Bolt: Clothing Virtual Characters at Scale

JONATHAN LEAF, NVIDIA, USA
DAVID SEBASTIAN MINOR, NVIDIA, Canada
GILLES DAVIET, NVIDIA, France
NUTTAPONG CHENTANEZ, NVIDIA, Thailand
GREG KLAR, NVIDIA, New Zealand
ED QUIGLEY, NVIDIA, USA

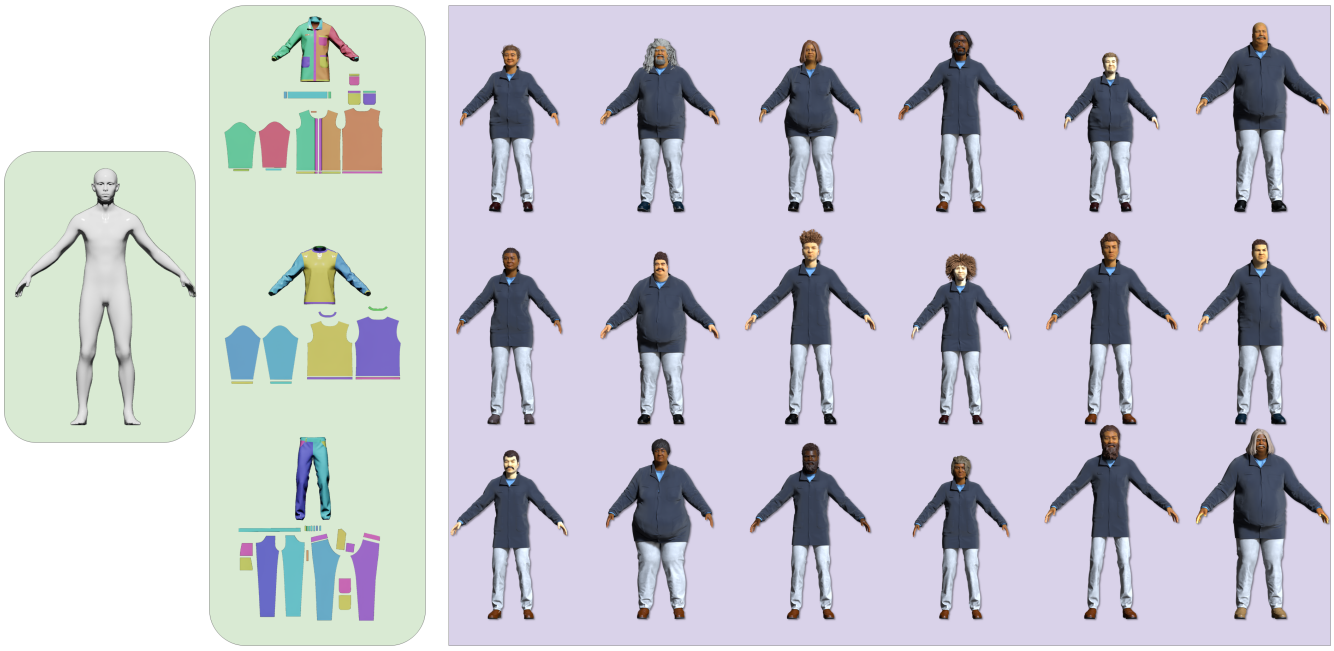


Fig. 1. We present Bolt, an automated method for assembling, refitting, draping, and rigging outfits from a source character to many new characters. On the left is a source character and three garments, each with their corresponding sewing pattern, that fit to its body. Using our method we automatically transfer the 3D garments, adjust the 2D sewing pattern, untangle and drape the outfit on the new character body, and rig the garments for animation. Our method works for a wide range of target body shapes and sizes

Clothing virtual characters is a time-consuming and often manual process. Outfits can be composed of multiple garments, and each garment must be fitted to the unique shape of a character. Since characters can vary widely in size and shape, fitting outfits to many characters is a combinatorially large problem. We present Bolt, a system designed to take outfits originally authored on a source body and fit them to new body shapes via a three stage transfer, drape, and rig process. First, our new garment transfer method transforms each garment's 3D mesh positions to the new character, then optimizes the garment's 2D sewing pattern while maintaining key features of the original seams and boundaries. Second, our system simulates the transferred garments to progressively drape and untangle each garment in the outfit. Finally, the garments are rigged to the new character. This entire process is automatic, making it feasible to clothe characters at scale with no human intervention. Clothed characters are then ready for immediate use in applications such as gaming, animation, synthetic generation, and more.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Collision detection**.

Additional Key Words and Phrases: cloth, simulation, draping, fitting, garments, sewing pattern, panel, pattern pieces, optimization, rigging, character, scale, digital human, outfit, untangling, transfer, proxy

Authors' addresses: Jonathan Leaf, NVIDIA, USA; David Sebastian Minor, NVIDIA, Canada; Gilles Daviet, NVIDIA, France; Nuttapong Chentanez, NVIDIA, Thailand; Greg Klar, NVIDIA, New Zealand; Ed Quigley, NVIDIA, USA.

1 INTRODUCTION

Many industries and applications rely heavily on clothed virtual characters, including video games, animated films, virtual and augmented reality applications, social media, fashion, e-commerce, medicine, industrial applications, and more.

Some applications require large numbers of diverse characters, such as crowds in a video game. Deep learning applications also need lots of high quality training data, but collecting comprehensive images of real world scenarios with clothed humans can be prohibitively costly, difficult, or impossible to collect. For example, computer vision algorithms for autonomous vehicles need to observe a significant number of diverse humans in their dataset to reliably detect the presence of a pedestrian. Some cases are challenging to collect safely, such as children playing near a street or in a garage, but having them represented in the data is crucial for reliably detecting children near an autonomous vehicle. The difficulty of capturing cases like this motivates the synthetic generation of clothed virtual characters who can take the place of real humans in these training sets.

For designing clothed digital humans, sewing pattern-based digital content creation (DCC) tools have emerged as the industry's standard for creating realistic and high-quality 3D garments[1]. The sewing pattern creation process, which is akin to traditional tailoring methods, involves the creation of individual pattern pieces (2D panels) that are then stitched together to form a complete garment. The process begins with the creation of a 2D sewing pattern with stitching and fine details, which is then transformed onto a 3D model and draped by simulating the cloth on the character's body mesh.

Sewing pattern garment design allows for a high degree of customization and precision. However, generating large numbers of clothed characters with DCC tools is a time-consuming challenge even for expert users. The more complex the outfit, the larger the number of garments that must be prepared, assembled, simulated, and adjusted to the character to attain a desirable look. Preparing each garment requires either manually drafting, sewing, simulating and adjusting the garment's pattern from scratch, or by refitting an existing garment that has been fit to a previous character to this new character's body type. Finally, once a new outfit has been fitted to the character, skinning weights must be painted on the clothing either manually, or using external tools.

The large numbers of character body shapes and outfit combinations creates a combinatorial explosion in the number of possible characters that can be generated. Our tool is designed to specifically address this challenge by making the refitting and draping of any outfit onto any new character automatic. For applications that need lots of diverse characters, our approach can save significant amounts of artist time.

Our contributions are as follows:

(1) A novel garment transfer algorithm that transfers the 3D cloth mesh positions from a source character to a target character and then optimizes the 2D sewing pattern to resize and adjust the pattern while preserving style and seams.

(2) An end-to-end pipeline that assembles, refits, drapes, and rigs a multi-garment outfit onto a new character automatically.

[1]https://marvelousdesigner.com, https://clo3d.com, https://optitex.com, and https://browzwear.com

2 RELATED WORK

Sewing Pattern Creation. Recent works have focused on developing improved solutions for creating garments along with their corresponding 2D patterns. Korosteleva and Lee [2022] learn patterns and garments from point clouds. Korosteleva and Sorkine-Hornung [2023] defines a programmatic method for constructing garments, a modular approach that provides high-level primitives. Pietroni et al. [2022] establish how to create pattern pieces from a 3D garment mesh. He et al. [2024] demonstrate a text-to-garment system using a GPT-based architecture. Each of these approaches can reduce garment creation time, though the issues of refitting and draping will still remain, making the data produced from these methods suitable inputs to our system.

