Proxy Asset Generation for Cloth Simulation in Games

ZHONGTIAN ZHENG, LightSpeed Studios, China
TONGTONG WANG, LightSpeed Studios, China
QIJIA FENG, LightSpeed Studios, USA
ZHERONG PAN, LightSpeed Studios, USA
XIFENG GAO, LightSpeed Studios, USA
KUI WU, LightSpeed Studios, USA

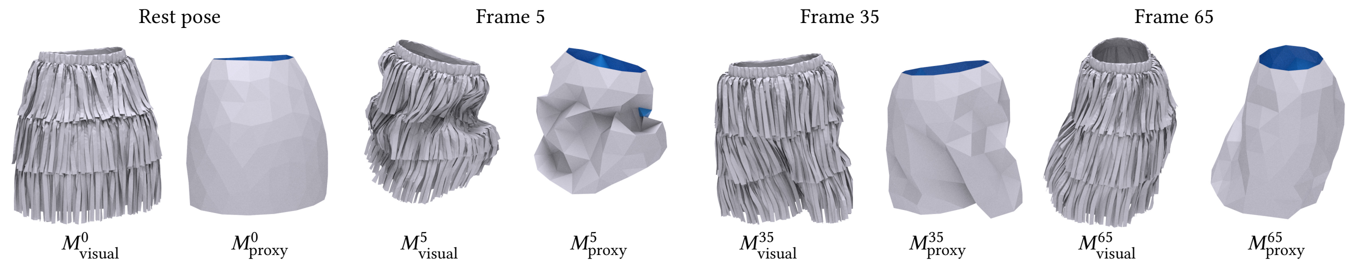


Fig. 1. *Given the input visual mesh M^0_{visual} for the skirt, containing complex geometry and multiple layers, we propose a pipeline to automatically create a single-layer low-poly mesh M^0_{proxy} with an extremely small number of vertices (128 vertices in this case). Additionally, we optimize the skinning weights by differential skinning with several well-designed loss functions, so we get plausible skinned visual mesh, M^*_{visual}, driven by the simulated proxy mesh, M^*_{proxy}, at frame 5, 35, and 65 of a dancing motion sequence. Please see the supplemental video for the full animation.*

Simulating high-resolution cloth poses computational challenges in real-time applications. In the gaming industry, the proxy mesh technique offers an alternative, simulating a simplified low-resolution cloth geometry, *proxy mesh*. This proxy mesh's dynamics drive the detailed high-resolution geometry, *visual mesh*, through Linear Blended Skinning (LBS). However, generating a suitable proxy mesh with appropriate skinning weights from a given visual mesh is non-trivial, often requiring skilled artists several days for fine-tuning. This paper presents an automatic pipeline to convert an ill-conditioned high-resolution visual mesh into a single-layer low-poly proxy mesh. Given that the input visual mesh may not be simulation-ready, our approach then simulates the proxy mesh based on specific use scenarios and optimizes the skinning weights, relying on differential skinning with several well-designed loss functions to ensure the skinned visual mesh appears plausible in the final simulation. We have tested our method on various challenging cloth models, demonstrating its robustness and effectiveness.

CCS Concepts: • **Computing methodologies → Mesh geometry models**.

Additional Key Words and Phrases: Cloth Simulation, Proxy Mesh, Differentiable Optimization

Authors' addresses: Zhongtian Zheng, zhongtzheng@tencent.com, LightSpeed Studios, Shenzhen, Guangzhou, China; Tongtong Wang, tongttwang@tencent.com, LightSpeed Studios, Shenzhen, Guangzhou, China; Qijia Feng, victorfeng@global.tencent.com, LightSpeed Studios, Irvine, CA, USA; Zherong Pan, zrpan@global.tencent.com, LightSpeed Studios, Seattle, WA, USA; Xifeng Gao, xifgao@global.tencent.com, LightSpeed Studios, Seattle, WA, USA; Kui Wu, kwwu@global.tencent.com, LightSpeed Studios, Los Angeles, CA, USA.

0730-0301/2024/7-ART73 $15.00
https://doi.org/10.1145/3658177

ACM Reference Format:
Zhongtian Zheng, Tongtong Wang, Qijia Feng, Zherong Pan, Xifeng Gao, and Kui Wu. 2024. Proxy Asset Generation for Cloth Simulation in Games. *ACM Trans. Graph.* 43, 4, Article 73 (July 2024), 12 pages. https://doi.org/10.1145/3658177

1 INTRODUCTION

Cloth simulation adds unprecedented visual realism to immersive games at the cost of an extremely high computational burden. Game developers must balance visual fidelity and performance to ensure the game runs smoothly on various hardware configurations. To this end, a popular technique is using a proxy mesh, which is a low-res version of the cloth geometry used for fast physical simulation in real-time. Given a simulated configuration of the proxy mesh, its deformation, encoded as vertex positions and normals, is then transferred to the corresponding vertices of a high-res visual mesh through Linear Blended Skinning (LBS). The transferred visual mesh is then rendered to create the final visual representation.

Although the proxy mesh technique can effectively balance visual fidelity and performance, creating a suitable proxy mesh with appropriate skinning weights is a non-trivial, labor-intensive task. In this work, our goal is to design a computer-assisted procedure to automate the design of proxy mesh. There are two major challenges to this end: First, although mesh simplification techniques can be used to create low-res meshes from high-res visual meshes, these methods fail to create proxy meshes because our input visual meshes are ill-conditioned. Indeed, our visual meshes contain intricate folds, wrinkles, disconnected components, layered structures, and non-manifold surfaces. Unfortunately, for plausible simulation results, the output proxy mesh must have elements with uniformly high quality without any ill conditions. Second, for the simulation to achieve real-time performance, the proxy mesh must be extremely simplified. Assigning vertex weights for the proxy mesh poses another significant challenge. An ideal skinning weight should maintain an overall similarity between the low- and high-res meshes and preserve the fine details, which demands a unique blend of technical expertise, artistic judgment, and an in-depth understanding of how characters move and deform. Artists are tasked with meticulously adjusting these weights, repeatedly testing animations, and making refinements until they achieve the desired level of mesh quality and deformation fidelity.

To the best of our knowledge, no existing method simultaneously achieves the aforementioned goals, leaving artists to endure days of trial and error to fine-tune proxy meshes. In this paper, we introduce an automatic pipeline to convert ill-conditioned visual cloth into an extremely simplified, high-quality, single-layer, low-poly proxy mesh with corresponding skinning weights. Our method comprises two stages: proxy mesh generation and optimization of skinning weights. In the first stage, we extract an iso-surface from an unsigned distance field around the visual mesh. This double-layered mesh, containing an excess of vertices, is overly complex for use as a proxy mesh. To address this, a guide graph is constructed through ray casting, and an Integer Linear Programming (ILP) formulation is employed to preserve only a single side around thin shell structures. Subsequently, Voronoi clustering uniformly simplifies the mesh into the proxy mesh with a specified vertex count. Given that the input visual mesh may not be simulation-ready, our approach proceeds to simulate the proxy mesh based on specific use scenarios. We then optimize the skinning weights for a simplified LBS model, using differential skinning supported by several well-designed loss functions to ensure that the visual mesh appears plausible in the final simulation. Inspired by [Thiery and Eisemann 2018], we introduce an As-Rigid-As-Possible (ARAP) term to preserve the shape of the visual mesh, a collision term to prevent self-collision, and an attachment loss to connect close but disconnected components. We have thoroughly tested our approach using various cloth models from real-world game projects. Through the evaluation, we emphasize the remarkable effectiveness and efficiency of our pipeline, highlighting its potential for significant advancements in the field. In summary, our contributions encompass the following:

- An automatic pipeline for generating extremely low-poly proxy meshes from ill-conditioned high-res visual meshes.
- A skinning weight optimization pipeline through differentiable skinning with several well-designed losses.
- A comprehensive evaluation of the effectiveness and efficiency of the proposed approach.

2 RELATED WORK

This section briefly reviews previous works on mesh simplification methods, cloth simulation, and skin deformation.

Mesh Simplification. Mesh simplification techniques [Khan et al. 2022] can be classified as local and global methods. Local methods selectively eliminate elements, including vertices [Schroeder et al. 1992] and edges [Hoppe et al. 1993], based on specific conditions or the minimization of certain metrics, such as the Quadric Error Metrics (QEM) [Garland and Heckbert 1997], visibility [Zhang and Turk 2002], and appearance-preserving [Cohen et al. 1998]. Animated meshes can also be simplified through local methods [Kircher and Garland 2005; Landreneau and Schaefer 2009], but outputs of which usually have a bad mesh quality that prevents them from being used by the physical-based simulator. Inspired by [Cohen-Steiner et al. 2004] and [Li and Nan 2021], today's researchers have mainly focused on global re-meshing techniques for superior robustness, e.g., instant meshes [Jakob et al. 2015]. Overall, generating extremely low-poly mesh still poses a major challenge to traditional mesh reduction techniques. To this end, researchers turn to voxelization-based remeshing techniques [Calderon and Boubekeur 2017; Chen et al. 2023; Mehra et al. 2009]. In the game industry, these low-poly meshing techniques have been used to generate LOD mesh [Gao et al. 2022] and occluder [Wu et al. 2022]. Unfortunately, none of these works generate low-poly, simulation-ready meshes along with skinning weights. From another perspective, mesh simplification can be a reconstruction problem. Indeed, many point cloud reconstruction algorithms [Bernardini et al. 1999; Chen et al. 2022b; Chen and Zhang 2021; Long et al. 2023; Zhou et al. 2023] and mesh repair methods [Chen et al. 2023; Zheng et al. 2023] simultaneously eliminate various defects in the mesh, but these methods focus on reconstructing fine details instead of simplifying the mesh. On the other hand, the introduction of unsigned distance fields (UDFs) [Chen et al. 2022a; Chibane et al. 2020; Liu et al. 2023; Ren et al. 2023; Wang et al. 2022] enable a new learning-based representation to depict both closed and open surfaces, but these methods often necessitate a specified dataset for optimal performance, which is unavailable in our problem domain. Some methods also aim at direct surface extraction from UDFs [Guillard et al. 2022; Hou et al. 2023; Zhou et al. 2023]. However, these techniques require heuristic initial guesses to guide the extraction of open surfaces, leading to a lack of robustness.

