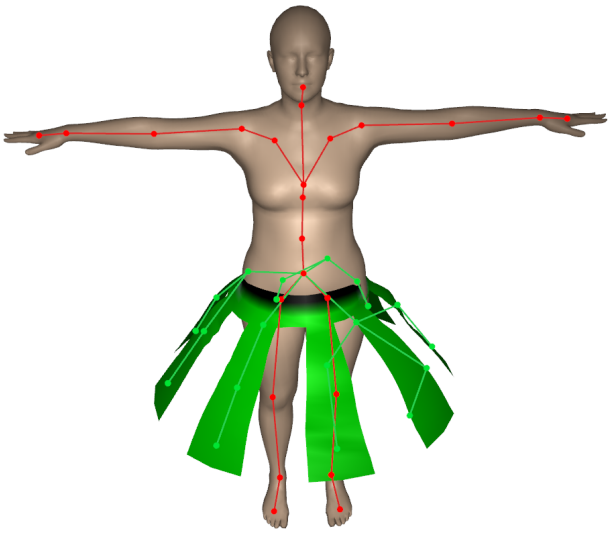


Figure 4: A demonstration of proxy joint initialization for a fringe skirt garment. Green garment vertices and joints are labeled as "Loose", red joints are labeled as "Tight", and black vertices are labeled as "Pinned". Note that this figure also shows proxy joints sampled behind the character.

Proxy joints act as coarse deformation handles capable of representing large-scale, low-frequency garment motion using full 3D rigid-body transformations. Importantly, proxy joints affect only the decoder output representation of the model: the integration scheme and loss formulation remain unchanged.

Concretely, the decoder is augmented to predict local rotations for proxy joints, which are propagated through the proxy hierarchy via forward kinematics to produce global proxy transformations. Thus we augment Equation 4 to:

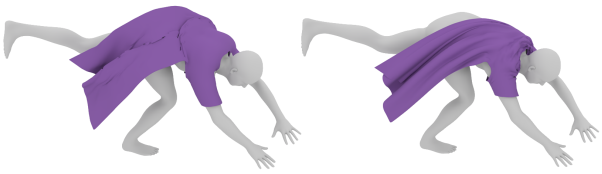
$$[\mathbf{a}_t, \mathbf{r}_t^p] = \mathcal{D}(\mathbf{z}_t), \tag{7}$$

where $\mathbf{r}_t^p$ represents concatenated proxy joint rotations in the 6D format [ZBL*19]. Full local proxy joint transformation matrices $\bar{\mathbf{T}}_t^p$ are subsequently reconstructed via orthogonalization and forward kinematics is performed to yield the global proxy joint transformation matrices $\mathbf{T}_t^p$. Finally, we concatenate the character's skeleton joints with our predicted proxy joints to form a unified transformation array, and modify Equation 6 as follows:

$$\mathbf{x}_t = \mathcal{LBS}\Big(\mathbf{c}_t, [\mathbf{W}^g, \mathbf{W}^p], [\mathbf{T}_t^s, \mathbf{T}_t^p]\Big), \tag{8}$$

where $\mathbf{W}^p$ are the proxy joint skinning weights. We add both proxy joint and character joint skinning weights $[\mathbf{W}^g, \mathbf{W}^p]$ as optimizable parameters.

Our system can thus be viewed as a hybrid, retaining blendshapes to model high-frequency details such as wrinkles, while macroscopic motion is learned via the spatial rigging hierarchy.



Proxy Joints Deformation Full Deformation

Figure 5: Decomposition of the output of SSILK into proxy-joint-only deformation and the full hybrid proxy-joint and blendshape representation.

This can be seen in Figure 5, where only utilizing the proxy joints gives a rough starting point from where wrinkles are added using blendshapes.

Our system is agnostic to the proxy joint setup process. Off-the-shelf rigging solutions or manual rigging can be used. We also propose an automatic process described in Appendix A.

3.5. Losses

In line with recent self-supervised neural garment simulation methods [BME20b, SOC22, BME22, GBH23, GBB*24], our model is trained without access to ground-truth cloth simulations. Instead, learning is driven entirely by physics-inspired energy objectives applied to the reconstructed garment geometry. We adopt the energy formulation introduced in NCS [BME22] for ease of comparison. The physical loss is defined as the sum of multiple potential energy terms:

$$\mathcal{L}_{physics} = \mathcal{L}_{cloth} + \mathcal{L}_{bending} + \mathcal{L}_{gravity} + \mathcal{L}_{inertia} + \mathcal{L}_{collision}. \quad (9)$$

Here, $\mathcal{L}_{cloth}$ corresponds to the internal cloth energy induced by the material model, $\mathcal{L}_{bending}$ penalizes excessive bending between adjacent mesh faces, and $\mathcal{L}_{gravity}$ represents gravitational potential energy. The collision term $\mathcal{L}_{collision}$ is defined using a signed distance function penalizing interpenetrations between garment and body vertices. Finally, $\mathcal{L}_{inertia}$ approximates inertial effects using an implicit Euler formulation.

3.6. Overall Training Objective and Optimization

For each training rollout, the model produces a sequence of latent states $\mathbf{z}_{0:n}$ which are decoded into world-space garment geometries $\mathbf{x}_{0:n}$. The total loss is accumulated across all timesteps in the rollout:

$$\mathcal{L}_{total} = \sum_{t=0}^{n} \mathcal{L}_{physics}(\mathbf{x}_t). \quad (10)$$

All losses are differentiable with respect to the latent states and decoder parameters, enabling end-to-end optimization through time. Gradients are propagated through the residual integration scheme using backpropagation through time over the rollout window. During training, the latent initializer outputs are detached to prevent trivial solutions and ensure that $\mathcal{P}$ learns to produce quasi static, energy-minimizing garment states. $\mathcal{L}_{inertia}$ is omitted for the first two initialized states, as it requires three states for computation.