Cloth Simulation. To drape garments onto a character, we rely on cloth simulation. Cloth simulation has been adding physically plausible dynamic motion to cloth models since the seminal work of Terzopoulos et al. [1987]. Baraff and Witkin [1998] introduce implicit integration which improves stability. Many works have followed since then, such as new methods for collisions [Bridson et al. 2002; Provot 1997] and bending strain [Gingold et al. 2004b] to name a few.

Many approaches like XPBD [Macklin et al. 2016], Projective Dynamics [Bouaziz et al. 2014], and others [Liu et al. 2017; Tang et al. 2018; Zeller 2005] demonstrate improved performance over prior methods. More recent work reformulates implicit time integration as an energy function [Gast and Schroeder 2015; Martin et al. 2011; Wang and Yang 2016] to be minimized. Additionally, Kim [2020] re-represents classic anisotropic material models within a FEM context.

Garment Refitting. Garment retargeting is an area that has been widely studied, primarily through the lens of capture and virtual try-on [Lee and Jung 2009]. Many works rely on parametric body models [Pons-Moll et al. 2017], while others attempt to learn correspondences for the garment-body interface [Naik et al. 2024].

Shi et al. [2021] learn garment transfer from data, specifically from simulation pairs of source and target bodies. Wang [2018] precisely optimizes sewing patterns, but only over a narrow range of differing body shapes. Grigorev et al. [2024] relies on the SMPL-X body shape system for handling cloth refitting, by modifying garment nodal positions using the same shape vectors as for the body.

de Goes et al. [2020] demonstrate an iterative relax and rebind method for transferring a 3D garment mesh from one body to another. Brouet et al. [2012] uses geometric constraints to transfer a garment and the corresponding pattern, but this method does not include a physically-based drape of the fabric. Chen et al. [2024] uses differentiable simulation to jointly optimize a garment's 2D panels using control cages and 3D positions using differentiable simulation. This combines the objectives of transfer and draping into a single step, and is demonstrated on single-garment outfits.

Our approach refits the 3D positions of a garment, then optimizes the 2D pattern positions based on the transfer. The result of our

cloth refitting process is a simulation ready mesh to be used for draping.

Untangling Cloth. The ability to untangle interpenetrating garment layers is critical for assembling outfits of multiple garments.

Historically, cloth untangling has focused primarily on untangling self-intersections. Baraff et al. [2003], Wicke et al. [2006], Ye et al. [2017] solve this problem by solving garment self-interpenetration globally. Volino and Magnenat-Thalmann [2006] opt for a more local approach, which is faster but comes with no global guarantees.

Industry tools like Marvelous Designer rely on interactive layer-based simulation to untangle garments. Users can specify layer-numbers, and the simulation will order the layers in the collision handling process.

Recent research has focused on untangling multiple garments. Assuming each garment is assumed to have a unique layer number, the objective is to have the garments untangle themselves in the user specified ordering. Buffet et al. [2019] use implicit field operators to untangle multi-layered garments configurations. Lee et al. [2023] use neural networks for untangle layers of clothing. Grigorev et al. [2024] use graph neural networks to resolve collisions between layers.

Our approach takes a sequential approach to solve this problem, inspired by practitioners in the VFX community. We simulate each garment one at a time, progressively draping them on the body. Other recent works like [Li et al. 2024] also rely on sequentially resolving collisions between layers.

3 OVERVIEW

With Bolt, we process an outfit composed of one or more garments in three primary steps.

- **Garment transfer**: We transfer a garment from a source character to a target character, adjusting both the 3D positions and the 2D sewing pattern of the garment, which we use as the rest state of the garment for simulation.
- **Progressive draping**: We simulate each garment, layer by layer, to untangle and drape each garment.
- **Rig transfer**: We transfer the skinning weights from the character's body rig to the corresponding garments.

See Figure 2 for a visual representation of our pipeline.

We develop our system as an offline automated pipeline, prioritizing flexibility, robustness, and scalability over performance. We wrote the entire pipeline using Nvidia's Python Warp [Macklin 2022].

4 GARMENT TRANSFER

As the first step, we transfer garments from a source character to a target character. We do this in two parts: 3D Mesh Transfer, then 2D Sewing Pattern Optimization.

4.1 3D Mesh Transfer

We transfer 3D mesh positions from the source body to the target body. Our algorithm extrapolates a 3D displacement field around the body from a source to a target mesh. We take inspiration from fluid dynamics to create the following physically inspired method.

- Define a sparse regular grid in a narrow band around the body mesh.
- Define displacement boundary conditions on the mesh surface.
- Solve for a viscous, weakly compressible flow in the 3D space around the body:
  - Viscosity allows diffusion of the displacement away from the surface.
  - Unilateral incompressibility moves garments away from areas where pinching might occur, such as under the armpits.

Formulation. We start with the inertialess incompressible viscous fluid equations, inspired by Stokes' equation. Let u be the mesh displacement and u_{body} be the target body displacement from the source. We define the domain Ω as the outside of the body mesh. Let $\delta\Omega_b$ be the body surface, i.e. the domain boundary on which we enforce the displacement boundary conditions. The problem we need to solve is

$$\begin{cases} -\nu D(u) + \nabla p = 0 \\ -\nabla \cdot u = 0 \\ u = u_{body} \quad \text{on } \delta\Omega_b \end{cases}$$

where ν is the viscosity, p is pressure, and $D(u) = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$, the symmetric part of the gradient of u.

We do not want or need pure incompressibility, as different body weight classes will introduce compressing/stretching regions. We instead replace the incompressibility constraint with an elastic term

$$-\nabla \cdot u - \gamma p = 0 \tag{1}$$

where γ is a compliance term set to 0.01.

The previous formulation uses a linear approximation of the volume change ($\nabla \cdot u = Tr(\nabla u)$), which does not behave satisfyingly for large displacements. Moreover, we do not want to penalize stretching, just compression (to move out cloth from zones like shrinking arm pits). We switch to a nonlinear elasticity formulation:

$$-\min(0, \det(\nabla u + I) - 1) - \gamma p = 0 \tag{2}$$

The solve is now a fixed-point iterations loop, where the linearized formulation is still used for each iteration but with an additional bias term accounting for the nonlinear part.

Implementation. To avoid having to generate a mesh, we use a non-conforming formulation. We define a dense regular grid around the body, and mark all cells within a given distance of the body as active. The solve will be done on this sparse subset of the regular grid.

This unfortunately means that the grid nodes do not match the body vertices, so we cannot enforce the boundary condition directly. We instead switch to a weak formulation,

$$\int_{\delta\Omega_g} u, v = \int_{\delta\Omega_g} u_{body}(\Pi(x)) \cdot v, \tag{3}$$

where $\Pi(x)$ denotes the projection of the integration point x onto the body mesh. In practice, we integrate over all "boundary cells" $\delta\Omega_g$, i.e. grid cells which intersect the body, and generate quadrature points over those cells. Then for each quadrature point, we compute the closest point on the surface mesh. If it is inside the mesh,