Cloth Simulation. Physics-based cloth simulation has been a popular topic in the graphics community for decades. The implicit Euler integration is used to simulate cloth robustly with large time steps [Baraff and Witkin 1998; Terzopoulos et al. 1987] while introducing excessive numerical damping. Liu et al. [2013] treat the implicit Euler integration as an energy minimization problem for the mass-spring cloth. With a similar idea, Projective Dynamics adds support for various hyperelastic materials [Bouaziz et al. 2014] and frictional contacts [Ly et al. 2020]. Recent efforts are focus on efficient cloth simulation on GPU, such as parallel Position-Based Dynamics (PBD) [Macklin et al. 2016; Müller et al. 2007a], Chebyshev acceleration [Wang 2015], parallel randomized Gauss-Seidel [Fratarcangeli et al. 2016] geometric multigrid scheme [Wang et al. 2018], and Galerkin multigrid scheme [Xian et al. 2019], While GPU-based cloth simulation has made significant strides in terms of speed, enabling the real-time simulation of hundreds of thousands of faces, it still falls short of meeting the demands of video games, which only have a few milliseconds budget for simulation. Recently, Zhang et al. [2022] have provided an efficient simulation method to preview quasi-static states of cloth for high-fidelity garment design. In contrast, our proposed technique focuses on the creation of low-poly cloth assets for real-time applications.

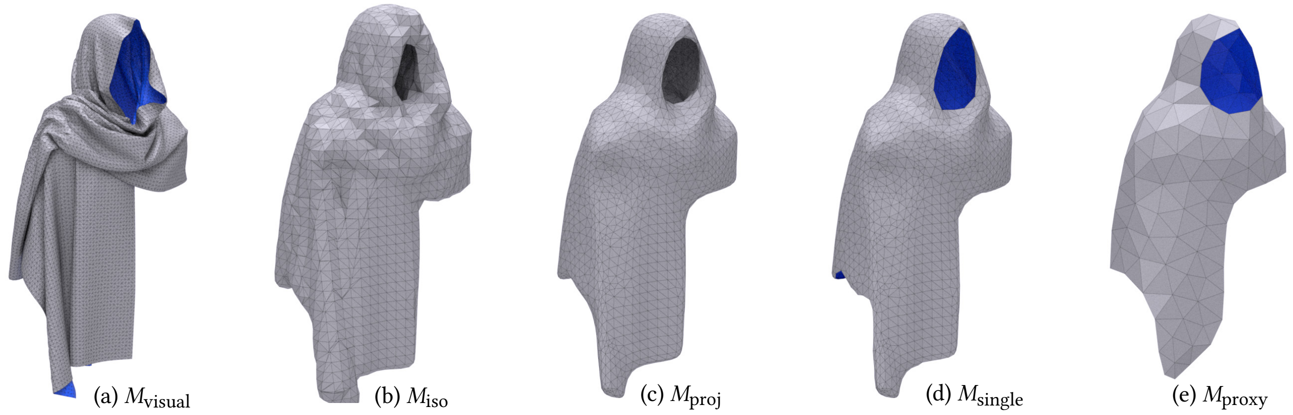


Fig. 2. ***Proxy mesh generation pipeline:*** *Given the input visual mesh M_{visual} (a), we use marching cubes to extract an iso-surface from UDF, denoted as M_{iso} (b). Subsequently, we project M_{iso} onto M_{visual} to enhance alignment, leading to M_{proj} (c). We then extract M_{single} by solving an ILP, which is refined into a single-layer proxy mesh M_{proxy} (e). The front and back sides of the face are highlighted in grey and blue.*

Skinning Deformation. Skinning deformation is a crucial technique in character animation for video games and 3D animation to balance fidelity and performance. In this paradigm, the 3D visual mesh is driven by an underlying skeleton based on given vertex weights. Although these vertex weights are assigned by artists manually, researchers recently automated this process through geometric [Bang and Lee 2018; Dionne and de Lasa 2013; Jacobson et al. 2011; Kavan et al. 2007; Kavan and Žára 2005; Thiery and Eisemann 2018], data-driven [James and Twigg 2005; Le and Deng 2014; Loper et al. 2015], and physics-based [Kim et al. 2017; Mukai and Kuriyama 2016; Si et al. 2015] methods. Besides their applications in character animations, the skinning technique has also been applied to use bones or low-res meshes [Kavan et al. 2011] to drive detailed cloth animations [Feng et al. 2008, 2010]. Recently, NeuroSkinning [Liu et al. 2019] uses a large garment dataset meticulously hand-painted by artists for training and utilizes graph convolution techniques to predict vertex weights. Meanwhile, RigNet [Xu et al. 2020] has introduced a neural rigging solution capable of jointly predicting both the skeletal structures and the corresponding skin weights. In the pre-learning era, classical methods primarily use geometric attributes like geodesic distance and Laplacian energy to determine vertex weights, which is unreliable in producing plausible skinning. Thiery and Eisemann [2018] present a robust LBS weights and skeleton joint optimization method based on ARAP deformations. However, their method does not alter the skeleton topology and relies on a well-conditioned mesh topology to perform the ARAP computations. On the other hand, data-driven and learning-based methods demand a simulation-ready cloth model or a larger dataset painted by artists as a prerequisite, none of which are available in our case.

3 PROXY MESH GENERATION

Our pipeline consists of two main stages. In this section, we detail our first stage, where we generate the proxy mesh, denoted as M_{proxy}, from the visual mesh M_{visual}. In our next section, we propose an optimization-based method to generate the skinning weights. The steps of our first stage are summarized in Fig. 2. In particular, our method first extracts the isosurface M_{iso} from the UDF of M_{visual} (Sec. 3.1). Then, we project M_{iso} onto M_{visual} to enhance alignment, resulting in M_{proj} (Sec. 3.2). After that, we build a guide graph on M_{proj}, on which we solve an ILP to extract the single-side surface M_{single} (Sec. 3.3), which is simplified into the final proxy mesh M_{proxy}.

3.1 Isosurface Extraction

The input M_{visual} may contain non-manifold shapes, layered structures, and disconnected components. To be robust to these artifacts, we follow prior work [Chen et al. 2023] and adopt a voxel-based remeshing. We construct a UDF from M_{visual} with voxel size D/N_v, where N_v is a user-defined voxelization resolution parameter and D is the maximum length of M_{visual}'s bounding box. The isosurface M_{iso} is then extracted using the marching cube algorithm [Lorensen and Cline 1987] with a distance threshold set at one voxel size, i.e. D/N_v, as it is a small iso-value required to extract a watertight iso-surfaces from the UDF to containing M_{visual}.

3.2 Mesh Projection

To further enhance the conformity to M_{visual}, we formulate an optimization problem to obtain a projected mesh M_{proj} that tightly encloses M_{visual}. In particular, we begin with initializing each vertex in M_{proj} as $v^i_{\text{proj}} \leftarrow v^i_{\text{iso}}$. Then, we find the nearest point p^i_{visual} on M_{visual} for each vertex v^i_{proj} and minimize the following energy:

$$\sum_i \left\| v^i_{\text{proj}} - p^i_{\text{visual}} \right\|^2 + \lambda_L \sum_i \left\| v^i_{\text{proj}} - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} v^j_{\text{proj}} \right\|^2, \tag{1}$$

where $\mathcal{N}(i)$ denotes the 1-ring neighboring vertices of v^i_{iso}. The first term guides v^i_{proj} towards p^i_{visual}, while the second term is a Laplacian regularization that encourages smoothness and element uniformity. Through optimization using the vector Adam solver [Ling et al. 2022], M_{proj} is refined to better conform to M_{visual}. Note that we do not enforce self-intersection-free in the optimization, as our proxy generation and simulation pipeline do not mandate a self-intersection-free proxy mesh.

3.3 Single Layer Extraction

While M_{proj} is both topologically and geometrically benign to downstream processing, the watertight nature of the marching cubes algorithm generates excessive surface layers encapsulating open surfaces or thin shell structures. To further simplify the mesh, we propose a novel, graph-cut-like algorithm to extract a single-layer mesh, M_{single}. We show that our modified graph-cut algorithm can be formulated as an ILP, lending itself to efficient off-the-shelf solvers. A typical procedure is illustrated in Fig. 3. We first discuss our method to construct a graph on M_{proj} and then formulate our optimization problem and the solution.

