Interactive Multilayer Gaussian Garments for Low-Cost Try-On

Ryan S. Zesch
Texas A&M University
USA
rzesch@tamu.edu

I-Chao Shen
The University of Tokyo
Japan
jdilyshen@gmail.com

Haoran Xie
Japan Advanced Institute of Science and Technology
Japan
xie@jaist.ac.jp

Bo Zhu
Georgia Institute of Technology
USA
bo.zhu@gatech.edu

Shinjiro Sueda
Texas A&M University
USA
sueda@tamu.edu

Takeo Igarashi
The University of Tokyo
Japan
takeo@acm.org

Figure 1: We present a novel method for interactive mix-and-match avatar dressing of multilayer Gaussian garments. To address penetrations between the body and the garment, as well as between the garment layers, we incorporate a novel screenspace visibility culling method inside the 3D Gaussian splatting renderer. We also show a webcam-based interactive virtual try-on.

ABSTRACT

Numerous recent works have utilized 3D Gaussian Splatting to represent high-fidelity digital avatars. However, none have enabled interactive multilayer Gaussian garments for virtual try-ons without relying on expensive hardware, such as a camera array and/or multiple GPUs. To enable affordable mix-and-match dressing—dressing 3D avatars with realistic and complex combinations of garments—it is crucial to handle the interactions between multiple layers of garments using consumer-level capturing hardware. To address this, we present a novel screenspace layer resolution method combined with physical simulation and Gaussian garments to enable realistic multilayer mix-and-match avatar dressing at interactive rates using low-cost hardware. As an offline process, we capture multiple static garments individually using only a single mobile camera on a static mannequin and then perform a dual reconstruction of Gaussians and simulation mesh. During runtime, these Gaussians are driven by a fast but simple physics simulator, whose output may contain inter-penetrations across garment layers. Our method fixes these in screenspace by rasterizing the simulation mesh from various camera views and culling the Gaussians that are skinned to unseen mesh triangles. We show the effectiveness of our approach by demonstrating mix-and-match dressing results at interactive rates using short-sleeves, long-sleeves, a fur vest, and a singlet. Additionally, we showcase a webcam-based interactive try-on application to further illustrate the capabilities of our system.

Graphics Interface, Kelowna, BC, 2025

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Computing methodologies** → **Image-based rendering**.

KEYWORDS

Virtual Try-On; Gaussian Splatting; Multilayer Garment

ACM Reference Format:

Ryan S. Zesch, I-Chao Shen, Haoran Xie, Bo Zhu, Shinjiro Sueda, and Takeo Igarashi. 2025. Interactive Multilayer Gaussian Garments for Low-Cost Try-On. In *Proceedings of Graphics Interface.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Realistic and animatable 3D clothed human avatars play an important role in many graphics applications, such as telepresence and virtual try-on. The quality of these avatars has significantly improved due to recent advancements in shape representation and rendering techniques [70, 71], including those based on NeRF [13, 14, 50, 63] and 3D Gaussian Splatting (3DGS) [29, 31, 34, 35, 37, 74]. These works typically capture the entire human body with clothing as a whole and represent the whole human body as a single layer. There are few works that support modeling the body and garments as separate layers, enabling garment transfer between avatars [13, 14, 35, 70].

However, these methods still have several limitations. First, they rely on data captured in a studio setup with camera arrays and custom lighting configurations, which restricts their applicability. Second, they typically model a single layer of garment, limiting the dressing to a single layer or pre-defined combinations of garments on avatars. Finally, while some multilayer garment dressing is achievable [70], their method cannot handle the case where the inner layers are significantly occluded, and requires extensive computing power to achieve an interactive rate.

In this paper, we propose an interactive multilayer mix-and-match avatar dressing framework based on 3DGS garment representation. The key idea of our framework is to use a screenspace layer resolution method, rather than a 3D collision resolution method, so that we can use a simple, lightweight physics simulator to drive the motion of the multilayer garment meshes with the Gaussian particles skinned to them. We generate collision-aware (but not necessarily collision-free) motions through physical simulation rather than relying on skinning and other geometric constraints used in previous works, such as D3GA [74]. Additionally, these Gaussians are reconstructed from *static* captures of individual garments, and their spherical harmonics are transformed at runtime based on the deformation of the cloth. This facilitates a simple capturing process that allows us to reconstruct the Gaussian particles and the physics simulation mesh simultaneously, with a monocular RGB camera and a single GPU.