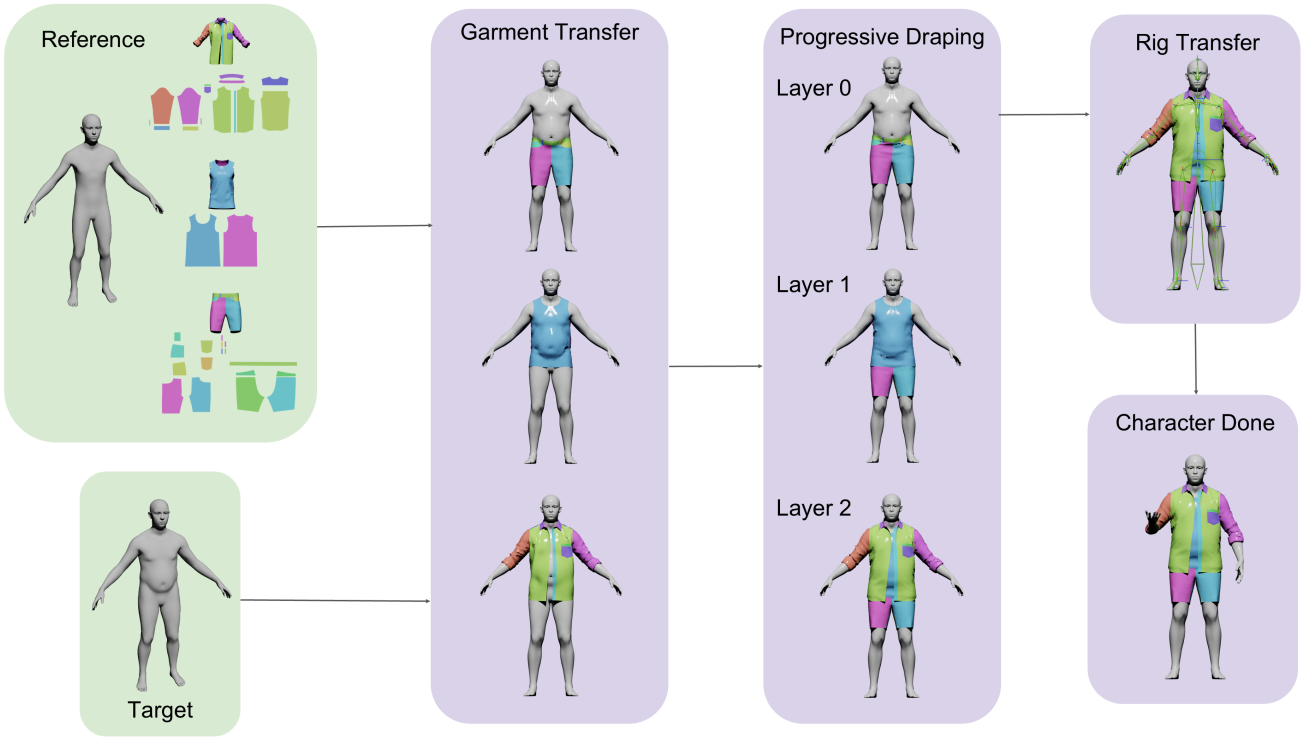


Fig. 2. Overview of the Bolt pipeline using an example character. In this example, we assemble a 3-layer outfit from a source character's body and garments, and target character's body. In the garment transfer step, we fit each garment to the target character separately. In the progressive draping step, we simulate the outfit layers one at a time on the character to drape and untangle the clothing. Finally, we rig garments by transferring the skinning weights from the target body rig to the garments and finish the character.

we enforce the displacement at that point to equal the prescribed displacement of the closest point on the surface. If it is outside the mesh, we project the quadrature point onto the surface and enforce displacement equality there.

As this technique is not guaranteed to generate fully compatible conditions, we do not use a hard constraint to enforce the displacement boundary condition, but a compliant penalty instead. This means that at the end of the solve, the cloth may end up inside the target body because the body has not moved as much as it should have. To work around this, at each "outer iteration" we displace not only the cloth, but the body surface mesh as well. We then compare the displaced body mesh to the target mesh; if we had conforming boundary conditions they should match exactly, but with our formulation there will be a discrepancy. If the gap is too big, we restart the transfer process from the displaced body mesh to the target mesh, and repeat this outer loop until the delta is small enough.

We compute the various terms using Warp's FEM library [Macklin 2022]. At each inner iteration, we get a linear system of the form

$$\begin{bmatrix} A & B^T \\ B & C \end{bmatrix} \begin{pmatrix} \Delta u \\ \Delta p \end{pmatrix} = \begin{pmatrix} f \\ k \end{pmatrix} \tag{4}$$

with A as the matrix corresponding to the viscosity and boundary conditions bilinear forms, B corresponding to the divergence form, and C to the elastic compliance. f corresponds to the prescribed boundary displacement linear form, and k contains the current nonlinear elasticy bias term. As C is diagonal, we can compute the Schur complement of the system, $A + B^T C^{-1} B$, and solve for the displacement Δu using the Conjugate Gradient method.

4.2 Sewing Pattern Optimization

We now adjust the 2D sewing pattern based on the already solved for 3D displacements of the garment. This process optimizes the position of the panel [2] vertices with the following considerations:

- Adjust the sewing pattern piece sizes to accommodate for the increase or decrease in the target body.
- Adjust the sewing pattern positions to minimize the amount of stretch.
- Minimize changes to the boundary of the sewing pattern pieces (i.e. make sure the edges remain smooth).
- Maintain the edge lengths of seams as much as possible on each incident pattern piece.

Our approach to this is inspired by Pietroni et al. [2022], which generates 2D sewing patterns based on a 3D cloth mesh. The method assigns orthonormal tangent vectors to each triangle on a 3D cloth

[2]Note that we use the terms "panel" and "sewing pattern piece" interchangeably.

mesh representing the warp and weft directions of the weave, which are computed using cross-fields. It also "binds" these vectors to the triangle vertex positions so the tangent vectors can deform with the triangle.

For any 2D panel layout assigned to a 3D cloth mesh, we can compute the corresponding warp/weft tangent vectors for each triangle in that layout using this binding information. The goal is to minimize the distortion of each triangle as it goes from the 3D pose to the 2D layout, so the method defines an energy that penalizes deviation from orthonormality for all of the triangle tangents, then finds the 2D layout that minimizes that energy using an optimization method.

We take a similar approach when reshaping our existing 2D panel layout for the target body, although our use case is slightly different.

4.2.1 Defining 3D tangent vectors. We already have enough information to calculate the ground truth warp/weft tangent vectors on the 2D sewing pattern for the original character, as they are just unit vectors in the x and y directions respectively. We can transfer these vectors onto the original 3D source pose for a given triangle like so:

- Find the two edge vectors of the triangle, $\boldsymbol{e}_0 = \boldsymbol{v}_1 - \boldsymbol{v}_0, \boldsymbol{e}_1 = \boldsymbol{v}_2 - \boldsymbol{v}_0$
- Stack these column vectors side by side into a 2x2 matrix and find its inverse $\boldsymbol{M}$
- Use this matrix to bind the tangent vectors to the triangle, and transfer the unit x,y vectors into 3D by finding the 3 by 2 edge matrix of the 3D triangle and right multiplying by $\boldsymbol{M}$.

Note that if we optimize for orthonormality in the way that we outlined above we will recover the original 2D panel layout, as the tangent vectors are all orthonormal in this configuration. To resize the sewing pattern to fit the clothes on the target character, we need to transfer the vectors onto the target character and make some modifications.

4.2.2 Tangent vectors on the target body. We transfer the tangent vectors onto the deformed garment for the target character using the previously computed binding information. If we just optimize for orthonormality again, we will recover the original 2D panel layout without resizing it to the target character, as the information about the target shape is encoded in how the tangent vectors are bound to the triangles.

Typically the 3D tangent vectors won't be orthonormal, as among other things, the transfer between the source character and the target character will scale and stretch the triangles somewhat. We can make them orthonormal again by concatenating them into a 3 by 2 matrix $\boldsymbol{F}$, finding the polar decomposition $\boldsymbol{F} = \boldsymbol{RS}$ and just use the columns of $\boldsymbol{R}$. We can then recompute the bindings of these orthogonalized tangent vectors and optimize for a 2D panel layout. We can bind a 3D tangent vector $\boldsymbol{v}$ to the edge frame by solving the following minimization problem, where $\boldsymbol{E}$ is the 3 by 2 matrix containing the triangle edge vectors as columns and $\boldsymbol{b}$ is a 2 element vector of bind weights. We solve the problem using a pseudo inverse:

$$\underset{\boldsymbol{b}}{\operatorname{argmin}} |\boldsymbol{Eb} - \boldsymbol{v}|^2 = \left(\boldsymbol{E}^T\boldsymbol{E}\right)^{-1}\boldsymbol{E}^T\boldsymbol{v}$$

Using this approach, the optimization will resize the sewing pattern to fit the target character. The fact that each 3D triangle has orthonormal tangent vectors means that its optimal shape in the 2D layout is just a rotated version of its shape in 3D, so the optimization will try and select a 2D layout that matches all the 3D triangle shapes as closely as possible. When simulated, this will lead to a loose fitting outfit, as the optimization is trying to avoid all stretching in the 2D->3D map.