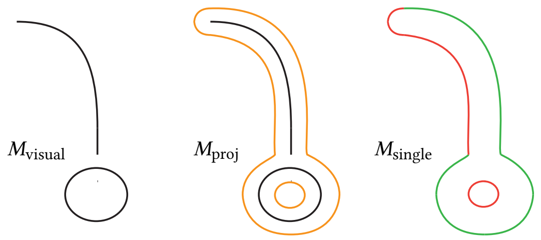


Fig. 3. ***Single layer extraction:*** *from left to right, the input visual mesh M_{viusal}, the projected mesh M_{proj} (orange) containing M_{viusal} tightly, and the single-layer mesh (green) M_{single} extracted via ILP.*

Graph Construction. The key to our approach lies in the extraction of the correspondence relationship between pairs of vertices on the opposite sides of M_{proj}, which encloses thin structures of M_{visual}, so we can remove the vertices from one side. Note that this procedure does not need to be accurate and consistent, as our ILP formulation will clean up the inaccuracy. We define that a pair of two vertices are opposite to each other if:

- Their distance is within $2D/N_v$;
- Their normal is in opposite direction $n^j_{\text{proj}} \cdot n^i_{\text{proj}} < -1 + \epsilon_o$.

Indeed, we define a valid pair of opposite vertices that should be within $2D/N_v$ of each other after mesh projection, as we extract the isosurface with iso-value at D/N_v, and exhibit similar normal directions. Here ϵ_o is some user-defined threshold. Based on these observations, we cast a ray from each vertex v^i_{proj} along its negative normal direction $-n^i_{\text{proj}}$ to find the closest hit point $p^i_{\text{hit}^-}$ on M_{proj} and we define the vertices of the triangle containing $p^i_{\text{hit}^-}$ as opposite vertices, i.e.:

$$\mathcal{V}^i_o \triangleq \left\{ v^j_{\text{proj}} \in T(p^i_{\text{hit}^-}) \middle| \|v^j_{\text{proj}} - v^i_{\text{proj}}\| < 2D/N \;\&\; n^j_{\text{proj}} \cdot n^i_{\text{proj}} < -1 + \epsilon_o \right\}.$$

We then construct the graph as $G \triangleq (\mathcal{V}_{\text{proj}}, \mathcal{E})$, where $\mathcal{V}_{\text{proj}}$ comprises all the vertices. The edge set $\mathcal{E}$ is formed by combining set $\mathcal{E}_{\text{proj}}$, which contains all edges derived from M_{proj}, with an additional supplementary edge set $\mathcal{E}_o$. This supplementary set $\mathcal{E}_o$ is specifically designed to connect each v^i_{proj} with every $v^j_{\text{proj}} \in \mathcal{V}^i_o$. This procedure is illustrated in Fig. 4.

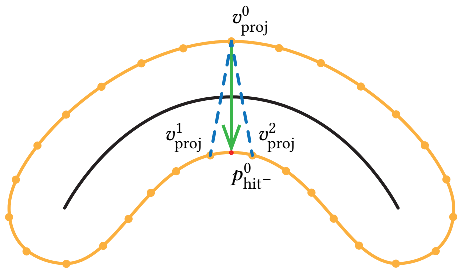


Fig. 4. ***Opposite vertex detection:*** *For each vertex, say v^0_{proj}, we shot a ray along the negative normal direction (green) to find the closest hit point p^0_{hit}. The triangle vertices containing p^0_{hit} is then treated as potential opposite vertices if they satisfy our two conditions. In this example, we define: $\mathcal{V}^0_o = \{v^1_{proj}, v^2_{proj}\}$. We then insert one edge (dashed blue) into our graph, connecting each pair of opposite vertices.*

Minimization Problem. Based on the graph G, we solve a graph-cut-like minimization problem for a consistent set of labels $l^i \in \{0,1\}$ for each vertex v^i_{proj}. $l^i = 1$ implies v^i_{proj} is on the extracted single layer of the proxy mesh and vice versa. Our optimization takes the following form:

$$\underset{l^i \in \{0,1\}}{\text{argmin}} \quad \lambda_s E_s + \lambda_o E_o, \tag{2}$$

with $\lambda_\bullet$ being the corresponding weights, where E_s enforces the smoothness of label assignments, while E_o encourages that only one label is selected between each pair of opposite vertices. Following the standard formulation of graph-cut, our smoothness energy is formulated as:

$$E_s \triangleq \sum_{ij \in \mathcal{E}_{\text{proj}}} w^{ij}_s \mathbb{I}[l^i \neq l^j], \tag{3}$$

where $\mathbb{I}$ is the indicator function: $\mathbb{I}[0] = 0$, $\mathbb{I}[1] = 1$ and w^{ij}_s is the cost for edge ij being non-smooth. We define our weights as $w^{ij}_s \triangleq 1 - (\max(|\kappa_i|, |\kappa_j|)/\kappa)^4$ with κ_i being the approximate mean curvature around v^i_{proj}, computed by locally fitting a quadric function [Gatzke and Grimm 2006] and $\kappa \triangleq \max_{v^j_{\text{proj}} \in \mathcal{V}^j_{\text{proj}}} |\kappa_j|$ is the maximum absolute curvature for normalization. As a result, edges with lower curvature values are assigned larger weights, making them more likely to be consistent, while edges with higher curvature values are assigned smaller weights, which are potential layer boundaries.

Our second term E_o intends to preserve the opposite labels between a pair of vertices in $\mathcal{E}_o$, so as favoring the preservation of only one side and discourages simultaneous preservation or removal of both sides, defined as:

$$E_o \triangleq \sum_{ij \in \mathcal{E}_o} \mathbb{I}[l^i = l^j] + \lambda_{\text{bias}} \mathbb{I}[d^i \neq d^j \;\&\; (l^i - l^j) \neq \text{sign}(d^i - d^j)]$$

$$d^i \triangleq \begin{cases} \|v^i_{\text{proj}} - p^i_{\text{hit}^+}\| & \text{if } p^i_{\text{hit}^+} \text{ exists} \\ \infty & \text{otherwise} \end{cases},$$

where d^i is the distance from v^i_{proj} to its first hit point $p^i_{\text{hit}^+}$ along its positive normal n^i_{proj}. Our first term simply encourages that exactly one of the opposite vertices be selected, while the second term biases the solution towards keeping the outer layer around M_{visual} instead of the inner layer when $d^i \neq d^j$, as the ray from the outer layer along the normal direction is more likely to hit nothing, thereby $d^i = \infty$.

ILP Formulation. Our objective function resembles that of a graph-cut segmentation problem, with each objective function term taking the following $E_\bullet^{ij}(l^i, l^j)$. Regretfully, we cannot utilize their polynomial-time algorithm due to the violation of the regularity condition: $E_o^{ij}(0,0) + E_o^{ij}(1,1) \nleq E_o^{ij}(1,0) + E_o^{ij}(0,1)$ [Kolmogorov and Zabin 2004]. Therefore, we propose to formulate our problem as a standard ILP as discussed by Komodakis and Tziritas [2007]. Specifically, we introduce additional continuous variables: $l_{00,11,10,01}^{ij}$ and transform the objective function term into:

$$l_{00}^{ij} E_\bullet^{ij}(0,0) + l_{11}^{ij} E_\bullet^{ij}(1,1) + l_{10}^{ij} E_\bullet^{ij}(1,0) + l_{01}^{ij} E_\bullet^{ij}(0,1), \tag{4}$$

and introduce the additional constraints:

$$\begin{aligned} l_{00}^{ij} + l_{01}^{ij} &= 1 - l^i, \quad & l_{10}^{ij} + l_{11}^{ij} &= l^i, \\ l_{00}^{ij} + l_{10}^{ij} &= 1 - l^j, \quad & l_{01}^{ij} + l_{11}^{ij} &= l^j. \end{aligned} \tag{5}$$

Although the number of binary variables is large, the above problem can be solved very efficiently in practice using off-the-shelf ILP solvers such as Mosek [ApS 2019].

3.4 Simplification

After obtaining the single layer mesh M_{single}, we further simplified it until the user-defined desired vertex count N_t is reached, to which end we adopt the Voronoi-diagram-based technique [Valette and Chassery 2004]. Specifically, we cluster vertices in M_{single} into a Centroidal Voronoi Diagram with N_t groups. Then, M_{proxy} is extracted from the centroid of neighboring clusters, followed by removing non-manifold faces and splitting non-manifold vertices. This approach outputs our uniformly meshed surface M_{proxy} with the specified number of vertices.

4 SKINNING WEIGHT OPTIMIZATION

Given M_{proxy} generated from the previous stage, our subsequent stage generates LBS weights to create a plausible skinned M_{visual} from the pre-simulated M_{proxy}. Unlike previous data-driven weight generation techniques such as [Liu et al. 2019], where the visual mesh needs to be simulation-ready, our method deals with an input visual mesh that might contain geometric ill-conditions, causing it to be non-simulation-ready. Alternatively, we propose to formulate it as an inverse problem, which could be solved via differentiable programming. Given that M_{proxy} is well-conditioned, aka, manifold, uniformly meshed, and simulation-ready surface, we run cloth simulation on M_{proxy} and sample a dataset of frames. We then utilize differentiable LBS to optimize several loss functions to ensure that the reconstructed frames of animated M_{visual} appear plausible and consistent with M_{proxy}. We use superscript t to denote the frame index of both meshes, and $t = 0$ corresponds to the rest pose.