This formulation encourages the network to discover latent trajectories that minimize physical energy while remaining amenable to explicit time integration, resulting in stable, temporally consistent simulations over long horizons.

3.7. Training Procedure and Dataset

We train all components of the model jointly in a single end-to-end optimization phase. This includes the latent initializer $\mathcal{P}$, the latent residual stepper $\mathcal{S}$, the neural decoder $\mathcal{D}$, the integration coefficients α and β, as well as the skinning weights $\mathbf{W}^g$ and $\mathbf{W}^p$. Training is performed in a self-supervised manner using physics-inspired loss functions, without access to ground-truth cloth simulation data.

Training Rollouts. During training, we sample temporal windows of fixed length from character animation sequences. In our experiments, each training sample consists of a window of 32 consecutive character poses. The first two poses θ_0 and θ_1 are passed through the latent initializer to produce initial latent states $\mathbf{z}_0$ and $\mathbf{z}_1$. These initial latents are detached from the gradient to ensure that the initializer learns to produce convergent quasi-static garment states. To improve robustness and prevent overfitting to deterministic rollouts, we add Gaussian noise sampled from $\mathcal{N}(0, 0.1)$ to the latent states during training. Subsequent latent states $\mathbf{z}_{2:n}$ are predicted autoregressively using the latent residual stepper. The full latent rollout $\mathbf{z}_{0:n}$ is decoded into world-space garment vertex positions $\mathbf{x}_{0:n}$, which are used to compute the physics-based and regularization losses described in Section 3.5. This training strategy exposes the model to its own predictions over multiple timesteps, mitigating error accumulation and enabling stable long-horizon rollouts at inference time.

Dataset. Our method does not require ground-truth garment simulations and can be trained using only character animation data together with a rest garment mesh and skinning information. Each training sequence consists of animated character poses parameterized by skeletal joint transformations. Garments are provided in rest configuration along with skinning weights transferring deformation from the character skeleton to the garment. Because the loss formulation is fully self-supervised, the dataset does not constrain the simulator to reproduce specific garment trajectories. Instead, the network learns physically plausible garment behavior by minimizing the total energy of the reconstructed garment geometry over time.

Data Augmentation. To improve robustness and generalization, particularly to rare or out-of-distribution character poses, we apply strong data augmentation to the character animations during training. The goal of this augmentation is not to synthesize plausible character motion, but rather to expand the coverage of pose space encountered during optimization. Specifically, for each training window, we sample a random global root rotation and apply it consistently to all frames in the sequence. This transformation can produce configurations that are physically unrealistic, such as inverted, flipped, or airborne poses. While such augmentations would

be problematic if the objective were to learn human motion dynamics, our focus is instead on garment dynamics conditioned on pose. In this context, increasing the diversity of pose configurations improves the robustness of the model. By exposing the network to atypical and low-probability configurations, the latent integrator is encouraged to learn stable garment behavior across a broader range of inputs. In practice, this results in improved generalization to out-of-distribution motions, including challenging cases such as flips and cartwheels.

4. Results

We evaluate our method using a consistent and controlled experimental setup. We focus our comparison on NCS [BME22], as it employs loss functions closely aligned with ours, enabling a meaningful analysis of temporal behavior and garment dynamics. For further breadth of coverage, we additionally include a comparison against SNUG [SOC22] in Appendix B. All methods are retrained on the CMU motion dataset using an 80/20 train–validation split. From the validation set, we additionally select clips exhibiting high inertial effects and character flips to assess performance under challenging dynamic conditions. To further evaluate generalization, we report results on a subset of the DanceDB dataset from AMASS [MGT*19]. Experiments are conducted across a range of garment types, including a cape, a T-shirt from NCS [BME22], a fringed skirt we designed, a hooded dress, pants, and long-sleeve shirt from Grigorev et al. [GBH23]. Models are retrained for each individual garment as pose-based neural garment simulators including ours and NCS are locked to a fixed garment topology due to the decoder formulation.

Like prior self-supervised neural cloth simulators, our method has no ground truth target simulation. Therefore, it is difficult to ascertain whether the cloth deformation generated is close to optimal. Comparisons against off-the-shelf cloth simulators are challenging, as different simulators rely on different energy formulations, material models, and discretizations, producing visually distinct yet physically valid outcomes. To address this in a controlled setting, we design a custom cloth simulator operating on the same energy formulation as SSILK and NCS, allowing us to evaluate the capacity of our model to replicate the behavior described by the loss function.

4.1. Qualitative Comparison

To highlight the strengths of our method, we qualitatively compare a challenging cartwheel sequence from the CMU dataset against NCS across several garments. As shown in Figure 6, our method is able to better handle extreme poses compared to NCS, achieving more plausible garment states. We notice a trend with NCS where looser garments perform worse, failing to learn proper dynamics and instead snapping into various garment configurations. This effect can be seen in the supplementary video. However, even for typically tight character garments such as the T-shirt, our method is better capable of handling extreme poses such as the upside down character. We attribute this mainly to our latent integrator formulation. Furthermore, we run our custom cloth simulator on the same cartwheel sequence for the cape garment and present the results in Figure 7. While we note that the simulator dynamics are not fully captured by our method, our results are visually closer to the simulator behavior compared to NCS.