This is not necessarily the desired outcome, since the cloth is often stretched tight over the body in the source pose and there is already strain on the triangles. To get a better fit on the target character's body, our method needs to transfer this strain over to the tangent vectors in the target pose. To do this, we do the following:

- Find the 3D tangent vectors on the source character using the method in the previous section.
- Find the polar decomposition of each triangle's tangent frame, $\boldsymbol{F}_{ref} = \boldsymbol{R}_{ref}\boldsymbol{S}_{ref}$. Even though this is the source character, each triangle may be stretched relative to its shape in the 2D layout if the clothes are tight fitting, and this stretch information is contained in $\boldsymbol{S}_{ref}$, which we save for later.
- Find the 3D tangent vectors on the target character in the same way.
- Find the polar decomposition of each target triangle's tangent frame, $\boldsymbol{F}_{target} = \boldsymbol{R}_{target}\boldsymbol{S}_{target}$
- Swap the stretch information in this tangent frame with the stretch information in the source pose by building the tangent frame $\boldsymbol{F}_{target} = \boldsymbol{R}_{target}\boldsymbol{S}_{ref}$

Rebinding these tangent frames to the triangles and optimizing for a 2D layout preserves the original fit when simulated.

4.2.3 Optimization. We use a simpler triangle deformation energy than Pietroni et al. [2022], who have a quadratic term per triangle forcing the y component of the warp tangent to a target value, a similar term for the x component of the weft and then an As Rigid As Possible [Sorkine and Alexa 2007] term to encourage rigidity. Instead we use an energy that forces the tangent frames to an identity matrix. This energy is translationally invariant, yielding an underdetermined problem. To address this, we add a quadratic penalty term, constraining vertices to their original 2D positions:

$$E = \frac{1}{2}\sum_t A_t|\boldsymbol{F}_t - \boldsymbol{I}|^2 + \frac{1}{2}\epsilon\sum_v |\boldsymbol{p}_v - \boldsymbol{p}_{0v}|^2, \tag{5}$$

Here, the t index runs over all the triangles, A_t is the area of triangle t, and similarly v runs over all vertices and ϵ is a small value, typically 10^{-8}. The symbols $\boldsymbol{p}_v$ and $\boldsymbol{p}_{0v}$ refer to the 2D positions of the v^{th} vertex in the pose we're optimizing and the rest pose respectively. We can concatenate these into a 2N component vector $\boldsymbol{x}$, and write the energy as a quadratic expression using sparse matrices, as the tangents, $\boldsymbol{F}_t$, are linear functions of the 2D vertex positions.

4.2.4 Edges and Seams. Minimizing this energy gives reasonable results, but often leaves untidy edges, whose lengths may not match if they're sewn to other edges at seams. To improve this, we add some extra energy terms that try to force the edges to have the same direction and curvature as they did in the original pattern, while allowing them to scale. For this, we consider the vectors from a vertex i on the edge to its two neighbors, call them z_{0i} and z_{1i} and find a scale factor S_i that brings their values in the original configuration, z^r_{0i} and z^r_{1i}, as close to their current values as possible, ie:

$$S_i = \underset{S}{\operatorname{argmin}} \left(|z_{0i} - S z^r_{0i}|^2 + |z_{1i} - S z^r_{1i}|^2 \right)$$

We then define an energy based on this scaling of the rest pose like so, where L_i is half the sum of the lengths of vertex i's two edge vectors in the rest pose:

$$W_i = \frac{1}{2} L_i \left(|z_{0i} - S_i z^r_{0i}|^2 + |z_{1i} - S_i z^r_{1i}|^2 \right) \tag{6}$$

By ensuring that the vertex neighborhood along the edge is a scaled version of the neighborhood in the original pose, we can preserve the straightness and direction of the edges while still allowing them to change size.

These new energy terms are no longer quadratic, so we need a different optimization method that can handle more complicated energies. For this we use ADMM [Overby et al. 2017]. We formulate the minimization as follows:

$$\underset{\boldsymbol{x},\boldsymbol{z}}{\operatorname{argmin}} E(\boldsymbol{x}) + \sum_i g_i(\boldsymbol{z}_i)$$
$$s.t. \boldsymbol{z}_i = \boldsymbol{D}_i \boldsymbol{x}$$

Where $E(\boldsymbol{x})$ is the energy from equation 5 and $g_i(\boldsymbol{z}_i)$ are the edge energy terms in equation 6. The constraints ensure that the $\boldsymbol{z}_i$ vectors are the vertex neighbor vectors mentioned above. Following [Overby et al. 2017], we can optimize this objective by iterating the following optimizations:

$$\boldsymbol{x}_{n+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left[E(\boldsymbol{x}) + \frac{1}{2} \sum_i w_i \left| \boldsymbol{D}_i \boldsymbol{x} - \boldsymbol{z}_{n,i} - \boldsymbol{u}_{n,i} \right|^2 \right]$$
$$\boldsymbol{z}_{n+1,i} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \left[W_i(\boldsymbol{z}) + \frac{1}{2} \sum_i w_i \left| \boldsymbol{D}_i \boldsymbol{x}_{n+1} - \boldsymbol{z} - \boldsymbol{u}_{n,i} \right|^2 \right]$$
$$\boldsymbol{u}_{n+1,i} = \boldsymbol{u}_{n,i} + \boldsymbol{D}_i \boldsymbol{x}_{n+1} - \boldsymbol{z}_{n+1}$$

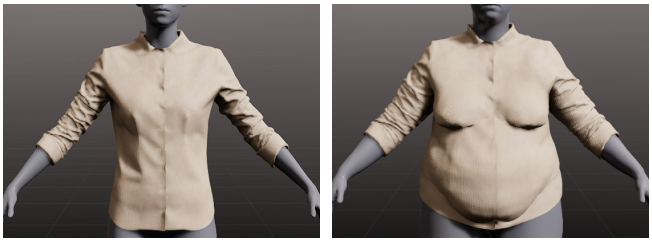
The first optimization can be performed by solving a linear system and the optimizations for $\boldsymbol{z}_i$ have straightforward solutions and can be performed in parallel.

This approach gives cleaner results, although it still allows two edges which are sewn together to vary in length. To fix this remaining issue we tie the scale factors together for the neighborhoods of vertices which have been sewn together.

4.2.5 Final tidy up. By optimizing this energy we achieve good sewing pattern outlines as shown in Figure 3. However, the internal vertices can be noisy, particularly if the cloth is crumpled in the transfer pose. We do a final pass to clean this up by pinning all vertices involved in seams and borders and minimizing a Dirichlet energy over the remaining vertices, where the resulting Laplacian matrix is computed on the original cloth pattern. This smooths the triangle deformations out and gives them similar shapes to their original ones.

(a) source (b) Target

Fig. 3. The original sewing pattern adjusted for the size of the target character.



(a) source (b) Target

Fig. 4. The blouse originally fitted to the source character, and transferred onto the target character using our garment transfer method.

5 PROGRESSIVE DRAPER

The progressive draper untangles and simulates an outfit on the target character. The draper runs a simulation once per layer of clothing to achieve this result. See Algorithm 1 for the steps we take to progressively drape each layer of clothing, and Figure 5 for an example of the draping process on an outfit with four layers.

5.1 Unified Collision Signed Distance Functions

A crucial challenge we face for collision detection is to develop a signed distance function (SDF) for spatial inside/outside queries of a static garment or mesh collider. Common SDF algorithms cannot provide an accurate value range on meshes with open holes, as garment meshes lack topological water-tightness. To address this issue, we employ the generalized winding number method to determine the inside/outside region around the garment. Jacobson et al. [2013] demonstrated that generalized winding numbers robustly determine inside/outside fields, even when meshes contain holes and artifacts. We set the winding number sampling threshold to 0.25. This threshold effectively "seals" the holes in garments created from arms, legs, torsos, and other body parts.