Standard LBS techniques blend the translation and rotation associated with each bone, but estimating local rotations for each vertex of the proxy mesh in real-time incurs additional computational costs. Therefore, we employ a simplified LBS formulation that considers only translations. In particular, our simplified LBS formulation dictates that the ith vertex of M_{visual} should be reconstructed as:

$$v_{\text{visual}}^{i,t} = v_{\text{visual}}^{i,0} + \sum_{j \in \mathcal{B}(i)} w^{ij} (v_{\text{proxy}}^{j,t} - v_{\text{proxy}}^{j,0}), \tag{6}$$

where $\mathcal{B}(i)$ represents the set of related proxy vertex indices influencing visual vertex i. The weight w^{ij} determines the influence of each vertex on the transformation, which is further assumed to be a convex combination, aka., $w^{ij} \geq 0$ and $\sum_{j \in \mathcal{B}(i)} w^{ij} = 1$. Note that although we utilize a simplified LBS, our weight optimization framework can be extended to standard LBS or Dual Quaternion Skinning (DQS). In such cases, however, the runtime cloth simulator must also be enhanced to model and integrate the governing equations for additional rotational degrees of freedom over time.

Data Preparation. We assume $\mathcal{B}(i)$ is fixed and only optimize w^{ij}. At rest pose, we adopt k-nearest neighbors (kNN) to find the corresponding $\mathcal{B}(i)$ for each $v_{\text{visual}}^{i,0}$. We then simulate M_{proxy} using Position Based Dynamics (PBD) [Müller et al. 2007b] and generate a set of frames, from which we sample N frames at regular intervals to form our dataset. Specifically, we employ two types of springs in our simulation. The first type is used along each edge of M_{proxy} to provide stretch resistance, while the second type is connecting non-shared vertices of each face pair with springs to offer bending resistance. The stiffness parameters for these springs are manually set to match those used in our runtime simulation.

Skinning Weight Optimization. After the generation of frames, we optimize the skinning weights by minimizing the following loss:

$$\operatorname*{argmin}_{w^{ij}} \quad \lambda_r L_r + \lambda_c L_c + \lambda_a L_a \tag{7}$$

$$s.t. \begin{cases} w^{ij} \geq 0 & \forall w^{ij} \\ \sum_{j \in \mathcal{B}(i)} w^{ij} = 1 & \forall \mathcal{B}(i) \end{cases}, \tag{8}$$

with $\lambda_\bullet$ being the corresponding weights, where our first term enforces the smoothness of M_{visual}, which uses an As-$\underline{\text{R}}$igid-As-Possible (ARAP) energy [Sorkine and Alexa 2007] to ensures the approximate non-stretchable material properties of M_{visual}:

$$L_r \triangleq \sum_{t=1}^{N} \sum_i \min_{R_r^{i,t}} \sum_{j \in \mathcal{N}(i)} \left\| (v_{\text{visual}}^{i,t} - v_{\text{visual}}^{j,t}) - R_r^{i,t} (v_{\text{visual}}^{i,0} - v_{\text{visual}}^{j,0}) \right\|^2,$$

where $R_r^{i,t}$ is the local rotation matrix for $v_{\text{visual}}^{i,t}$. We use our second term $\underline{\text{c}}$ollision energy to reduce collision between two cloth layers. To this end, we use the k-nearest neighbors in the initial state of M_{visual} rather than local connectivity:

$$L_c \triangleq \sum_{t=1}^{N} \sum_i \min_{R_c^{i,t}} \sum_{j \in \mathcal{K}(i)} \left\| (v_{\text{visual}}^{i,t} - v_{\text{visual}}^{j,t}) - R_c^{i,t} (v_{\text{visual}}^{i,0} - v_{\text{visual}}^{j,0}) \right\|^2,$$

where $\mathcal{K}(i)$ is the k-nearest neighbors of $v_{\text{visual}}^{i,0}$ and $R_c^{i,t}$ is the rotation matrix. The main difference between L_r and L_c lies in the selection of vertex pairs. In L_r, $\mathcal{N}(i)$ represents the 1-ring neighbors of vertex v_{visual}^i based on the topology of M_{visual} and are used to preserve its local shape. In the second equation, $\mathcal{K}(i)$ identifies the k-nearest neighbors of v_{visual}^i according to the geometry of M_{visual}.

We use $\mathcal{K}(i)$ to prevent penetration among spatial close vertices. Finally, we found that for ill-conditioned visual meshes, geometrically nearby but topologically disconnected vertices are supposed to be distance-preserving. We thus introduce a final attachment energy of the following form:

$$L_a \triangleq \sum_{t=1}^{N} \left\| V_{\text{visual}}^t - (I - K) V_{\text{visual}}^t \right\|^2, \tag{9}$$

where V_{visual}^t is the concatenation of all vertex positions of M_{visual} at frame t and $I - K$ is the weighted average over the k-nearest neighbors defined as:

$$K_{ij} \triangleq \begin{cases} 1 & \text{if } i = j \\ -\frac{(\|v_{\text{visual}}^{i,0} - v_{\text{visual}}^{j,0}\| + \epsilon_z)^{-1}}{\sum_{k \in \mathcal{K}(i)} (\|v_{\text{visual}}^{i,0} - v_{\text{visual}}^{k,0}\| + \epsilon_z)^{-1}} & \text{if } j \in \mathcal{K}(i) \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where the weights are designed such that two vertices are close to each other if they are so at the rest pose. Based on the proximity of vertices in the initial state, the closer the vertices are in the initial state, the greater the attachment strengths should be used in L_a, as indicated by the value of K_{ij}.

To solve the constrained optimization above, we introduce surrogate parameters s^{ij} and re-define: $w^{ij} \triangleq |s^{ij}| / \sum_{j \in \mathcal{B}(i)} |s^{ij}|$. We solve the optimization iteratively using AdamW solver [Loshchilov and Hutter 2017] in PyTorch with the new unconstrained decision variables s^{ij}. During each iteration, we employ singular value decomposition (SVD) to estimate $R_r^{i,t}$ and $R_c^{i,t}$.

5 RESULTS

We implement our pipeline in Python. All experiments are performed on a computer with a 12th Gen Intel(R) Core (TM) i9-12900K 16-core Processor at 3.2 GHz, 256 GB of RAM, and an NVIDIA GeForce 3090 GPU with 24 GB of memory. To generate basic poses, we use PBD in Nvidia PhysX [NVIDIA 2021]. The ILP problem is efficiently solved by off-the-shelf software [ApS 2019].

Parameters Study. The voxelization resolution N_v and the target proxy mesh vertex count N_p play pivotal roles in balancing between preserving details and managing the complexity of the resulting proxy mesh. Smaller values of N_v ensure the preservation of the fundamental shape of M_{visual}, while larger N_v values capture more details and distinctly delineate parts of M_{visual}. On the other hand, smaller N_p values are advantageous for real-time simulation, although extremely low values may struggle to represent intricate structures due to the need for uniform meshing. Conversely, larger N_p values offer a better representation of structures but may introduce runtime computational burdens. The impacts of these parameters are demonstrated in Fig. 5. In our experiments, we set $N_p = 128$ based on rules of thumb in game development, where mobile platforms typically support 256 bones for LBS. Additionally, we set $N_v = 32$ for the remaining experiments.

Dataset. We conducted a series of experiments to evaluate the proposed method using a dataset comprising 100 distinct clothing items as shown in Fig. 6, some of which are from [miHoYo 2020]. All the inputs and our proxy meshes can be found in supplementary

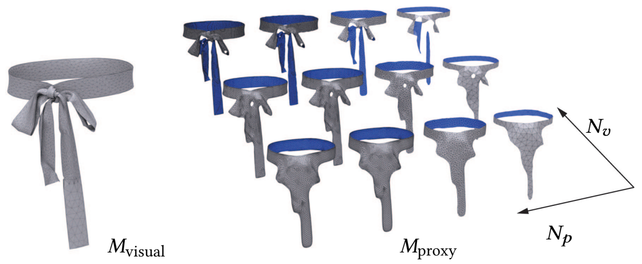


Fig. 5. ***Voxelization resolution N_v and target proxy mesh vertex count N_p:*** *N_v from front to back, 32, 64, and 128, controlling the details and categorization of single layers. N_p from right to left, 128, 1024, 8192, and 65536, controlling the desired size of the uniform mesh.*

materials. The dataset provides a comprehensive representation of various categories and in-game situations. For proxy mesh generation, we conducted a comparative analysis with several state-of-the-art mesh extraction methods, including MeshUDF [Guillard et al. 2022], LevelSetUDF [Zhou et al. 2023], and DCUDF [Hou et al. 2023]. Although these SOTA methods are designed to extract the mesh surface, our final goal is to generate the extremely low-poly mesh as the proxy mesh. Thus, in our comparison, we utilize these methods first to generate the single layer mesh M_{single} and then compare the M_{proxy} generated from them in Sec. 5.1. For MeshUDF, we used the same marching cube resolution 32^3 as ours for mesh generation. For LevelSetUDF, we uniformly sampled 100k points for each mesh as input and used two grid sizes 256^3 and 32^3, the default in LevelSetUDF and ours, respectively. The DCUDF uses a pre-trained multilayer perceptron (MLP) to represent the UDF in the original paper; however, its training requires tens of minutes to hours. In our experiment, we compute the unsigned distance field discretized in a 256^3 grid, as we already have the explicit mesh geometry, and our UDF is more accurate than their trained UDF. We tested both DCUDF default resolution of 256^3 and projection parameter $\lambda_1 = 2000$ and our tuned parameters (resolution at 32^3 and $\lambda_1 = 200$). Subsequently, we employed the same Voronoi clustering method to simplify these meshes with a target vertex count of 128 as ours does. The resulting statistics are collected in Table 1, and we pick eight examples to demonstrate the issue of the existing methods in Fig. 17.