Our system is agnostic to how the garment motion is generated—it simply takes as input the motion over time of possibly overlapping layers of garments. This input can in theory come from any traditional physics simulator [2, 6, 46] or a neural simulator [4, 18, 58, 59]. Despite the existence of many high-performance GPU cloth simulators [26, 33, 68] and collision resolution techniques [3, 7, 28, 43, 65, 69], it remains a challenge to incorporate these works into an interactive try-on system for two reasons. First, these techniques still require non-trivial collision parameter tuning or collision proxy tweaks by experts, often for each simulation garment, to generate penetration-free results. Second, it requires significant engineering to incorporate these techniques without adversely affecting the performance of the 3DGS renderer. Extreme care must be taken to not only keep the simulation cost low but also to minimize the amount of memory transfers and copies. There has also been recent work on mixing 3DGS and simulation [72], but extending this work for multilayer cloth is challenging for the same two reasons: collision parameters must be tuned carefully for cloth-cloth interactions, all the while maintaining interactive rates.

Therefore, we use a simple GPU cloth simulator based on Position-Based Dynamics (PBD) [44] that can directly communicate with the 3DGS renderer with minimal overhead. Any body-cloth or cloth-cloth intersections produced by this light-weight simulator are then fixed by our screenspace layer resolution method, which is particularly well-suited for facilitating an interactive application with frequently switched contexts under limited GPU resources. Additionally, the robustness of PBD is advantageous for handling the potentially noisy initial conditions. Due to our mix-and-match system, we inevitably start the simulation with some initial penetrations between the cloth and body, as well as between the cloth layers.

Our main contributions can be summarized as:

- A novel screenspace layer resolution method for multilayer garment visibility checking (§3.1).
- A simple yet effective method for the dual reconstruction of Gaussian particles and simulation meshes from static garment captures (§3.2).
- An end-to-end interactive system, including a webcam-based interface, for dressing 3D avatars using low-cost hardware for capture and visualization, avoiding prohibitive techniques such as camera arrays (§3.3 and 3.4).

2 RELATED WORK

2.1 Clothed Human Avatars

The task of reconstructing and animating a photorealistic human avatar has been a recent research focus in graphics. Early attempts using RGB-D cameras and mesh-based representations [51, 52, 57] enable applications such as movie production, VR, and telepresence. However, the 3D data required to reconstruct these avatars is limited and expensive to acquire. In recent years, avatars represented by NeRF [13, 14, 50] and 3DGS [29, 31, 34, 35, 37, 53, 74], just to name a few, have gained attention due to the ease of obtaining input data and the higher rendering quality. Initially, these methods represented a clothed human avatar using a single-layer representation [34] for simplicity, meaning that the body and the cloth are modeled together, preventing their use in virtual try-on applications. Later, methods have been proposed to solve this problem using two-layer representation such as mesh+NeRF [13, 14] and compositions of 3DGS [35, 74]. Among them, only LayGA [35] considers the collision between the body layer and the cloth layer and achieves high-quality garment transfer results. However, their method is not real-time, and is limited to handling a single layer of garment and generating motions using simple skinning techniques and geometric constraints rather than physical simulation. Therefore, they cannot achieve realistic mix-and-match avatar dressing. In contrast, our method reconstructs 3DGS for each garment separately and generates realistic mix-and-match multilayer avatar dressing driven by a physical simulation with a screenspace scheme to resolve the different layers of garments. Similarly, while Gaussian Garments [56] achieve multilayer try-on, they still require an expensive camera setup and fail to produce realtime interactive try on. While many other multilayer clothed human techniques exists, mesh based methods often fail to be realtime [27, 61], and often have many limitations such as pose restrictions [60] or lack of dynamics [10]. In the same vein, screenspace try-on methods based