Algorithm 1: Progressive Draper Algorithm

Input: Garment meshes G**, Layer numbers** L**, collider meshes** C **;**

Output: Garment meshes G' **;**

G_s = **SortByLayer** (G, L)
$s_{total} \leftarrow \emptyset$ **;**
foreach $c \in C$ **do**
  $s \leftarrow$ **SDF** (c)**;**
  $s_{total} \leftarrow$ **SDFUnion** (s_{total}, s)**;**
end
$G_{prev} \leftarrow \emptyset$ **;**
foreach $g \in G_s$ **do**
  $s_{total} \leftarrow$ **SDFUnion** $(s_{total},$ **SDF** $(G_{prev}))$**;**
  $g' \leftarrow$ **Simulate** (g, s_{total})**;**
  $G_{prev} \leftarrow G_{prev} \cup g'$
end

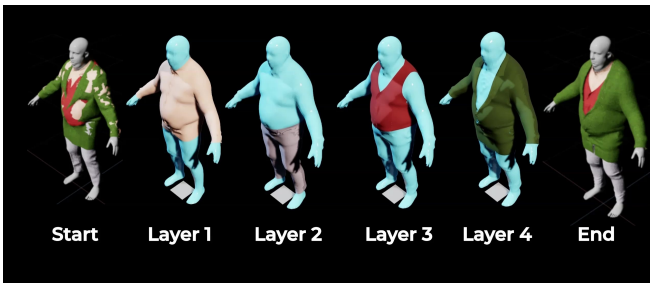


Fig. 5. The progressive draper iteratively simulates each layer to untangle the clothing. Previous layers are frozen, and their geometry is combined into the collider SDF to ensure the next layer successfully pushes it out of the previous layer.

We create a single collision SDF for collision detection between the garment simulation and the underlying collision geometry, i.e. the body and lower garment layers (if any). We take the union of all Signed Distance Functions (SDFs) of each lower garment layer and the body. We combine the SDFs by taking the minimum and adding a small offset ϵ_{sdf} to slightly expand the field. See the blue meshes in Figure 5 for a visual example of the SDFs created at each per-layer simulation.

5.2 Cloth Simulator

We use a panel-based cloth simulator to resolve collisions and drape the garment over the character in a physically plausible manner.

5.2.1 Seams Handling. We represent each seam as a list of vertex pairs within the overall mesh. Some vertices may participate in multiple seams, for example at the corners of the sleeve pattern pieces of a t-shirt. Therefore, we precompute a list of "seam groups": the set of vertices that are all attached together via seam lines.

Artists may sew some panels at a position offset from another panel, such as a pocket attached to the chest of a t-shirt panel. We store a "seam offset" based on the initial condition of sewn vertices imported into the tool and enforce these seam offsets throughout the entire simulation. Most offsets will be zero for sewing pattern pieces that attach along manifold edges. To enforce the seam group constraints after a position step, we first find the seam group's new center by computing the average position and velocity of the seam group. We then update each vertex position to be offset from the group's center and update each vertex velocity as the group's velocity.

5.2.2 Intrinsic Energies. We use anisotropic constitutive models for stretch and bending, to capture the features of real fabrics. Industry tools such as Marvelous Designer measure physical stretch and bending parameters in multiple directions: weft, warp, and shear. These parameters and their relationship to each other create emergent material behaviors [Marvelous Designer 2023]. To accommodate this for stretch, we use the energy described by Kim [2020].

We take inspiration from Gingold et al. [2004a] for our bending energy. We derive the bending energy ψ_{bend} from the triangle-level shape operator, expressed in the coordinate system of the warp and weft directions. Let S be the 2x2 curvature matrix, where $S_{0,0}$ is the curvature in the warp direction, $S_{1,1}$ is the curvature in the weft direction. Let S^r be the rest curvature. We define bending stiffnesses k per panel in each direction: warp, weft, and shear.

$$\psi_{bend} = 1/2\Big((k_{warp}(S_{0,0} - S^r_{0,0})^2 + k_{weft}(S_{1,1} - S^r_{1,1})^2 + k_{shear}(S_{1,0} - S^r_{1,0})^2)\Big) \tag{7}$$

We use a spring-based seam bending energy inspired by [Bridson et al. 2002]. We measure the angle between two triangles and compare it to the rest angle to produce a spring force along with an elastic response and damping term.

5.2.3 Collisions. To efficiently detect collisions between cloth triangles, we leverage a BVH to find nearby triangles. For any contact pair, we verify that the involved triangles are not topologically adjacent, and are not connected via a seam vertex. Using this method, we avoid spurious collisions across sewn pattern pieces.

For collision response, we apply impulses between the closest points between two colliding triangles. We interpolate impulses into the triangle faces as described by Bridson et al. [2002]. Each impulse is made up of an elastic portion, a damping portion, and a coulomb friction portion. For collision between the cloth triangles and all the clothing layers below and the body, we utilize the SDFs as described in Section 5.1. We detect the collisions between triangles' vertices and the union signed distance field and apply impulses to the colliding vertices. The collision normal is simply the gradient of the SDF.

Typical parameters used in the simulation for collision are available in Table 1. Since initial garment configurations frequently start off in heavily interpenetrated states, we globally damp velocities by 90% at each step of the simulation to reduce the likelihood of a large energy in the initial interpenetrated configuration.

6 AUTOMATIC SEMANTIC PROXY GENERATION

We develop a method to automatically filter out panels based on type, to produce a simulation-ready proxy mesh. Using this approach, we

Parameter	Value
Collision Force Coefficient	2500000
Collision Damping Coefficient	25
Coulomb Friction Coefficient	25
ϵ_{sdf}	0.2

Table 1. Simulation Parameters

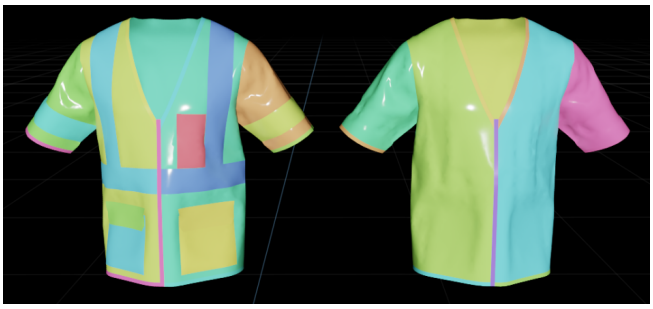


Fig. 6. Initial garment (left) and generated proxy mesh (right).

can address the simulator's struggle to drape certain garments with complex construction, such as thin decorative pieces like pockets or straps. For instance, a high visibility vest may have reflective straps that are tightly constrained to follow the garment's surface. Improper tuning of the thickness parameter for this material can cause significant collision forces between the strap and its attached surface, without adding significant value to the simulation result. We consistently label each panel in our dataset with semantic meaning, to make panel filtering simple. We show the proxy mesh of a high visibility vest generated using our method in Figure 6. Note that this approach requires us to make the following assumptions:

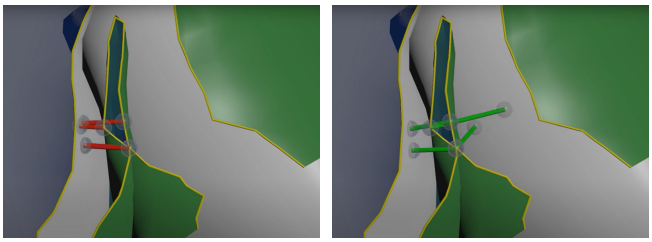
- We assume that all vertices of a removed panel piece have a nearby part of the proxy mesh.
- We assume the proxy mesh does not undergo significant rotation. Since we use world space offsets, we are not robust to rotations of the garment.

We use these generated proxy meshes during garment transfer and the progressive draper step. Note that for the collisions of the draper to be correct on each layer, we use the detailed mesh of the lower layers as part of the collision surface. This ensures we include the thickness of pockets/straps in subsequent layer simulations.

7 RIG TRANSFER

As the final step to prepare a character's outfit for animation, we transfer the skinning weights of the character's rig from the body to the cloth.

We initially tried a straightforward approach by searching for the closest point between a cloth vertex and the body surface, then propagating the skinning weights from the body to the cloth. While this approach may work in some simple cases, it was not capable of handling situations where cloth layers can get arbitrarily close or even inverted, since it may associate the cloth piece with the wrong



(a) Transfer by positions (b) Transfer by normals

Fig. 7. Cross-section of garment layers and their associated body surface locations for skinning weights transfer based on different strategies. Using garment normals helps each face find the correct body location to inherit skinning weights from.