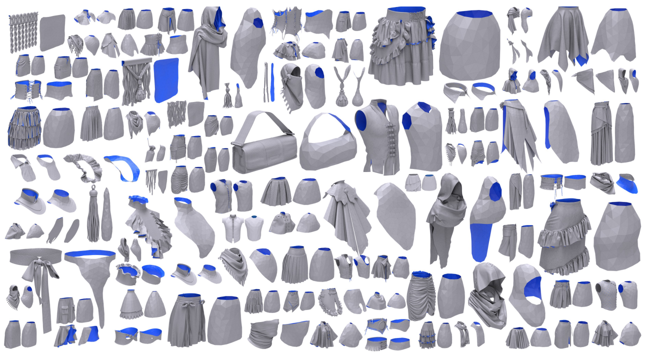


Fig. 6. *A collection of 100 distinct clothing items from games and the low-poly proxy mesh generated by our method.*

Table 1. *Statistics of 100 models in Fig. 6. Comp# and B-loop# denote the number of components and boundary loops, respectively. Note that DCUDF has up to 20 trials as their method relies on a random seed, while only one attempt is needed for each model for other methods.*

	Success	Comp#	B-loop#	HD	LFD	Time
MeshUDF 32^3	**100%**	2.1	3.8	0.28	4.6e3	**0.3**s
LevelSetUDF 32^3	**100%**	15.3	15.7	0.59	1.6e4	424.7s
LevelSetUDF 256^3	**100%**	1.8	2.3	0.63	7.1e3	491.2s
DCUDF 32^3	74%	1.1	**1.4**	0.35	4.7e3	40.8s
DCUDF 256^3	75%	2.4	3.4	0.33	4.5e3	82.1s
Ours 32^3	**100%**	**1.0**	2.0	**0.22**	**3.6e3**	8.4s

5.1 Proxy Mesh Evaluation

HD and LFD. We utilize two qualitative metrics, Hausdorff distance (HD) and light-field distance (LFD), to evaluate the output quality compared to the input. Due to the lack of connectivity in the single-layer mesh, LevelSetUDF often fails to produce a result even close to the input after simplification. MeshUDF, which uses a heuristic strategy to extract the single-layer mesh, cannot handle complex cases well. "Tassel" in Fig. 17 shows MeshUDF could be seriously disrupted when the gradient of the UDF is complex. Employing graph cuts guided by random initial seeds, DCUDF may segment the double layers improperly and even fail to produce a valid segment for complex models within the maximum of 20 trials. In "Scarf", DCUDF cannot well extract the single layer and either miss the left side of the input (32^3) or duplicate it (256^3). In summary, our method outperforms all listed state-of-the-art techniques.

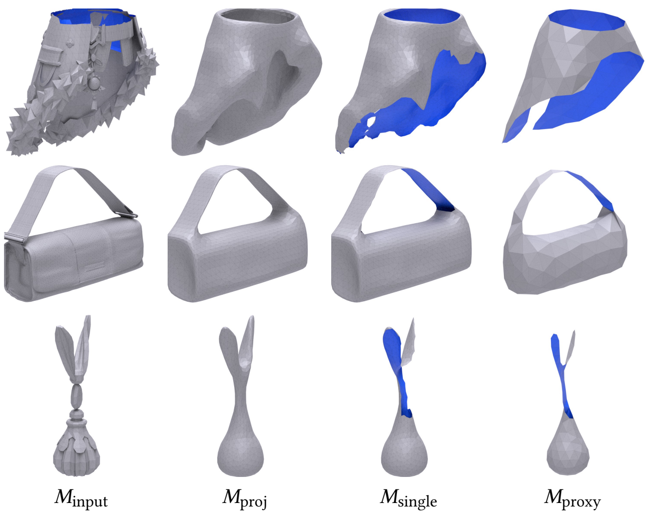


Fig. 7. ***Intermediate results of three examples:*** *Our pipeline is able to handle challenging inputs with a diverse range of topologies.*

Intermediate Results. Fig. 7 illustrates the intermediate results of three examples. The process begins with isosurface extraction and mesh projection, resulting in M_{proj} that tightly encloses M_{input} to ensure a close approximation of the original geometry. Then, an ILP-based method robustly extracts a single-layer structure M_{single} from M_{proj}. This structure, M_{single}, is then simplified to M_{proxy} while preserving essential geometric features.

Component and Boundary Loop Numbers. Generally, we expect the output proxy mesh to be the single component with fewer holes. However, both MeshUDF and LevelSetUDF may produce multiple disconnected components and holes due to the challenges in locating the 0-level set of the input. Additionally, DCUDF might struggle to segment double layers correctly depending on the random initial guess, resulting in unnecessary components and holes. In contrast, our method consistently generates proxy meshes as a single component with an average of 2.0 boundary loops, underscoring the robustness of our approach. It's worth noting that while our results do not exhibit self-intersection, our proxy generation and simulation pipeline do not mandate a self-intersection-free proxy mesh. Detection and resolution of self-intersection are typically disabled in real-time applications, such as games, for performance reasons.

Comparison with DCUDF [Hou et al. 2023]. Our method follows a similar pipeline as DCUDF but differs in solving an ILP problem with a novel opposite energy E_o instead of utilizing graph cut to extract the single layer surface. In DCUDF, faces are randomly initialized with keep/removal labels, and a graph cut problem with smooth energy is solved to optimize the face label, aiming to match the initial label while minimizing smooth loss. However, DCUDF results are significantly influenced by the initial guess, and the smooth energy alone proves inadequate for separating one from two layers in shell structures, a common occurrence in cloth, as demonstrated in Fig. 8. On the contrary, our opposite energy E_o enforces differences in labels between opposite pairs and cannot be incorporated into the graph cut. Additionally, as mentioned in [Hou et al. 2023], DCUDF requires multiple initial guess attempts and does not guarantee finding a valid cut within the given number of trials. Table 1 shows that, given 20 trials, DCUDF achieves only a three-quarters success rate.

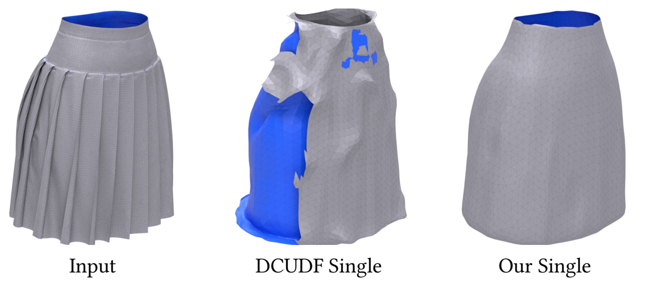


Fig. 8. ***Example of single layer extraction:*** *DCUDF cannot produce valid single layers using graph cut with smooth energy only, while our method can extract the single layer mesh with the same resolution of* 32^3.

Time Usage. As our method solves an ILP formulation, it takes more time to extract the single layer than MeshUDF, which employs a heuristic strategy for single-layer mesh extraction. Fortunately, the average process time of our whole proxy mesh generation pipeline is less than 10 seconds.

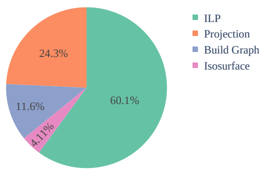


Fig. 9. *Time breakdown*

In particular, ILP optimization and projection occupy almost two-thirds and one-quarter of the computation time, respectively, as shown in Fig. 9.

Mesh Quality. As our resulting proxy meshes need to be used in the simulation, mesh quality is an important metric for predicting the performance of the application. Thanks to the Voronoi-diagram-based simplification, we achieve satisfactory quality in terms of four common qualitative metrics, min face angle, edge ratio, aspect ratio, and radius ratio. The mean and standard deviation values are as follows: (47.9, 6.6), (0.78, 0.11), (0.76, 0.13), and (0.93, 0.085), respectively. We plot the histogram in Fig. 10.

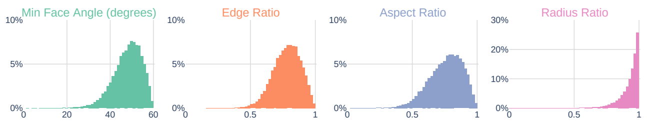


Fig. 10. ***Distribution of qualitative measures for our resulting proxy meshes:*** *min face angle, edge ratio, aspect ratio, and radius ratio [Shewchuk 2002].*

Optional Surface Carving. While our primary goal is to simplify the expression of M_{visual}, it is desirable for artists to have the option of preserving the original separated parts in M_{proxy} so that they can move freely, as illustrated in Fig. 11. To accommodate this preference, we project M_{visual} to M_{single} to create a 2D signed distance field on the surface. Subsequently, we carve out the surface along the isoline at a user-defined threshold ϵ_{carved}, resulting in M_{carved}. This option is left to the artists and is turned off by default.