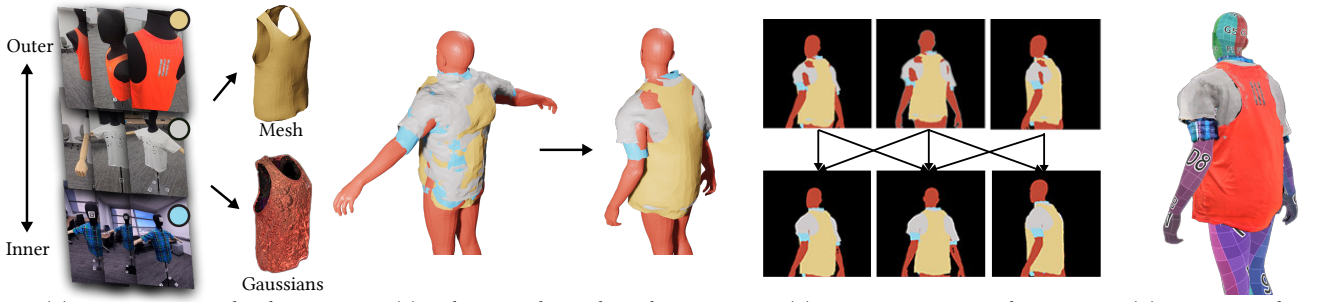

Figure 2: Overview of our pipeline. For each individual garment, we optimize surface-aligned Gaussians and skin them to a co-optimized garment mesh. (A) Given a set of garments and a predefined order (outer to inner), our method generates a multilayer clothed human avatar in the following steps. (B) At each frame, we first perform a real-time physics simulation, which may contain erroneous protrusions. (C) We then fix the remaining layering problems in screenspace, and (D) render Gaussians only if their skinning triangle is visible in our processed triangle map.

on diffusion [9, 73] or image warping [30] fail to be interactive, though we view these as orthogonal to our approach. Additionally, many methods of mesh reconstruction exist. Splatting based methods [19, 21, 67] are often not well suited for simulation without further processing. Sewing patten methods [32, 36] are in principle compatible with our method, though Gaussian correspondence may face difficulties. We opt for a simple, effective solution for reconstruction, and our online method is compatible with meshes generated by any methods, provided they are simulation-ready, elements are reasonably sized, and splats are surface-aligned.

2.2 Screenspace Collisions

Screenspace collision resolution approaches have been used to compute collisions for many decades [45, 62]. With the advent of GPUs, these approaches became highly efficient, able to find potentially colliding sets of triangles at interactive rates [15–17]. More recently, raycasting on GPUs has been used to detect and minimize intersection volumes [12, 66]. Although related, these approaches are not directly applicable in our context. Critically, related work on image-based collisions focuses primarily on collisions between volumetric objects [1, 11], whereas our method focuses on a more specific problem of resolving the visibility of cloth layers with known ordering. Our screenspace approach is designed specifically for layered garments and is used not for determining overlapping primitives or volumes but as a way to show or hide Gaussian particles.

3 METHODS

We now present our methods for achieving interactive multilayer clothed avatars using 3D Gaussians skinned to garment meshes. As illustrated in Fig. 2, our method first reconstructs garments individually for mix-and-match try-on. Then, users select ordered garments, and we perform a lightweight, real-time cloth simulation using the garment meshes, which may contain many erroneous protrusions of inner garments through outer garments. Then, our screenspace layer resolution method corrects them by iteratively identifying incorrectly ordered triangles from multiple viewpoints, and aggregates this information among these viewpoints. The major advantage of our method is we replace expensive 3D collision

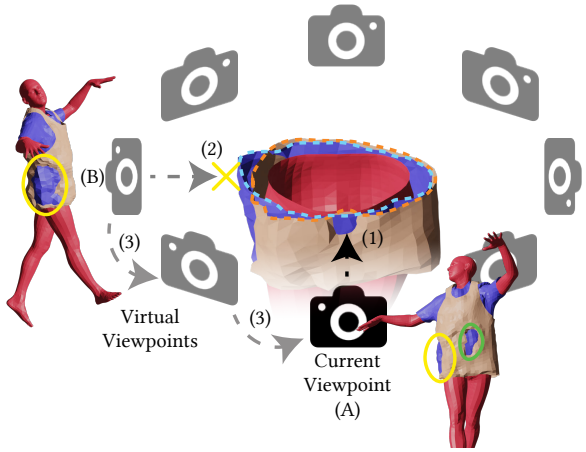

Figure 3: The propagation of layer penetration information between virtual viewpoints. We illustrate how our method handles penetrations of a blue inner garment with an orange outer garment. (1) Penetrations seen from the current viewpoint (A) are easily detected—we can check if triangles on this ray are out of order. (2) Some penetrations may not be detectable from the current viewpoint (A) by traversing the triangles along a ray from that viewpoint, but can be detected from virtual viewpoints (B). (3) Virtual viewpoints share penetration information to remove penetrations from the current viewpoint.