Table 1: Validation set comparison of SSILK vs. NCS, evaluating L_{total} and collision error (ϵ_{col}) as a percentage of colliding asset vertices on various datasets. Lowest value is best (in bold). NCS* refers to NCS trained with our data augmentation strategy.

		CMU Valid		DanceDB		CMU Inertia		CMU Flips	
Asset	Method	L_{tot}	ϵ_{col}	L_{tot}	ϵ_{col}	L_{tot}	ϵ_{col}	L_{tot}	ϵ_{col}
Cape	SSILK	**1.7252**	**0.58%**	1.7835	**0.78%**	1.7700	**0.51%**	**1.1124**	0.55%
	NCS	1.7372	0.79%	**1.7816**	1.00%	**1.7595**	0.83%	1.4740	0.93%
	NCS*	1.8183	0.87%	1.8549	0.99%	1.8374	0.95%	1.5286	**0.48%**
T-Shirt	SSILK	**5.2646**	**1.60%**	**5.3979**	1.77%	**5.3979**	**1.46%**	**3.4863**	1.19%
	NCS	6.9831	1.86%	6.3837	**1.75%**	7.1457	1.86%	4.7760	1.27%
	NCS*	6.5763	1.90%	6.1283	1.79%	6.7300	1.81%	4.5494	**1.13%**
Skirt Fringe	SSILK	**0.7030**	**0.34%**	**0.7243**	**0.34%**	**0.7296**	**0.37%**	**0.4969**	**0.99%**
	NCS	0.7122	0.39%	0.7377	0.76%	0.7467	0.46%	0.7046	1.43%
	NCS*	0.7979	0.81%	0.8093	0.59%	0.8234	0.82%	0.8034	1.80%
Hooded Dress	SSILK	**7.8222**	**0.48%**	**8.1418**	**0.69%**	8.0499	**0.48%**	**5.3570**	**0.38%**
	NCS	7.9418	0.79%	8.2434	0.85%	8.0845	0.87%	6.4317	0.51%
	NCS*	7.9582	0.74%	8.2348	0.72%	**8.0378**	0.75%	6.4700	0.53%
Pants	SSILK	**3.5043**	0.51%	**3.8121**	**1.25%**	**3.7987**	0.72%	**3.0584**	**1.00%**
	NCS	3.5439	0.53%	3.9402	1.56%	3.8196	**0.53%**	3.8268	1.37%
	NCS*	3.7558	**0.50%**	4.0842	1.26%	3.9610	**0.53%**	4.0543	1.33%
Long Sleeve	SSILK	**5.1621**	**1.02%**	**5.5063**	**0.99%**	**5.2518**	**0.81%**	**3.4801**	**0.45%**
	NCS	5.5032	2.36%	5.7658	2.30%	5.5005	2.22%	4.0782	2.05%
	NCS*	5.5157	2.29%	5.7733	2.32%	5.5038	2.05%	4.0174	1.64%

We also qualitatively compare against a version of our model with only blendshapes and a version that uses only proxy joints. As seen in Figure 8, exclusively modeling using proxy bones results in overly smooth garment deformations. The difference between utilizing proxy joints and utilizing many blendshapes is subtle - but manifests in slightly more dynamic garment deformations and the ability for the garment to be more decoupled from the skeletal kinematics.

4.2. Quantitative Analysis

We compare our model with respect to NCS on the four mentioned datasets. For fairness, we also retrain a version of NCS augmented with our data-augmentation strategy. Each validation dataset is split into non-overlapping sequences of 512 frames at 30 frames per second, or approximately 17 seconds of motion. We report our findings in Table 1.

We observe that our method outperforms NCS on $\mathcal{L}_{total}$ across the majority of garments and datasets. On a case-by-case basis, NCS is able to occasionally achieve superior performance. We also observe that for the cape and the skirt examples, adding our data augmentation strategy to NCS significantly decreases their performance. We believe this is attributable to NCS struggling with loose garments that exhibit a high amount of inertia, which is compounded by the shifted skeletal configurations generated by our data augmentation. This inertia manifests in instabilities in the training and failure of the model to learn garment dynamics.

We likewise perform a quantitative comparison of our method, NCS, and our simulator, evaluating on the cape garment for the inertia and flips datasets, reporting the results in Table 2. For the flips