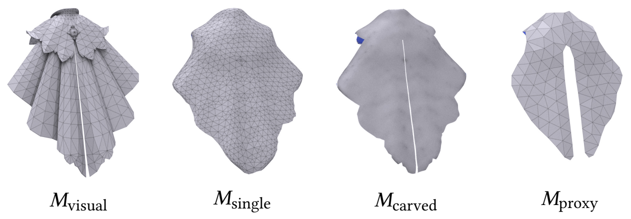
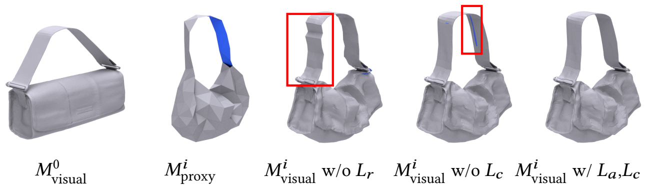


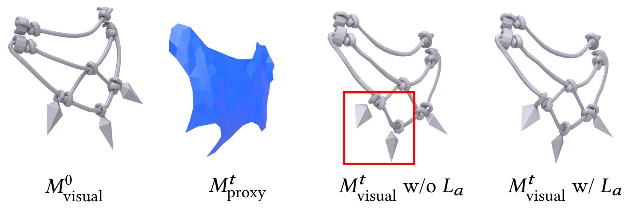
Fig. 11. ***Example of surface carving:*** *After generating M_{single} from M_{visual}, M_{visual} is projected to M_{single} to create a 2D signed distance field on the surface, which is extracted as M_{carved}. Lastly, M_{carved} is simplified as M_{proxy}.*

5.2 Skinning Weights Evaluation

Data Collection & Weights Optimization. We generate 200 frames of simulation data for skinning weight optimization for each proxy mesh. We set the proxy mesh with the highest 5% vertices fixed and the rest vertices are blown by the wind with random direction and strength. The data generation takes an average of 0.54 seconds per proxy mesh. The weights optimization uses AdamW optimizer [Loshchilov and Hutter 2017] with a learning rate of 10^{-3} and a maximum of 200 epochs, which takes 9 minutes on average.

Ablation Studies. We assess the effectiveness of our losses by comparing the visual mesh driven by skinning weights optimized with and without ARAP loss, collision loss, and attachment loss. In Fig. 12, without L_r, the belt lacks smoothness, and without L_c, the belt exhibits self-collision. Conversely, the skinning weights optimized with all the losses produce satisfactory results. In Fig. 13, two components, dart and knot, are affixed to the visual mesh. However, in the absence of attachment loss L_a, these components separate in the skinned visual mesh during significant proxy mesh deformation. On the contrary, the skinning weights optimized with our attachment loss ensure a tight attachment.


M^0_{visual}
M^i_{proxy}
M^i_{visual} w/o L_r
M^i_{visual} w/o L_c
M^i_{visual} w/ L_a,L_c
Fig. 12. ***Ablation study on ARAP and collision loss:*** *Given the input mesh M^0_{visual}, the proxy mesh is simulated at the pose M^i_{proxy}. The absence of our ARAP loss L_r results in a non-smooth belt (highlighted in red). Without our collision loss L_c, the belt exhibits self-collision. The optimized skinning weights with our attachment loss L_r and L_c ensure a smooth belt without collisions even under substantial proxy mesh deformation.*


M^0_{visual}
M^t_{proxy}
M^t_{visual} w/o L_a
M^t_{visual} w/ L_a
Fig. 13. ***Ablation study on attachment loss:*** *Given the input mesh M^0_{visual}, the proxy mesh is simulated at the pose M^t_{proxy}. Without our attachment loss L_a, the dart is detached from the knots (highlighted in red), while the skinning weights optimized with our attachment loss L_a make the dart and knots attached tightly even under large proxy mesh deformation.*

Validation. We first validate our optimized weights by blowing the proxy mesh in a direction not encountered in data collection and scrutinizing the shape of the skinned visual mesh. The supplementary video, featuring 20 randomly selected models, serves as a visual testament to the efficacy of our differential skinning optimization framework. Fig. 15 demonstrates one of them. It is important to note that while our results exhibit generalization to unforeseen scenarios, in production practice, we can pre-simulate most scenarios that happened in games for better data coverage.

As the visual mesh is usually not simulatable, we cannot simulate it and compare the shape directly. Instead, to validate our pipeline, we generate two proxy meshes with 128 vertices and 1024

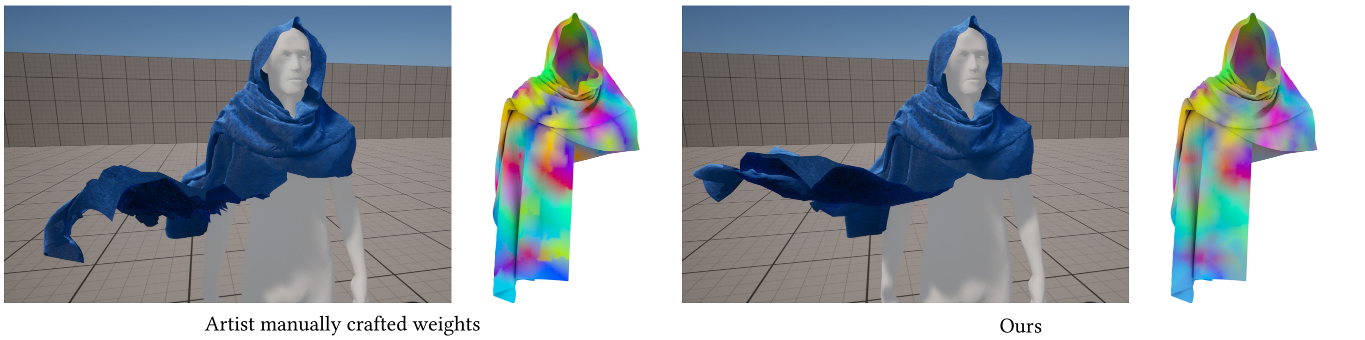


Fig. 14. ***Comparison with the result crafted by artist manually:*** *Left, despite 4 hours of artistic labor, the skinning appears clumpy and spiky; right, our approach yields a smoother and more plausible result in just 14 minutes. The skinning weights are visualized on the visual mesh colored based on the bone index. A lighter color indicates a smaller bone impact.*

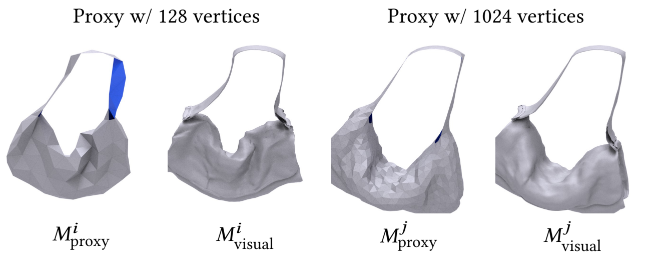


Fig. 15. ***Validation:*** *Given the same visual mesh as input, we choose the simulated proxy mesh with 128 and 1024 vertices under similar deformation at frame i and j as M^i_{proxy} and M^j_{proxy}, respectively. Our optimized skinning weights can provide consistent results, while the one with 1024 vertices has more details.*

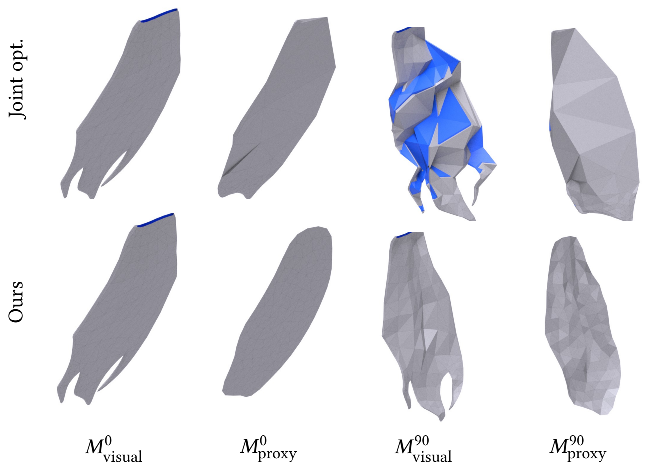


Fig. 16. ***Comparison with joint optimization:*** *Given the same input mesh, joint optimization based on edge decimation (Top) yields ill-posed proxy mesh. While our method (Bottom) produces uniform proxy meshes with natural deformations.*

vertices from the same visual mesh. As shown in Fig. 15, our skinning weights can provide consistent results, while the one with 1024 vertices has more details.

Complex Motion. We further validate our optimized skinning weights under complex motions. In particular, we extract a motion sequence for the waist joint from a dance routine, which includes movements such as spinning, twisting, and shaking. As shown in Fig. 1, our losses ensure that the skinned visual mesh maintains a plausible appearance. Please see the supplementary video for the full animation.

User Study. To further validate the effectiveness of our weight optimization, we conducted an experiment with an artist with over a decade of experience in crafting assets for games and film. Specifically, we provided the proxy mesh of the "Scarf" model, as shown in Fig. 2, and tasked them with manually painting the skinning weights, following industry standards for creating skinned meshes. The weight painting is a process of trial and error. The artist meticulously painted the weights, exported the model to the Unreal Engine, and subjected it to simulation to observe how the visual mesh responded to the skinning. The process is tedious and time-consuming and is neither trivial nor intuitive. Fig. 14 shows the results crafted by the artist for 4 hours. Although the visual mesh roughly follows the proxy mesh movement, the skinning result is still clumpy and spiky, and distant from a production-ready state, while our result is much more plausible and takes only 14 minutes. We also visualize the skinning weights on the visual mesh, which are colored according to the corresponding bone index. Our color map demonstrates a smooth transition, whereas manually adjusted weights exhibit drastic changes in the transitional areas. It is worth noting that we deployed this demo on a Samsung S20 mobile phone with an SD865 CPU. It takes 0.86 ms to simulate the proxy mesh with 128 vertices in one PBD iteration with Unreal Engine Chaos Cloth, which highlights the application scenarios of our framework.