processes with a rasterization-based visibility check to determine which Gaussians can be seen from a given viewpoint. We describe our layer resolution method in §3.1, the preprocessing steps in §3.2, and other details in §3.3 and 3.4.

3.1 Screenspace Garment Layer Resolution

In our system, multiple garments, represented as Gaussian splats skinned to triangle meshes, are simulated and displayed based on a predefined garment order. Perfectly maintaining this garment order

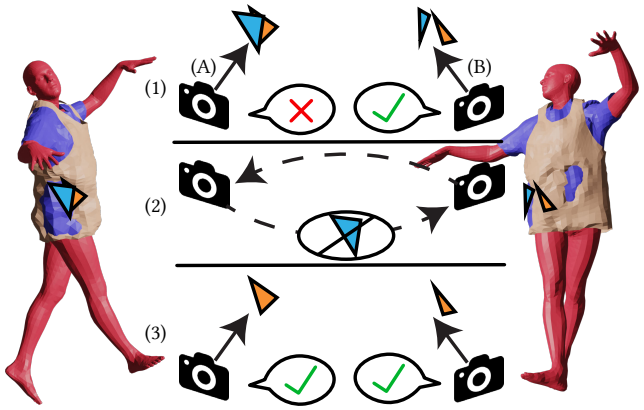

Figure 4: We illustrate how our method handles intersecting triangles. In our intra-view pass (1), our method detects when garment triangles have the wrong ordering. This wrong ordering is only detectable from some viewpoints—from camera (A), but not camera (B). In our inter-view pass (2), viewpoints share culling information, so penetrating triangles can be removed from all viewpoints (3).

everywhere is challenging for simulations at realtime framerates. To achieve visually pleasing multilayer clothed avatar rendering, we use the knowledge of the intended garment order to design a screenspace layer resolution strategy.

Intuitively, a Gaussian, which is skinned to a simulation triangle, should be displayed only if that simulation triangle can be seen from the current camera viewpoint. However, penetrating triangles (*e.g.,* those from the inner garment) may be visible from the camera since we do not rely on the interactive physics simulator to be collision-free. A fundamental question is therefore, how do we identify penetrations while rendering the Gaussians? Consider tracing a ray from the camera. Along this ray, if there is a penetration between garment layers, one would expect to incorrectly encounter an inner garment before an outer garment. In reality, such an ordering is neither a sufficient nor necessary condition to determine if a penetration is present—more information is needed. For example, as demonstrated in yellow in Fig. 3, a garment may be penetrating perpendicularly to the view direction, in which case the ray would hit the inner garment without encountering the outer garment. Using information from a single camera view is not sufficient to address such cases.

To address this issue, at each frame, we render our simulated garment meshes at low resolutions from several virtual viewpoints, in addition to the current camera viewpoint. At each viewpoint, we render triangle maps (or "depth peels") for the first k triangles encountered per pixel. We then perform culling in order to reject certain triangles from being considered as visible from any virtual viewpoint. If a triangle of an inner garment is seen penetrating from one viewpoint, that triangle should be discarded from all viewpoints. Our method is therefore based on iteratively identifying and discarding penetrating triangles from multiple viewpoints.

From each viewpoint, we rasterize a triangle ID map, a normal map, and a depth map for k depth-peeling layers. We rasterize all garment meshes and body mesh (SMPL-X [49]) together, using an orthographic projection, from n virtual viewpoints. (We use $n = 17$ in our implementation.) In order to detect penetrating triangles, we iterate between two steps. First, we perform *intra-view* culling, which detects triangles that are penetrating from each virtual viewpoint. Then, we propagate the triangles that are found to be penetrating between each virtual viewpoint in *inter-view* culling. The intuition behind this is visualized in Fig. 3 and Fig. 4. In short, penetrations occurring parallel to a camera's viewing direction can be caught by considering following triangles along that ray, but penetrations perpendicular to the viewing direction are not easily detected—we solve this issue by using cameras from many viewpoints which share penetration information. We now explain our two passes in more detail.