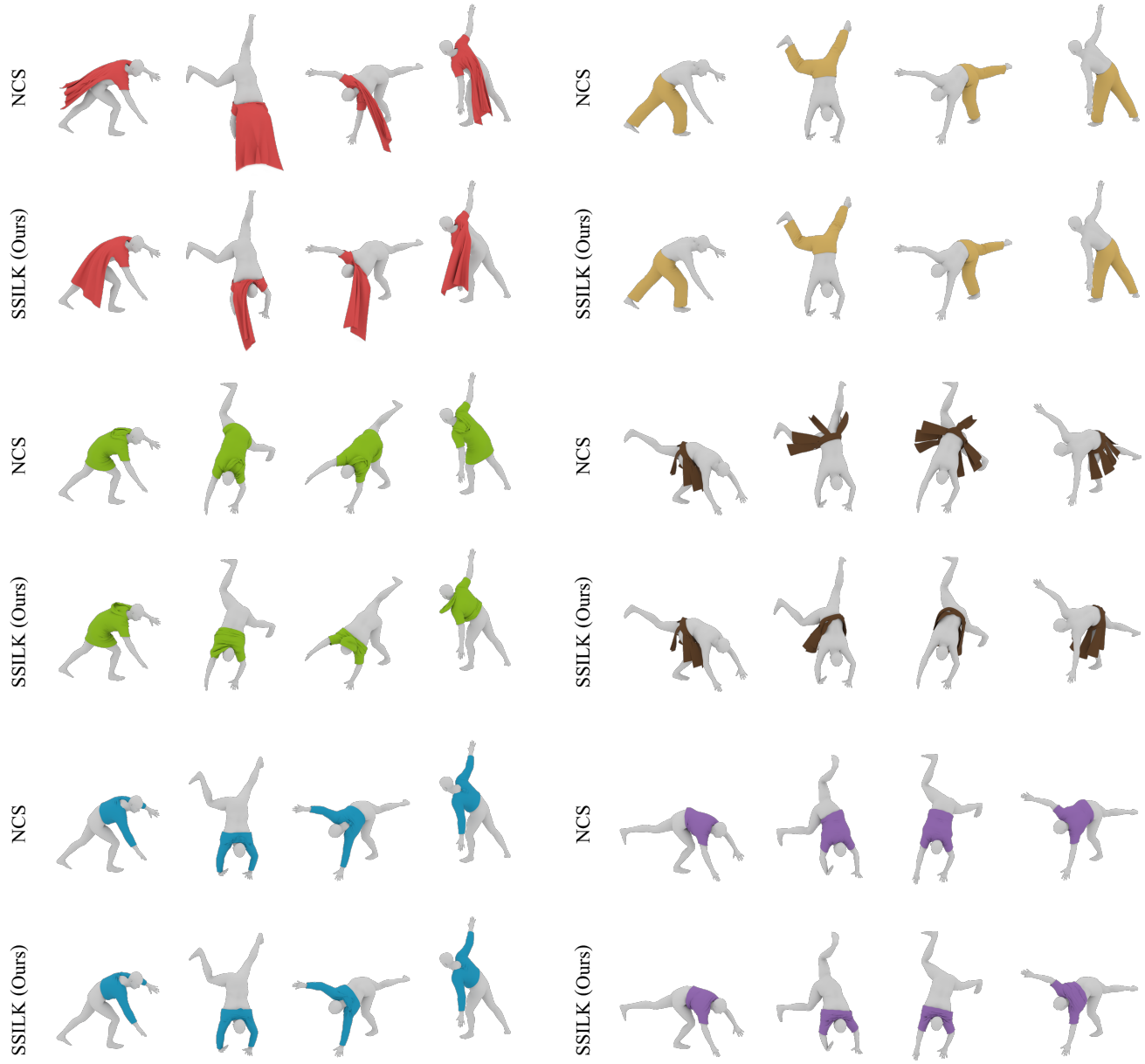


Figure 6: NCS vs SSILK for various garments with a character performing a cartwheel motion.

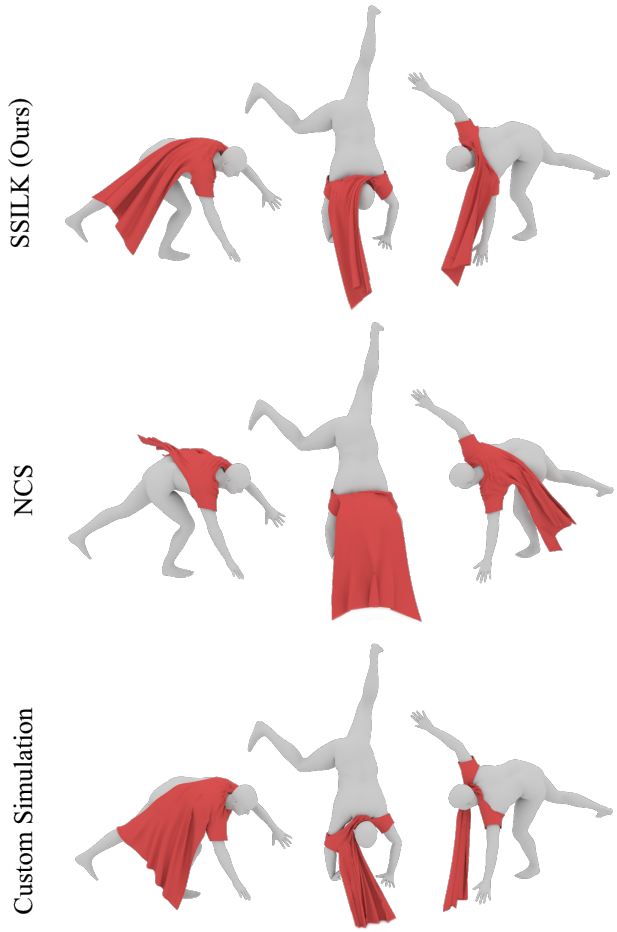


Figure 7: Comparison between SSILK , NCS, and our custom optimizer.

Table 2: Cape validation: SSILK vs. NCS family vs. our simulator. Lowest value is best (in bold). NCS* refers to NCS trained with our data augmentation strategy.

	CMU Inertia		CMU Flips	
Method	L_{tot}	ϵ_{col}	L_{tot}	ϵ_{col}
SSILK	1.7700	0.51%	1.1124	0.55%
NCS	1.7595	0.83%	1.4740	0.93%
NCS*	1.8374	0.95%	1.5286	0.48%
Simulator	**1.6867**	**0.21%**	**0.9664**	**0.38%**

dataset, our method reaches a loss value closer to the ground truth simulator. For the inertia dataset, NCS is slightly lower than ours, but our method has significantly less collisions. We attribute this superior performance on datasets containing flips to a combination of our data augmentation and our hybrid proxy joint decoder.