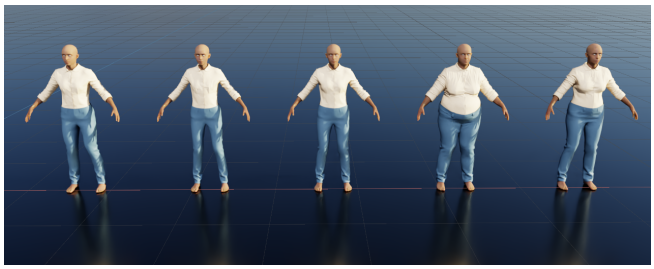


Fig. 8. A single outfit fit onto characters of different body types.

body part (Fig. 7a). We observe these problem cases happen most often in tightly pinched areas, such as the armpit of the character. To ensure a more robust rig transfer, we use the cloth's normals to find the best points on the body mesh to transfer skinning weights from (Fig. 7b). We compare these two approaches in Figure 7.

We apply normal smoothing to improve the smoothness of the skinning weights assigned to each cloth vertex. We also constrain skinning weights to be identical across seam lines, which ensures continuity across seam boundaries. Finally, we use proximal transfers as a fall-back strategy if the normal based search path does not find a body vertex to use for skinning weights association. With all of these strategies together, we create more robust animated character rigs.

8 RESULTS

Using our pipeline, we can assemble, fit, drape, and rig a wide range of outfits onto new characters. As shown in Figure 8, we can fit a single outfit onto multiple characters of widely varying body shapes.

We designed the pipeline as an offline process. We measured performance of the two slowest steps of the pipeline, garment transfer and draping, on a wide range of outfits using an NVIDIA RTX 5000 Ada Generation Laptop GPU. These metrics show the scalability of Bolt for generating large character sets compared to manually transferring outfits.

Figure 9 shows the transfer time for a wide range of outfits fit to a single obese class-1 male character. The total transfer time depends on the number of garments, the convergence rate of the conjugate gradient solver, and the number of source bodies involved in the

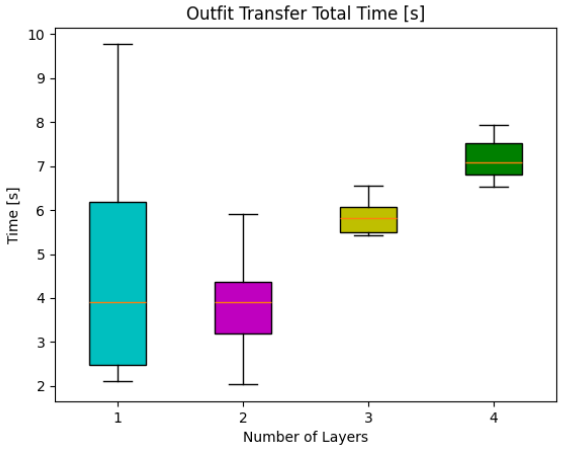


Fig. 9. Box plot of transfer times for different outfits fit to a single obese male character, organized by number of layers in each outfit.

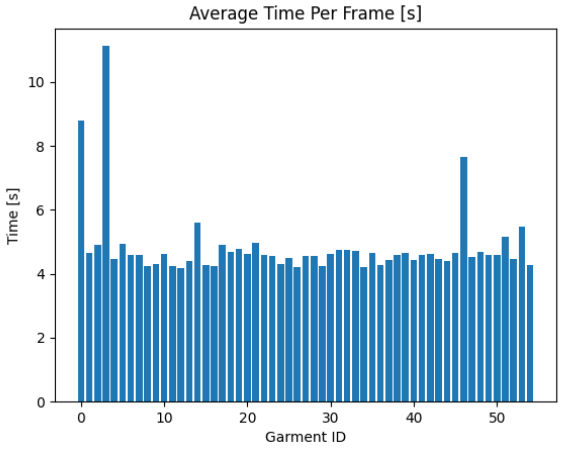


Fig. 10. Histogram of average simulation times per frame across 55 different garments. Since the draper simulation only simulates one garment at a time, an outfit's total simulation time is the sum of the N layers involved in the outfit. All garments are simulated for only 6 frames.

transfer. If two garments are fit to two different source bodies, say a skirt fit to a female source character and a button-up shirt fit to a male source character, the total transfer time includes the time to transfer both garments from their respective source bodies to the target body.

Figure 10 shows the average draping time per frame for a range of garments. The total time to assemble an outfit will depend on the combination of garments, and their individual simulation times. For our results, we run each garment for 6 frames of the simulation. Across our dataset of garments, the average frame time is 4.8 seconds.

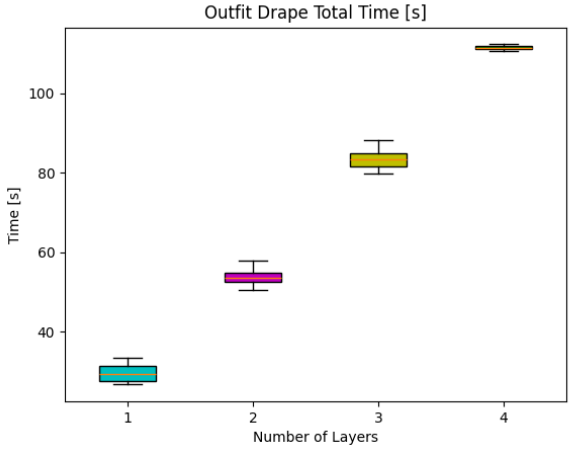


Fig. 11. Box plot of draping times across a collection of outfits fit to the obese class 1 individual, sorted by number of layers in the outfit.

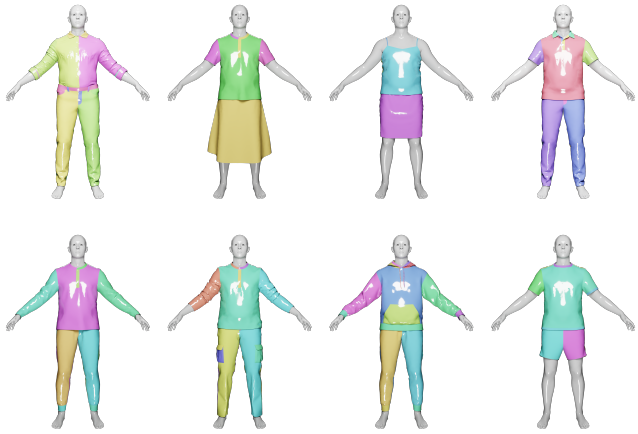


Fig. 12. Multiple outfits on the same character. In this visualization each panel is assigned a different random color.

Figure 11 shows the draping simulation time for an outfit collection fit to an obese male character. We observe that the progressive draper step is the main performance bottleneck of the system, since we simulate each layer separately. We demonstrate successful fits on large characters of up to 4 layers, as previously shown in Figure 5. For our 4-layer outfits, the slowest completes in 112 seconds. Figure 12 shows a subset of outfits on a single class one obese character. See Table 2, and Table 3 for examples showing different characters wearing a variety of outfits.

Note that these performance numbers do not include file I/O, rigging, or any of the transfer initialization and simulation initialization processes. We ran an experiment to generate 220 characters, and observed an average wall-clock time of approximately 6 minutes per character (run on a cloud instance with an A40 GPU). Using cloud resources, and with a budget of 64 A40 GPUs, we were able to

generate all 220 in under 30 minutes. We have used our system to produce characters in practical applications. At the time of writing, we have generated and published just over 100 characters for demos or public releases. As part of our submission video, we show images of nearly 1000 characters we generated using this tool.

9 CONCLUSION

We have demonstrated a tool for the automatic assembly, fitting, draping, and rigging of many garments onto many characters. Using this system, we can assemble garments into new outfits, and fit them onto new characters in a scalable way. As the demand for large numbers of digital humans continues to grow, our approach can help keep up with that demand.

9.1 Limitations

In the garment transfer step, we assume a per-vertex correspondence between the source body and target body. To support general character refitting, we would need a spatial correspondence map between any two characters. We derive all of our characters from a single parametric model, which ensures they all share the same topology and are compatible in our pipeline.