In our *intra-view* pass, we perform pixel-wise analysis on our rasterized maps in order to mark pixels (and their corresponding triangles) as penetrating. At the i^{th} iteration of our method, we compare the top depth peel against the i^{th} depth peel. For each viewpoint, we identify pixels in the depth peels that 1) face that camera, 2) have an incorrect garment order, and 3) are within a depth threshold of each other. If such a pixel is identified, we mark the corresponding triangle as penetrating, and aim to remove this triangle in subsequent *inter-view* passes.

In our *inter-view* pass, which we illustrate in Fig. 4, we propagate identified penetrating triangles across all viewpoints. In particular, we aim to identify any pixels in the top depth peel corresponding to penetrating triangles, across all viewpoints. In order to do this efficiently, we re-rasterize the identified penetrating triangles and use them as a candidate mask. Finally, if a pixel in this candidate mask is determined to correspond to a penetrating triangle in the top depth peel, we update the top depth peel's pixel's data with that of the corresponding pixel in the i^{th} depth peeling layer.

We iterate these two passes for all depth-peeling layers. Multiple iterations are required to solve complex layering orderings, such as a body with three garments, or folds from physics-simulated garments. After all iterations are completed, we render the Gaussian particles that are skinned to triangles visible in the resulting top depth-peel ID map. To counter any instability due to the low resolution of these rasterization cameras, we add a simple history filter—we only render Gaussian particles that are skinned to triangles visible from any of the last m frames. Our method is summarized in Alg. 1.

3.2 Dual Gaussian/Mesh Reconstruction

While some methods use camera arrays for capturing humans in motion [35, 74], we opt for an approach that is more accessible to the average user. We dress a mannequin in a single garment and capture a short monocular video showing the garment from many angles under a neutral pose using a cell phone camera. We sample 200-300 frames from this video and segment the garment from the scene using Segment Anything [24, 54], placing the garment on a black background.

We perform the dual reconstruction of the 3D Gaussian splats and the simulation mesh by first optimizing the Gaussians, fitting a template mesh, and then reoptimizing the Gaussians. We optimize an initial set of Gaussians, with two regularization losses in addition

Algorithm 1 Triangle Culling Algorithm. Here, *id*, *depth*, and *normal* are rasterized maps across all cameras, with subscripts indexing the depth peeling layer.

procedure TriangleCulling($id, depth, normal$)
  $bad \leftarrow \emptyset$
  for $i = 1, \ldots,$ #depthpeels **do**
    // Intra-view pass
    $id\_mask \leftarrow id_i > id_0$
    $d\_mask \leftarrow d_0 - d_i < \varepsilon_d$
    $n\_mask \leftarrow n_i.z > \varepsilon_n$
    $mask \leftarrow id\_mask \ \& \ d\_mask \ \& \ n\_mask$
    $bad \leftarrow bad \cup id_0[mask]$
    // Inter-view pass
    $candidate \leftarrow$ **Rasterize**$(V, F[bad])$
    $replace \leftarrow id\_mask[candidate]$ **in** bad
    $id_0[candidate][replace] \leftarrow id_i[candidate][replace]$
    $d_0[candidate][replace] \leftarrow d_i[candidate][replace]$
    $n_0[candidate][replace] \leftarrow n_i[candidate][replace]$
  return id_0

to traditional $L1$ and SSIM losses. Namely, loss on the largest and smallest scaling parameters per Gaussian i, $||\max(S_i) - \min(S_i)||_2$, and difference between the scaling parameters and the median Gaussian size, $||S_i - \text{med}(S)||_2$. Together, these encourage uniformly sized Gaussians that are roughly spherical. In practice, these losses also encourage Gaussians to be roughly surface-aligned.

Next, we fit a template garment mesh to the Gaussians. We use a cropped portion of SMPL-X as a template and fit using an Adam optimizer [23], though other reconstruction methods can also be used. We include losses on Chamfer distance, deviation from median triangle area, and angles deviating from equilateral, $(\cos(\theta) - 0.5)^2$ for each triangle angle θ. Finally, we employ losses on triangle normal consistency and median edge length from Pytorch3D [55]. These losses encourage garments to fit the Gaussians and to be simulation-ready.