In-order to evaluate the stability of our model on long animation sequences, we manually create a looped handstand sequence, repeated for 10,000 frames. Then, we infer our method for the skirt garment, evaluating the per-frame loss. Similarly to evaluations done for stability analysis in prior work by Grigorev et al. [GBH23], we plot out the per-frame energy of our system in Figure 10. We notice the cyclical nature of the energy, with the absence of any form of drift in energy for the 10,000-frame duration. We therefore conclude that our method is stable for long animation sequences.

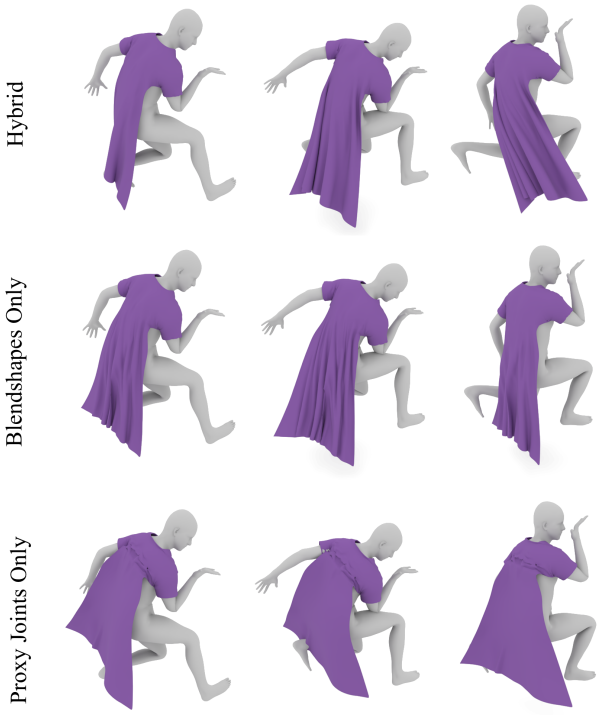


Figure 8: Comparison between using only blendshapes, our hybrid model, or only using proxy bones, for the cape asset.

4.3. Performance

Device	Metric	NCS (ms/frame)	SSILK (ms/frame)
CPU	Full Pipeline	5.34 ± 4.01	7.26 ± 3.15
CPU	Model only	3.64 ± 3.99	2.83 ± 1.75
CPU	Model + FK	3.64 ± 3.99	2.85 ± 1.75
CPU	Blendshapes + Skinning	1.69 ± 0.37	4.41 ± 2.62
GPU	Full Pipeline	4.38 ± 0.18	5.38 ± 0.52
GPU	Model only	2.19 ± 0.07	2.75 ± 0.20
GPU	Model + FK	2.20 ± 0.07	2.77 ± 0.20
GPU	Blendshapes + Skinning	2.18 ± 0.16	2.61 ± 0.48

Table 3: Single-frame runtime comparison between NCS and SSILK (ms/frame), evaluated in PyTorch on both the CPU and GPU. Evaluations were run on an Intel Xeon W-2255 at 3.7 GHz with a Nvidia GeForce RTX 2070 SUPER.

We compare our method to NCS in terms of inference cost on the skirt fringe garment. We explicitly separate running the neural network modules from running the full pipeline. Pose-based neural

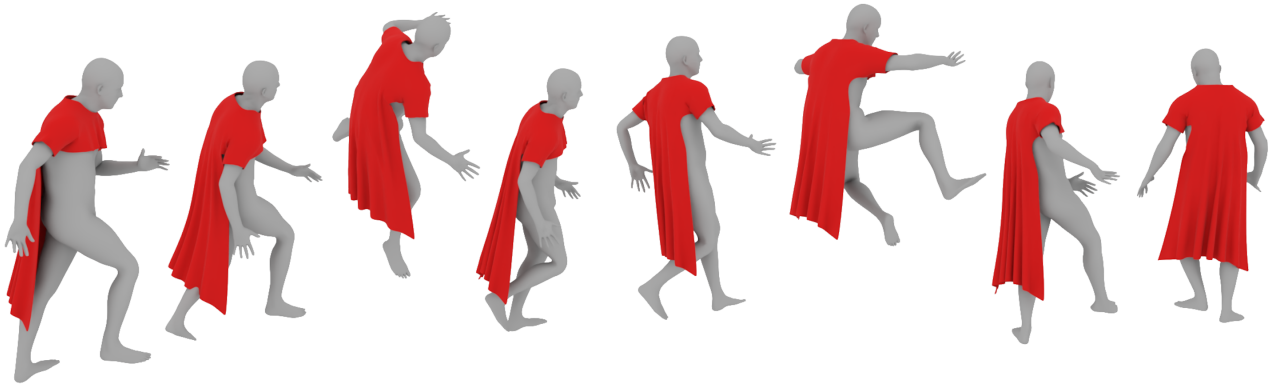


Figure 9: Frames extracted from a 340-frame animation generated using SSILK. The animation showcases a running and jumping character wearing a cape.

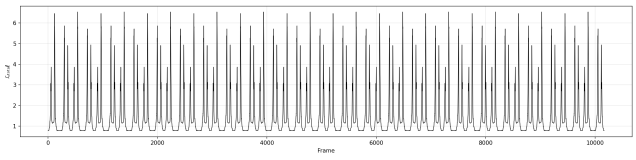


Figure 10: Plotted out energy for a looped handstand motion over 10,000 frames.