In the progressive draping step, we cannot propagate information from later simulation layers back to earlier layers. For example, with a collared shirt on layer 0 and a sweater on layer 1, we cannot achieve a look where the collar is above the sweater since the sweater's simulation will see the final "frozen" dress shirt positions as a collision surface. To address this issue, we would need to untangle all garments simultaneously as part of the same simulation, and/or carefully break down the garment sewing pattern pieces into "sub-layers" that could interleave with one another. For example, we could put the body of the shirt on layer 0, the sweater on layer 1, and the collar of the shirt onto layer 2, as long as we add constraints to ensure the collar remains connected to its counterpart pattern piece on layer 0.

Our cloth simulator cannot be used in full scale animations because our seam handling approach does not allow for seam group rotations. The simulator also does not have a robust self-collision untangling method, so if the cloth starts off in a tangled state, due to an artifact introduced in an earlier stage of the pipeline, the simulator will not fix the tangling artifact, and it will persist even after the draping stage.

Our system may introduce artifacts for characters with extreme body shape differences. For example, if a class 3 obese character's body has a region that causes the surface to fold back onto itself, the transfer process will place the cloth close to the surface of the body mesh, essentially "pinching" the fabric underneath the body region. Once the cloth starts off in a pinched configuration, it rarely succeeds in pulling the cloth out from the pinched section in a way that is nicely untangled.

9.2 Future Work

In addition to addressing the limitations mentioned previously, we would like to investigate other approaches to improve the robustness of our pipeline. We would like to investigate untangling approaches that can resolve self-collisions, especially in tight/pinched areas, to help the system recover from poor garment initialization or issues during garment transfer.

Our garment transfer method resizes the garment solely on the character's differences in geometry. Sometimes this approach can make garments too form-fitting on larger characters, and place extra material in ways that modify the original style or design. Additionally, we are inspired by the traditional tailoring process where a tailor takes specific measurements of the body for fitting a suit. We would like to investigate using similar measurements to achieve smarter garment refitting.

ACKNOWLEDGMENTS

We thank Nvidia for supporting this project and in particular Vanni Brighella, Miguel Guerrero, Adeline Aubame, and Qiao Wang for help with creating the input assets and software support. We also thank Michael Honke for proof-reading our paper, and Ken Museth, Simon Yuen, and Miles Macklin for helpful discussions and feedback.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 43–54.

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling cloth. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 862–870.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.* 21, 3 (2002).

Remi Brouet, Alla Sheffer, Laurence Boissieux, and Marie-Paule Cani. 2012. Design preserving garment transfer. *ACM Transactions on Graphics* 31, 4 (2012), Article–No.

Thomas Buffet, Damien Rohmer, Loic Barthe, Laurence Boissieux, and Marie-Paule Cani. 2019. Implicit untangling: A robust solution for modeling layered clothing. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Hsiao-yu Chen, Egor Larionov, Ladislav Kavan, Gene Lin, Doug Roble, Olga Sorkine-Hornung, and Tuur Stuyck. 2024. Dress Anyone: Automatic Physically-Based Garment Pattern Refitting. *arXiv preprint arXiv:2405.19148* (2024).

Fernando de Goes, Donald Fong, and Meredith O'Malley. 2020. Garment refitting for digital characters. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*. 1–2.

T. F. Gast and C. Schroeder. 2015. Optimization integrator for large time steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 31–40.

Yotam Gingold, Adrian Secord, Jefferson Y. Han, and Eitan Grinspun. 2004a. A Discrete Model for Inelastic Deformation of Thin Shells. https://cims.nyu.edu/gcl/papers/secord2004sds.pdf. (2004). Accessed: 2023-11-27.

Yotam Gingold, Adrian Secord, Jefferson Y Han, Eitan Grinspun, and Denis Zorin. 2004b. A discrete model for inelastic deformation of thin shells. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Citeseer.

Artur Grigorev, Giorgio Becherini, Michael Black, Otmar Hilliges, and Bernhard Thomaszewski. 2024. ContourCraft: Learning to Resolve Intersections in Neural Multi-Garment Simulations. In *ACM SIGGRAPH 2024 Conference Papers*. 1–10.

Kai He, Kaixin Yao, Qixuan Zhang, Jingyi Yu, Lingjie Liu, and Lan Xu. 2024. DressCode: Autoregressively Sewing and Generating Garments from Text Guidance. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–13.

Alec Jacobson, Ladislav Kavan, and Olga Sorkine. 2013. Robust Inside-Outside Segmentation using Generalized Winding Numbers. *ACM Trans. Graph.* 32, 4 (2013).

Theodore Kim. 2020. A Finite Element Formulation of Baraff-Witkin Cloth. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* 39, 8 (2020).

Maria Korosteleva and Sung-Hee Lee. 2022. NeuralTailor: Reconstructing Sewing Pattern Structures from 3D Point Clouds of Garments. *ACM Trans. Graph.* 41, 4 (2022), 16 pages. https://doi.org/10.1145/3528223.3530179

Maria Korosteleva and Olga Sorkine-Hornung. 2023. GarmentCode: Programming Parametric Sewing Patterns. *ACM Transaction on Graphics* 42, 6 (2023), 16 pages. https://doi.org/10.1145/3618351 SIGGRAPH ASIA 2023 issue.

Dohae Lee, Hyun Kang, and In-Kwon Lee. 2023. ClothCombo: Modeling Inter-Cloth Interaction for Draping Multi-Layered Clothes. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–13.

Yu Lee and Moon-Ryul Jung. 2009. Retargeting Motion of Clothing to New Characters.. In *GRAPP*. 280–285.

Ren Li, Benoît Guillard, and Pascal Fua. 2024. ISP: multi-layered garment draping with implicit sewing patterns. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1751, 26 pages.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. *Acm Transactions on Graphics (TOG)* 36, 3 (2017), 1–16.

Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).

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

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

Marvelous Designer. 2023. FABRIC PHYSICAL PROPERTIES: Adjust Stretch-Weft/Warp, Shear. https://support.marvelousdesigner.com/hc/en-us/articles/360037396031-FABRIC-PHYSICAL-PROPERTIES-Adjust-Stretch-Weft-Warp-Shear. Accessed: 2023-11-27.

Shanthika Naik, Kunwar Singh, Astitva Srivastava, Dhawal Sirikonda, Amit Raj, Varun Jampani, and Avinash Sharma. 2024. Dress-Me-Up: A Dataset & Method for Self-Supervised 3D Garment Retargeting. *arXiv preprint arXiv:2401.03108* (2024).

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Trans. Vis. Comput. Graph.* 23, 10 (2017), 2222–2234. http://dblp.uni-trier.de/db/journals/tvcg/tvcg23.html#OverbyBLN17

Nico Pietroni, Corentin Dumery, Raphael Guenot-Falque, Mark Liu, Teresa Vidal-Calleja, and Olga Sorkine-Hornung. 2022. Computational Pattern Making from 3D Garment Models. arXiv:2202.10272 [cs.GR]

Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. 2017. ClothCap: Seamless 4D clothing capture and retargeting. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–15.

Xavier Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation'97: Proceedings of the Eurographics Workshop in Budapest, Hungary, September 2–3, 1997*. Springer, 177–189.

Min Shi, Yukun Wei, Lan Chen, Dengming Zhu, Tianlu Mao, and Zhaoqi Wang. 2021. Learning a shared deformation space for efficient design-preserving garment transfer. *Graphical Models* 115 (2021), 101106.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Geometry Processing*, Alexander Belyaev and Michael Garland (Eds.). The Eurographics Association. https://doi.org/10.2312/SGP/SGP07/109-116

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (November 2018), 204:1–10.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 205–214. https://doi.org/10.1145/37401.37427

Pascal Volino and Nadia Magnenat-Thalmann. 2006. Resolving surface collisions through intersection contour minimization. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 1154–1159.

Huamin Wang. 2018. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–10.

Martin Wicke, Hermes Lanker, and Markus Gross. 2006. Untangling cloth with boundaries. In *Proc. of Vision, Modeling, and Visualization*, Vol. 349. 356.

Juntao Ye, Guanghui Ma, Liguo Jiang, Lan Chen, Jituo Li, Gang Xiong, Xiaopeng Zhang, and Min Tang. 2017. A unified cloth untangling framework through discrete collision detection. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 217–228.