Comparison with Joint Optimization. To validate our pipeline for separately optimizing proxy mesh geometry and skinning weights, we develop a joint optimization framework that iteratively simplifies M_{single} while simultaneously optimizing skinning weights. In each iteration, we first optimize the skinning weights following Eq. 7. Then, we collapse the edge of the proxy mesh using a sum of quadric energies from all visual vertex i influenced by proxy vertex j at each frame t, defined as follows:

$$E^{j,t}(\hat{v}) = \sum_i (\hat{v}^{ij,t})^T Q^{i,t}_{\text{visual}} \hat{v}^{ij,t} + \hat{v}^T Q^{j,t}_{\text{proxy}} \hat{v} \quad s.t.\ j \in \mathcal{B}(i)$$

where $\hat{v} = [v; 1]$ represents the vertex v in homogeneous form and $\hat{v}^{ij,t} = \hat{v}^{i,t}_{\text{visual}} - w^{ij}\left(\hat{v}^{j,t}_{\text{proxy}} - \hat{v}\right)$ denotes the new visual vertex position when moving $v^{j,t}_{\text{proxy}}$ to $\hat{v}$. $Q^{i,t}_{\text{visual}}$ and $Q^{j,t}_{\text{proxy}}$ are the corresponding quadric matrices. The first term accounts for the quadratic energy derived from the visual mesh, as suggested by [Landreneau and Schaefer 2009]. The second term introduces a regularization component for the proxy vertex $v^{j,t}_{\text{proxy}}$ that does not affect the visual mesh. We collapse the edge in M_{proxy} with the lowest energy by contracting two edge vertices to the midpoint with summed skinning weight. Unfortunately, the non-uniform distribution of vertices in the simplified mesh poses challenges in optimizing skinning weights effectively and results in unnatural deformations during the simulation (see Fig. 16).

6 CONCLUSION

We have presented an automatic pipeline for generating extremely low-poly proxy meshes from ill-conditioned high-res visual meshes containing non-manifold shapes, layered structures, and disconnected components. Given the proxy mesh generated from our pipeline, we have proposed a skinning weight optimization framework to ensure the skinned visual mesh appears plausible in the final simulation through differentiable skinning with several novel losses. Lastly, we demonstrate a comprehensive evaluation of the effectiveness and efficiency of our approach in a dataset with 100 cloth models used in the game industry.

Limitations and Future Works. While the Voronoi-diagram-based technique for simplification can produce a uniform proxy mesh, it may struggle to preserve the input boundary while representing intricate structures when N_p is extremely low. In future work, we aim to enhance the method by incorporating local operations such as edge collapse and flip to improve boundary preservation. Additionally, introducing continuous collision detection during projection to prevent self-intersection could be an interesting future direction. While our current approach assumes the cloth with a non-stretchable material model for the simulation, future efforts will explore optimization strategies for accommodating stretchable inputs, using the generalization of ARAP method [Thiery and Eisemann 2018]. Another promising avenue is to leverage our method to generate a series of simplified meshes at different resolution levels, which would be useful for progressive cloth simulation [Zhang et al. 2022]. Additionally, we also plan to explore the joint optimization of the mesh and skinning weights in the future to enhance the quality of the results.

ACKNOWLEDGMENTS

We would like to thank our colleagues from LightSpeed Studios, Fengquan Wang and Dong Li, for their project support.

REFERENCES

MOSEK ApS. 2019. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.* MOSEK.

Seungbae Bang and Sung-Hee Lee. 2018. Spline Interface for Intuitive Skinning Weight Editing. *ACM Trans. Graph.* 37, 5, Article 174 (sep 2018), 14 pages.

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* Association for Computing Machinery, New York, NY, USA, 43–54.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics* 5, 4 (1999), 349–359.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages.

Stéphane Calderon and Tamy Boubekeur. 2017. Bounding Proxies for Shape Approximation. *ACM Trans. Graph.* 36, 4, Article 57 (jul 2017), 13 pages.

Weikai Chen, Cheng Lin, Weiyang Li, and Bo Yang. 2022a. 3PSDF: Three-Pole Signed Distance Function for Learning Surfaces with Arbitrary Topologies. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE Computer Society, Los Alamitos, CA, USA, 18501–18510.

Zhen Chen, Zherong Pan, Kui Wu, Etienne Vouga, and Xifeng Gao. 2023. Robust Low-Poly Meshing for General 3D Models. *ACM Trans. Graph.* 42, 4, Article 119 (jul 2023), 20 pages.

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022b. Neural dual contouring. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

Zhiqin Chen and Hao Zhang. 2021. Neural marching cubes. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

Julian Chibane, Gerard Pons-Moll, et al. 2020. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems* 33 (2020), 21638–21652.

Jonathan Cohen, Marc Olano, and Dinesh Manocha. 1998. Appearance-Preserving Simplification. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* Association for Computing Machinery, New York, NY, USA, 115–122.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational Shape Approximation. *ACM Trans. Graph.* 23, 3 (aug 2004), 905–914.

Olivier Dionne and Martin de Lasa. 2013. Geodesic Voxel Binding for Production Character Meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California) *(SCA '13).* Association for Computing Machinery, New York, NY, USA, 173–180.

Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. 2008. Real-Time Data Driven Deformation Using Kernel Canonical Correlation Analysis. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08).* Association for Computing Machinery, New York, NY, USA, Article 91, 9 pages.

Wei-Wen Feng, Yizhou Yu, and Byung-Uck Kim. 2010. A Deformation Transformer for Real-Time Cloth Animation. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10).* Association for Computing Machinery, New York, NY, USA, Article 108, 9 pages.

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: A Practical Gauss-Seidel Method for Stable Soft Body Dynamics. *ACM Trans. Graph.* 35, 6, Article 214 (Nov. 2016), 9 pages.

Xifeng Gao, Kui Wu, and Zherong Pan. 2022. Low-Poly Mesh Generation for Building Models. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22).* Association for Computing Machinery, New York, NY, USA, Article 3, 9 pages.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97).* ACM Press/Addison-Wesley Publishing Co., USA, 209–216.

Timothy D Gatzke and Cindy M Grimm. 2006. Estimating curvature on triangular meshes. *International journal of shape modeling* 12, 01 (2006), 1–28.

Benoît Guillard, Federico Stella, and Pascal Fua. 2022. MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 576–592.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1993. Mesh Optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93).* Association for Computing Machinery, New York, NY, USA, 19–26.

Fei Hou, Xuhui Chen, Wencheng Wang, Hong Qin, and Ying He. 2023. Robust Zero Level-Set Extraction from Unsigned Distance Fields Based on Double Covering. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–15.

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-Time Deformation. *ACM Trans. Graph.* 30, 4, Article 78 (jul 2011), 8 pages.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Trans. Graph.* 34, 6, Article 189 (nov 2015), 15 pages.

Doug L. James and Christopher D. Twigg. 2005. Skinning Mesh Animations. *ACM Trans. Graph.* 24, 3 (jul 2005), 399–407.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (Seattle, Washington) *(I3D '07)*. Association for Computing Machinery, New York, NY, USA, 39–46.

Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. 2011. Physics-Inspired Upsampling for Cloth Simulation in Games. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 93, 10 pages.

Ladislav Kavan and Jiří Žára. 2005. Spherical Blend Skinning: A Real-Time Deformation of Articulated Models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (Washington, District of Columbia) *(I3D '05)*. Association for Computing Machinery, New York, NY, USA, 9–16.

Dawar Khan, Alexander Plopski, Yuichiro Fujimoto, Masayuki Kanbara, Gul Jabeen, Yongjie Jessica Zhang, Xiaopeng Zhang, and Hirokazu Kato. 2022. Surface Remeshing: A Systematic Literature Review of Methods and Research Directions. *IEEE Transactions on Visualization and Computer Graphics* 28, 3 (2022), 1680–1713.

Meekyoung Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J. Black, and Sung-Hee Lee. 2017. Data-Driven Physics for Human Soft Tissue Animation. *ACM Trans. Graph.* 36, 4, Article 54 (jul 2017), 12 pages.

Scott Kircher and Michael Garland. 2005. Progressive multiresolution meshes for deforming surfaces. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. Association for Computing Machinery, New York, NY, USA, 191–200.

Vladimir Kolmogorov and Ramin Zabin. 2004. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence* 26, 2 (2004), 147–159.

Nikos Komodakis and Georgios Tziritas. 2007. Approximate labeling via graph cuts based on linear programming. *IEEE transactions on pattern analysis and machine intelligence* 29, 8 (2007), 1436–1453.

Eric Landreneau and Scott Schaefer. 2009. Simplification of Articulated Meshes. *Computer Graphics Forum* 28, 2 (2009), 347–353.

Binh Huy Le and Zhigang Deng. 2014. Robust and Accurate Skeletal Rigging from Mesh Sequences. *ACM Trans. Graph.* 33, 4, Article 84 (jul 2014), 10 pages.

Minglei Li and Liangliang Nan. 2021. Feature-preserving 3D mesh simplification for urban buildings. *ISPRS Journal of Photogrammetry and Remote Sensing* 173 (2021), 135–150.

Selena Zihan Ling, Nicholas Sharp, and Alec Jacobson. 2022. VectorAdam for Rotation Equivariant Geometry Optimization. *Advances in Neural Information Processing Systems* 35 (2022), 4111–4122.

Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. 2019. NeuroSkinning: Automatic Skin Binding for Production Characters with Deep Graph Networks. *ACM Trans. Graph.* 38, 4, Article 114 (jul 2019), 12 pages.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-Spring Systems. *ACM Trans. Graph.* 32, 6, Article 214 (Nov. 2013), 7 pages.

Yu-Tao Liu, Li Wang, Jie Yang, Weikai Chen, Xiaoxu Meng, Bo Yang, and Lin Gao. 2023. Neudf: Leaning neural unsigned distance fields with volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 237–247.

Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. 2023. NeuralUDF: Learning Unsigned Distance Fields for Multi-View Reconstruction of Surfaces with Arbitrary Topologies. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 20834–20843.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6, Article 248 (oct 2015), 16 pages.

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 163–169.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* 0, 0 (2017), 0.

Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective Dynamics with Dry Frictional Contact. *ACM Trans. Graph.* 39, 4, Article 57 (July 2020), 8 pages.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. Association for Computing Machinery, New York, NY, USA, 49–54.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J. Mitra. 2009. Abstraction of Man-Made Shapes. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–10.

miHoYo. 2020. HoYoLAB. https://ys.biligame.com/gczj/.

Tomohiko Mukai and Shigeru Kuriyama. 2016. Efficient Dynamic Skinning with Low-Rank Helper Bone Controllers. *ACM Trans. Graph.* 35, 4, Article 36 (jul 2016), 11 pages.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007a. Position Based Dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (April 2007), 109–118.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007b. Position Based Dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (April 2007), 109–118.

NVIDIA. 2021. NVIDIA PhysX. https://developer.nvidia.com/physx-sdk. Version 4.1.

Siyu Ren, Junhui Hou, Xiaodong Chen, Ying He, and Wenping Wang. 2023. Geoudf: Surface reconstruction from 3d point clouds via geometry-guided distance representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE Computer Society, Los Alamitos, CA, USA, 14214–14224.

William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. 1992. Decimation of Triangle Meshes. *SIGGRAPH Comput. Graph.* 26, 2 (jul 1992), 65–70.

Jonathan Richard Shewchuk. 2002. What is a good linear element? interpolation, conditioning, and quality measures.. In *IMR*. n/a, n/a, 115–126.

Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2015. Realistic Biomechanical Simulation and Control of Human Swimming. *ACM Trans. Graph.* 34, 1, Article 10 (dec 2015), 15 pages.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-as-Possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association, Goslar, DEU, 109–116.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 205–214.

Jean-Marc Thiery and Elmar Eisemann. 2018. ARAPLBS: Robust and Efficient Elasticity-Based Optimization of Weights and Skeleton Joints for Linear Blend Skinning with Parametrized Bones. *Computer Graphics Forum* 37, 1 (2018), 32–44.

Sébastien Valette and Jean-Marc Chassery. 2004. Approximated Centroidal Voronoi Diagrams for Uniform Polygonal Mesh Coarsening. *Computer Graphics Forum* 23, 3 (2004), 381–389.

Huamin Wang. 2015. A Chebyshev Semi-Iterative Approach for Accelerating Projective and Position-Based Dynamics. *ACM Trans. Graph.* 34, 6, Article 246 (Oct. 2015), 9 pages.

Li Wang, Weikai Chen, Xiaoxu Meng, Bo Yang, Jintao Li, Lin Gao, et al. 2022. HSDF: Hybrid Sign and Distance Field for Modeling Surfaces with Arbitrary Topologies. *Advances in Neural Information Processing Systems* 35 (2022), 32172–32185.

Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel Multigrid for Nonlinear Cloth Simulation. *Computer Graphics Forum* 37, 7 (2018), 131–141.

Kui Wu, Xu He, Zherong Pan, and Xifeng Gao. 2022. Occluder Generation for Buildings in Digital Games. *Computer Graphics Forum* 41, 7 (2022), 205–214.

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A Scalable Galerkin Multigrid Method for Real-Time Simulation of Deformable Objects. *ACM Trans. Graph.* 38, 6, Article 162 (Nov. 2019), 13 pages.

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. 2020. RigNet: Neural Rigging for Articulated Characters. *ACM Trans. Graph.* 39, 4, Article 58 (aug 2020), 14 pages.

Eugene Zhang and Greg Turk. 2002. Visibility-Guided Simplification. In *Proceedings of the Conference on Visualization '02* (Boston, Massachusetts) *(VIS '02)*. IEEE Computer Society, USA, 267–274.

Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L James, and Danny M Kaufman. 2022. Progressive simulation for cloth quasistatics. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.

Zhongtian Zheng, Xifeng Gao, Zherong Pan, Wei Li, Peng-Shuai Wang, Guoping Wang, and Kui Wu. 2023. Visual-Preserving Mesh Repair. *IEEE Transactions on Visualization and Computer Graphics* 1, 1 (2023), 1–12.

Junsheng Zhou, Baorui Ma, Shujuan Li, Yu-Shen Liu, and Zhizhong Han. 2023. Learning a more continuous zero level set in unsigned distance fields through level set projection. In *Proceedings of the IEEE/CVF international conference on computer vision*. IEEE Computer Society, Los Alamitos, CA, USA, 3181–3192.

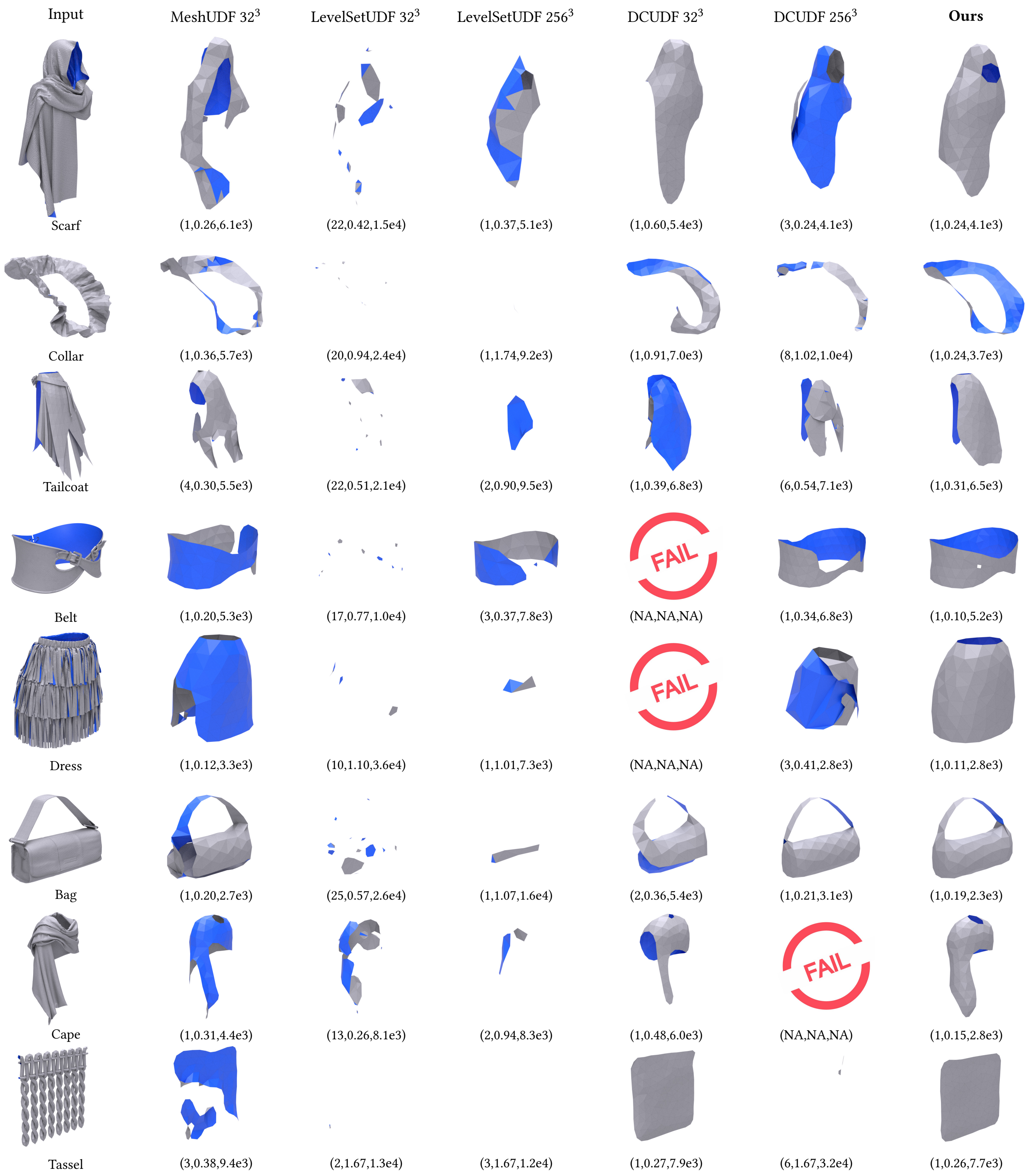


Fig. 17. ***Eight randomly selected garment examples:*** *We choose eight examples, "Scarf", "Collar", "Tailcoat", "Belt", "Dress", "Bag", "Cape", and "Tassel", to the issues of existing methods and our superiority over them.* (●, ●, ●) *indicates boundary loops number, Hausdorff distance, and LFD.*