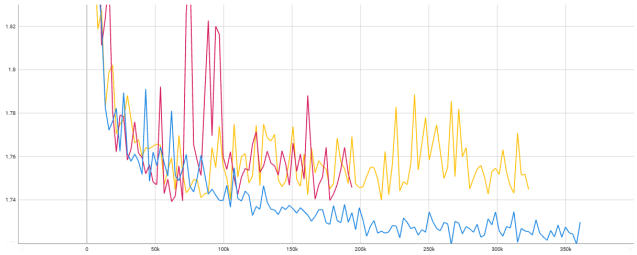


Figure 11: Ablation study of the integration scheme, visualizing validation loss curves on the CMU dataset. Amber represents the $\mathcal{S}_{direct}$, magenta represents $\mathcal{S}_{direct-residual}$, while blue is our method.

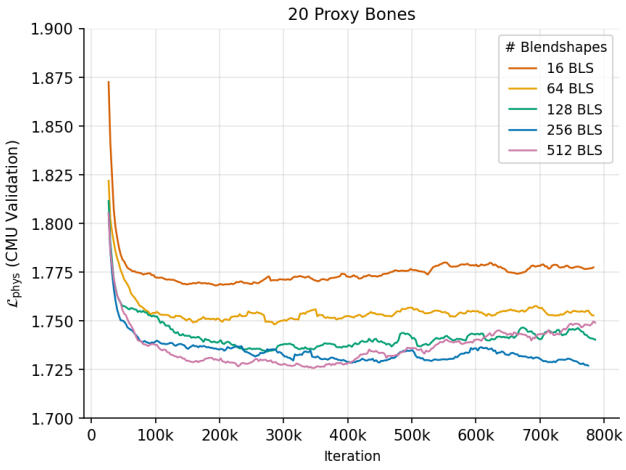


Figure 12: Ablation study of varying the blendshape count for our model, utilizing 20 proxy joints for the cape garment.

cloth simulators are implemented in such a manner to be compatible with existing game pipelines, typically running the predictive model on the CPU (up to Equation 7), and doing the blendshape resolution (Equation 5) and the skinning (Equation 8) on the GPU using CUDA. As such, it is important to compare performance of the models with this consideration. Table 3 demonstrates that our model is roughly on par with NCS, with slightly more expensive performance that we attribute to our proxy joint logic.

4.4. Ablation

In order to evaluate architecture choices of our method, we conduct an ablation study on several key components. We evaluate our model performance on the cape garment on several validation datasets.

Integration. We examine the effectiveness of our integration scheme by comparing against several baselines. We implement a naive baseline where we directly predict $\mathbf{z}_t$:

$$\mathbf{z}_t = \mathcal{S}_{\text{direct}}(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \theta_t^d). \tag{11}$$

Additionally, we implement a similar residual approach:

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \mathcal{S}_{\text{direct-residual}}(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \theta_t^d). \tag{12}$$

As can be seen in Figure 11, the chosen Verlet-inspired architecture outperforms the baselines. Likewise, we noticed that the training for this architecture is significantly more stable compared to the baselines.

Table 4: Ablation study evaluating the impact of the integration methods on the L_{total} validation loss. Lowest value is best (in bold).

Method	CMU Valid	DanceDB	CMU Inertia	CMU Flips
SSILK	**1.7252**	1.7835	**1.7700**	**1.1124**
SSILK Regularized	1.7278	**1.7746**	1.7760	1.1327

Table 5: Hyperparameter sweep evaluating the impact of individual configurations on L_{total} validation loss.

Parameter	Setting	CMU Valid	DanceDB	CMU Inertia	CMU Flips
Batch Size	8	1.7346	1.7787	1.7781	**1.0843**
	32	**1.7252**	1.7835	**1.7700**	1.1124
	64	1.7289	**1.7698**	1.7743	1.0955
	256	1.7500	1.7751	1.8008	1.1284
Window Size	8*		diverged		
	16	1.7394	1.7743	1.7884	**1.0946**
	32	**1.7252**	1.7835	**1.7700**	1.1124
	64	1.7260	**1.7691**	1.7837	1.1281
Latent Size	16	1.7551	1.8030	1.8045	1.1457
	64	1.7374	**1.7678**	1.7952	1.1362
	128	**1.7252**	1.7835	**1.7700**	**1.1124**
	512	1.7309	1.7803	1.7808	1.1239

Regularization. As an additional experiment, we evaluate applying regularization to the subspace. Our latent integration formulation is inspired by Holden et al. [HDDN19], who applied it to a linear subspace computed via PCA on cloth simulation data. This opens up the question on whether the linear Verlet step is meaningful when applied to curved subspaces. We examine utilizing an additional latent regularization loss inspired by Sharp et al. [SRJ*23], defined as:

$$\mathcal{L}_{reg} = \lambda \left(\log \frac{\|\mathbf{x} - \mathbf{x}'\|_M}{\sigma \|\mathbf{z} - \mathbf{z}'\|} \right)^2, \tag{13}$$

where $\mathbf{z}$ and $\mathbf{z}'$ are latent samples drawn from the same training batch, and $\mathbf{x}$ and $\mathbf{x}'$ are their corresponding decoded garment geometries. The norm $\|\cdot\|_M$ denotes a mass-weighted Euclidean distance in vertex space, while λ and σ are hyperparameters controlling regularization strength. As can be seen in Table 4, this formulation does not seem to provide a meaningful advantage on our validation sets. We propose two possible explanations for this: relatively flat local neighborhoods and the capacity of neural model $\mathcal{S}$ to overcome linear projections off the valid cloth manifold. We thus omit this term from our main model formulation.