Cyril Zeller. 2005. Cloth simulation on the gpu. In *ACM SIGGRAPH 2005 Sketches*. 39–es.

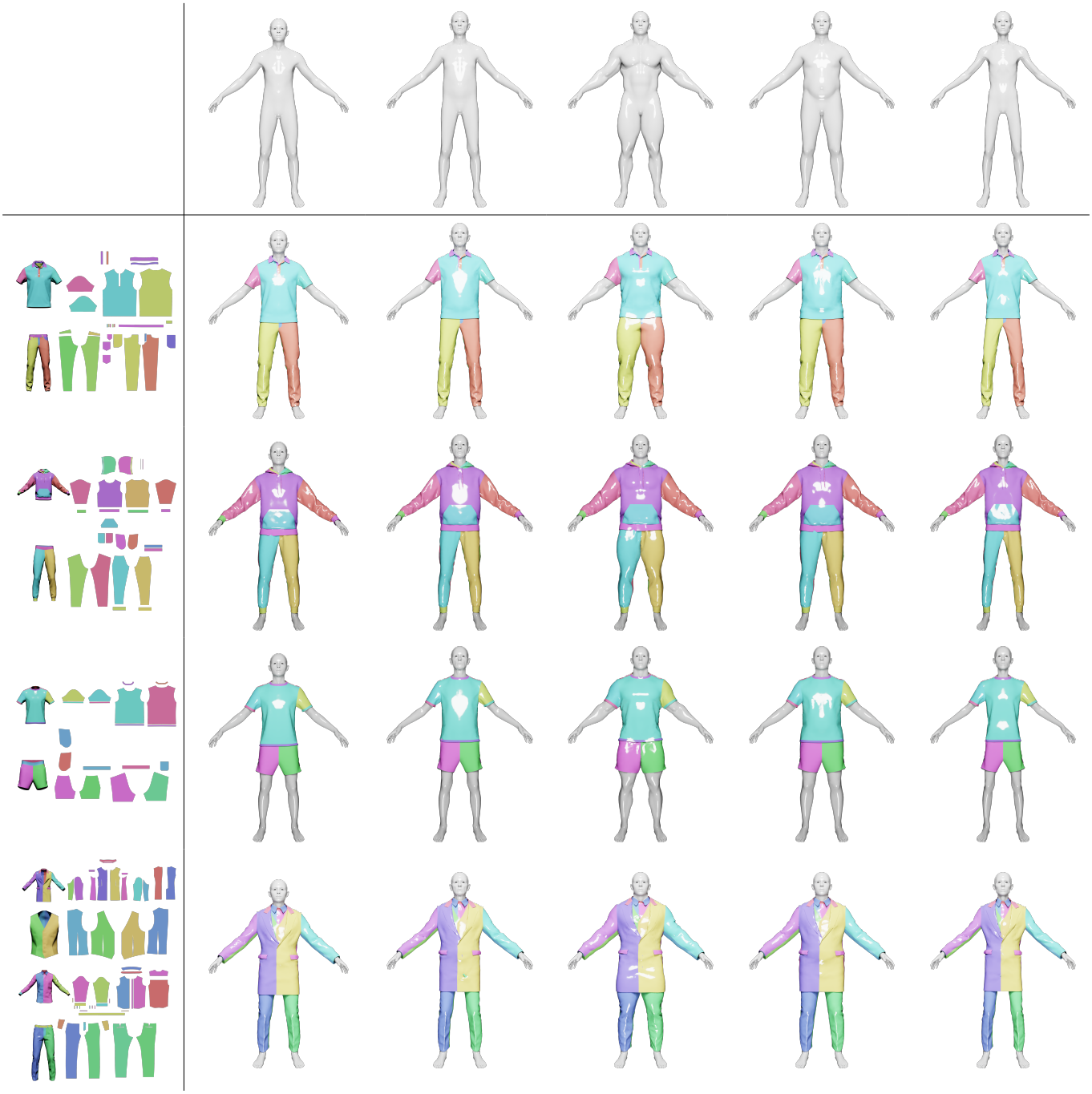


Table 2. We demonstrate the end result of refitting and draping outfits composed of diverse garments onto a variety of male characters using the Bolt pipeline.

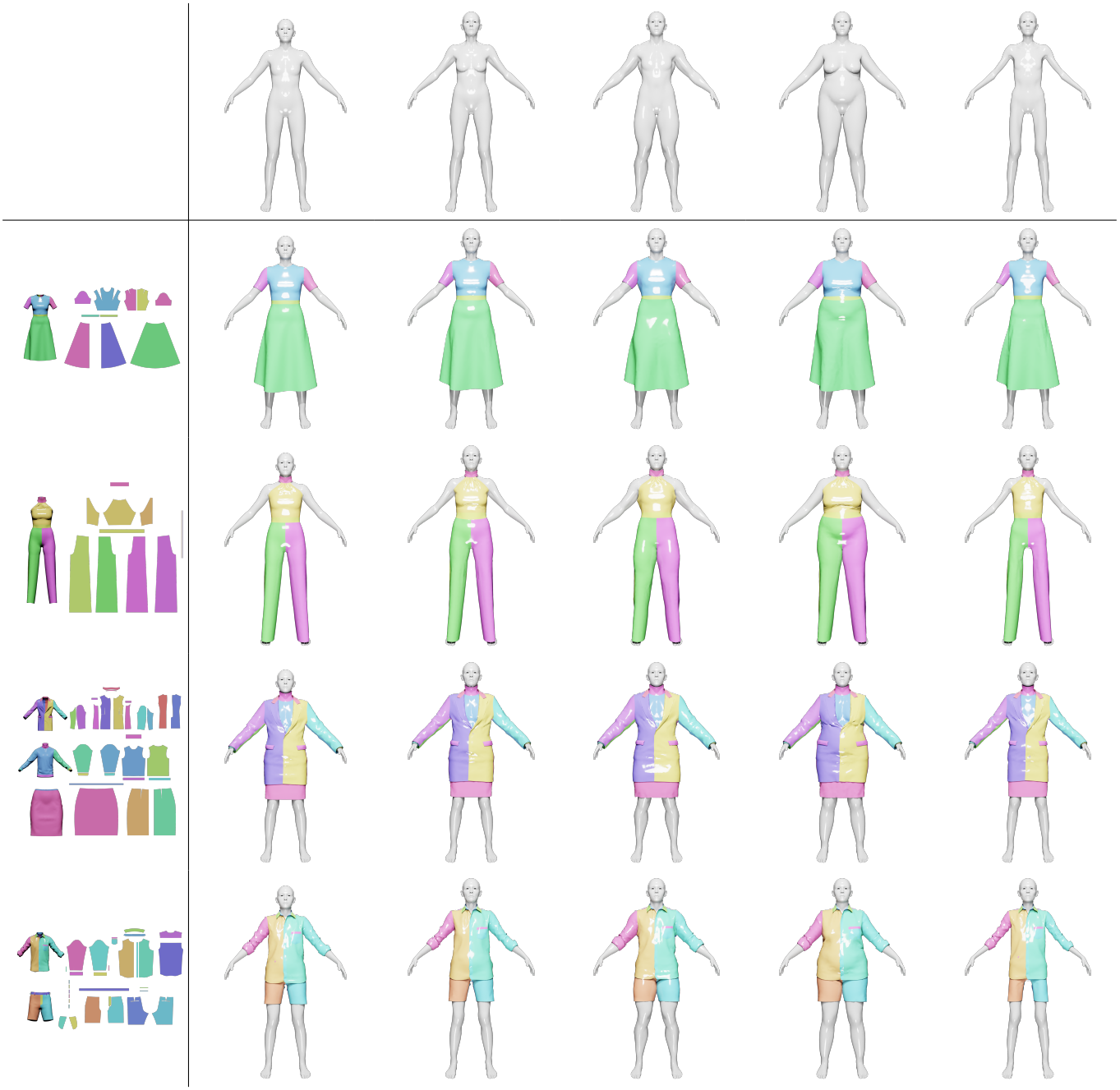


Table 3. A variety of outfits composed of different garments are refitted and draped onto a range of female characters using Bolt.