Blendshape Count. We conduct an ablation study on how varying the number of blendshapes affects the model performance, reporting the results of our model trained on the cape garment with varying blendshape counts in Figure 12. As can be seen, broadly speaking, increasing the blendshape count improves performance up to a certain point.

4.5. Hyperparameters

Physics-based neural network training is notoriously sensitive to hyperparameters. For instance, as noted by the authors of NCS [BME22], their model fails to learn dynamics for batch sizes of less than 512. We conduct a hyperparameter sweep for the cape garment over several important hyperparameters: batch size, window size, and latent size. We define a base model with a batch size of 32, window size of 32, and latent size of 128, from which we vary individual hyperparameters. We show the results of this sweep in Table 5, training for 250,000 iterations. We note that our model training is fairly stable with a variety of hyperparameters, with the exception of training with a small window size of 8 that causes divergence. We conclude that utilizing batch sizes of 32 with window sizes of 32 and a latent size of 128 is sufficient for our model.

5. Conclusion

We have presented **SSILK** - a self-supervised pose based neural garment simulator. Our method improves upon the limitations of existing pose-based neural garment simulators by employing an explicit latent-space integration strategy, coupled with a hybrid proxy joint and blendshape decoder. We conduct end-to-end training on our latent initializer, stepper module, learnable integration parameter vectors, decoder, blendshapes, and proxy joint skinning weights. Our training strategy includes data augmentation which allows us to capture a broader variety of character animations such as a cartwheel. We compare our method to another pose-based neural simulator, NCS, for a variety of garments, showcasing the benefits of our Verlet-inspired formulation. We furthermore conduct several ablation studies demonstrating benefits in our method compared to several baselines. We also conduct a hyperparameter sweep to identify suitable model parameters and evaluate the stability of our training procedure.

Our method has some limitations. A first limitation is the lack of support for body parameters, that has been shown to be possible with pose-based methods in prior methods [SOC22, CCP24]. Our method also does not support garment multi-layering, which has been explored previously in the pose-based paradigm [BME20b]. As our method limits the garment proxy joints to rotational joint chains, we believe the exploration of other groups of transformations such as translation, scaling, and shear to be a promising avenue to improve the expressiveness of the proxy joints. Likewise, while we propose an automated process for setting up the initial proxy joint configuration - an interesting avenue of exploration is to optimize the proxy joint configuration during training. Various features of graph-based neural network approaches could also be explored, including self-collision [GBB*24] and the reliance on a fixed garment topology per network, which is perhaps the most cumbersome limitation of current pose-based methods.

6. Acknowledgments

We would like to thank Ubisoft Divertissements Inc. for support and funding for this work.

References

[BJ05] BARBIČ J., JAMES D. L.: Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM transactions on graphics (TOG) 24*, 3 (2005), 982–990. 2

[BME20a] BERTICHE H., MADADI M., ESCALERA S.: Cloth3d: clothed 3d humans. In *European Conference on Computer Vision* (2020), Springer, pp. 344–359. 2

[BME20b] BERTICHE H., MADADI M., ESCALERA S.: Pbns: physically based neural simulator for unsupervised garment pose space deformation. *arXiv preprint arXiv:2012.11310* (2020). 2, 3, 6, 11

[BME22] BERTICHE H., MADADI M., ESCALERA S.: Neural cloth simulation. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–14. 2, 3, 4, 6, 7, 11

[BMTE21] BERTICHE H., MADADI M., TYLSON E., ESCALERA S.: Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 5471–5480. 2, 3

[BW23] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 767–778. 2

[CCP24] CHEN R., CHEN L., PARASHAR S.: Gaps: Geometry-aware, physics-based, self-supervised neural garment draping. In *2024 International Conference on 3D Vision (3DV)* (2024), IEEE, pp. 116–125. 2, 3, 11

[CLYY24] CHEN A. H., LIU Z., YANG Y., YUKSEL C.: Vertex block descent. *ACM Transactions on Graphics (TOG) 43*, 4 (2024), 1–16. 2

[GBB*24] GRIGOREV A., BECHERINI G., BLACK M., HILLIGES O., THOMASZEWSKI B.: Contourcraft: Learning to resolve intersections in neural multi-garment simulations. In *ACM SIGGRAPH 2024 conference papers* (2024), pp. 1–10. 2, 3, 6, 11

[GBH23] GRIGOREV A., BLACK M. J., HILLIGES O.: Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2023), pp. 16965–16974. 2, 3, 6, 7, 9

[GDY25] GILES C., DIAZ E., YUKSEL C.: Augmented vertex block descent. *ACM Transactions on Graphics (TOG) 44*, 4 (2025), 1–12. 2

[HDDN19] HOLDEN D., DUONG B. C., DATTA S., NOWROUZEZAHRAI D.: Subspace neural physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2019), pp. 1–12. 2, 4, 11

[HTC*14] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R. W., COLE F., MEYER M., DEROSE T., GROSS M.: Subspace clothing simulation using adaptive bases. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–9. 2

[JOG*24] JIN Y., OMENS D., GENG Z., TERAN J., KUMAR A., TASHIRO K., FEDKIW R.: A neural-network-based approach for loose-fitting clothing. *arXiv preprint arXiv:2404.16896* (2024). 2

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 165–172. 1, 2

[LLL*25] LI Y., LIN G. W.-C., LARIONOV E., BOZIC A., ROBLE D., KAVAN L., COROS S., THOMASZEWSKI B., STUYCK T., CHEN H.-Y.: Self-supervised learning of latent space dynamics. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 8*, 4 (2025), 1–18. 3

[LWK24] LIAO Z., WANG S., KOMURA T.: Senc: Handling self-collision in neural cloth simulation. In *European Conference on Computer Vision* (2024), Springer, pp. 385–402. 2, 3

[MGT*19] MAHMOOD N., GHORBANI N., TROJE N. F., PONS-MOLL G., BLACK M. J.: Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 5442–5451. 7

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118. 2

[MMC16] MACKLIN M., MÜLLER M., CHENTANEZ N.: Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), pp. 49–54. 2

[MTLT88] MAGNENAT-THALMANN N., LAPERRIRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88* (1988), Citeseer. 1, 3

[PLPM20] PATEL C., LIAO Z., PONS-MOLL G.: Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 7365–7375. 2

[PMJ*22] PAN X., MAI J., JIANG X., TANG D., LI J., SHAO T., ZHOU K., JIN X., MANOCHA D.: Predicting loose-fitting garment deformations using bone-driven motion networks. In *ACM SIGGRAPH 2022 conference proceedings* (2022), pp. 1–10. 2, 5

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (1989), pp. 215–222. 2

[SCG*26] SRIVASTAVA A., CHEN H.-Y., GOLDADE R., HERHOLZ P., JIANG Z., LIN G. W.-C., YANG L., SARAFIANOS N., STUYCK T., LARIONOV E.: Physkin: Physics-based bone-driven neural garment simulation. *arXiv preprint arXiv:2603.27013* (2026). 3

[SOC19] SANTESTEBAN I., OTADUY M. A., CASAS D.: Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 355–366. 2

[SOC22] SANTESTEBAN I., OTADUY M. A., CASAS D.: Snug: Self-supervised neural dynamic garments. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 8140–8150. 2, 3, 6, 7, 11, 13

[SRJ*23] SHARP N., ROMERO C., JACOBSON A., VOUGA E., KRY P., LEVIN D. I., SOLOMON J.: Data-free learning of reduced-order kinematics. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023), pp. 1–9. 11

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 205–214. 2

[ZBL*19] ZHOU Y., BARNES C., LU J., YANG J., LI H.: On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 5745–5753. 5

Appendix A: Proxy Joint Setup

We present an automated rigging and skinning process which can be used to setup the proxy bones. First, we subdivide the garment vertices into three classes: pinned, tight, and loose (see Figure 4). Pinned vertices strictly follow the output of linear blend skinning from the character skeleton and have no corrective blendshapes. Tight vertices are also skinned exclusively to the character skeleton but permit corrective blendshapes. Finally, loose vertices are deformed using the proxy joint kinematics coupled with blendshapes.

On the loose vertex surface, we evenly sample k proxy joints (the green markers in Figure 4). The number of proxy joints k is an adjustable parameter that will vary based on the asset type.

To define the joint hierarchy, we segment loose vertices into connected clusters. For each cluster, we calculate the geodesically nearest tight or pinned vertex, and determine the character joint with the highest skinning weight associated with that vertex. This corresponds to one of the hip joint for the example in Figure 4. For each cluster, we then attach the j-nearest joints to this character joint, where j is also a parameter, that is set to 4 in the particular example of Figure 4. We conduct a geodesic breadth-first-search to attach the remaining proxy joints to their nearest neighbors, forming a proxy kinematic chain.

Finally, we set up initial skinning weights for the newly created proxy joints, scaling the skinning weights with a distance-based falloff. Notably, only loose vertices are skinned to our proxy joints, whereas tight vertices remain exclusively skinned to the character skeleton.

Appendix B: SNUG Comparison

To expand the comparison set of our model, we re-implement SNUG [SOC22] based on their paper and publically available inference code, with our material model and parameters, and showcase the results in Figure 13 for the hooded dress garment. As can be seen, SNUG exhibits the same limitations as NCS when modelling loose garments in extreme poses. In comparison, our method is able to more accurately model these extreme garment configurations.

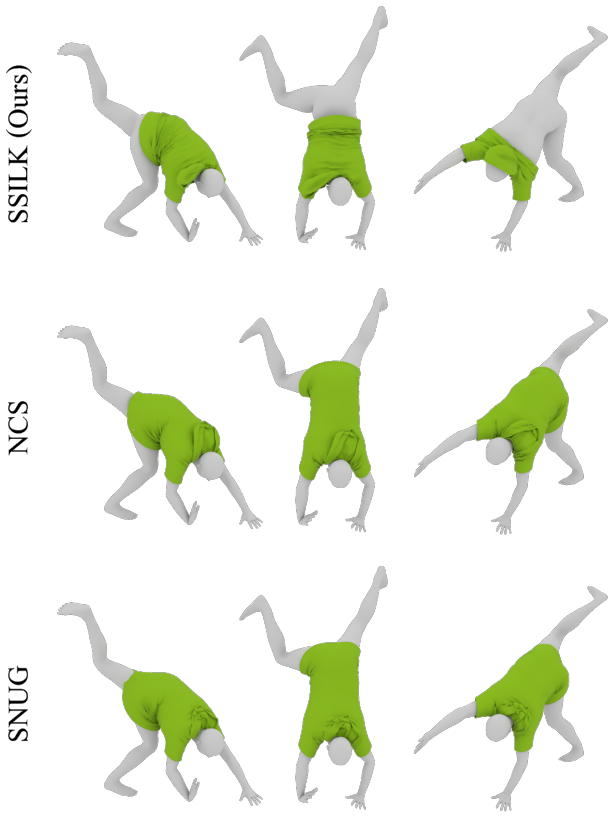


Figure 13: Comparison of our method against NCS and SNUG.