Finally, we reoptimize a new set of Gaussians with mesh-visibility augmentation. In this second optimization, we seed Gaussians on the garment mesh surface using Poisson-disc sampling [5]. We include losses on the distance of a Gaussian from the simulation mesh surface, opacity, and size. Notably, we do not enforce Gaussians to be surface aligned so as to capture volumetric details, as in our fur vest example.

An important step we introduce in optimizing our Gaussians is that of conditional rendering. During optimization, we render a Gaussian only if its nearest triangle on the fitted simulation mesh is visible from the current optimization camera. By doing so, Gaussians are forced to form a full-opacity cover for each simulation triangle. This aids in cases where a garment is uniform in color, as the front of the garment may otherwise be at risk of relying on Gaussians corresponding to the back of the garment, and vice-versa. Moreover, including this technique in Gaussian optimization mimics our method of layer resolution in our online method. After the optimization passes, we perform minor semi-automated cleanup of the generated Gaussians—filtering background-color splats and removing remaining unwanted Gaussians manually in Blender.

3.3 Transformation of Gaussians

In our visualizer, we transform the Gaussians by skinning them to the simulation meshes. Unlike previous work [74], we do not *explicitly* construct a tetrahedral cage around the mesh. Instead, we compute the deformation gradient F for each triangle via an *implicitly* defined tetrahedron [64]. The centers of Gaussian particles can then be updated as $X_0 + F\vec{r}$, where $\vec{r}$ is the rest pose offset from triangle vertex X_0. We additionally use the rotational component of F, obtained through singular value decomposition, to compute a transformed view direction for spherical harmonics of Gaussians.

3.4 Implementation Details

We simulate our cloths in NVIDIA Warp [38], on an existing XPBD cloth simulator [39, 42]. The simulation meshes contain 1000-2500 vertices and 2000-5000 triangles. We use a time step of 1/40s, with 30 substeps per time step. At each time step, we recompute the SMPL-X joint locations, and interpolate these over substeps. In order to handle collisions against the body, we attach collision capsules between SMPL-X skeleton joint locations. We find that using capsules provides a good balance between efficiency, accuracy, and stability, which are crucial for real-time simulations. We handle cloth-cloth collisions through a standard XPBD constraint, with one-sided springs between outer-garment cloth particles and their nearest inner-garment triangles. We apply a simple friction model in either collision case. We evaluate our collision constraints every substep. Critically, we prioritize efficiency and stability over collision-free results—errors that our screenspace layer resolution scheme is able to handle. We mark a subset of SMPL-X triangles, including the head, hands, and legs, as always visible, in order to focus camera placement on garments—close camera placement may not include these regions fully, and high-density regions such as the face are not suited for our method focused on low resolution rasterization.

We implement the full pipeline in Pytorch [48], building off of previous Gaussian rasterization works [22, 74]. We use NVDiffRast [25] for rasterization. By using both NVDiffRast for rasterization and Warp for simulation, we keep our entire pipeline GPU-based. We use 5 depth peeling layers at runtime, with a resolution of 512×512, across 17 camera viewpoints. We position cameras at rotations of $\frac{\pi}{8}$, $\frac{\pi}{4}$, $\frac{\pi}{2}$, and $\frac{3\pi}{4}$ in either direction about the Y and X axes (Y up), in addition to the head-on viewpoint. In our intra-view pass, we use a depth threshold of 5 cm, and only process triangles with a normal vector within 70 degrees of facing the camera.

4 RESULTS

Video data was recorded on Samsung S10 and S24 phones. Garments used range in size from medium to large, including multiple tee shirts, a running singlet, a hoodie, and a fur vest. The garments were placed on a consumer-grade mannequin with men's size L. We evaluate our implementation on a Windows 10 desktop machine with an Intel Core i9-14900KF @ 3200MHz, with 64GB RAM, 2TB SSD, and a single RTX 4090 GPU with 24GB of RAM. We note that our method relies only on *static* capture of garments rather than expensive hardware such as camera arrays or multi-view video, making comparison to competing techniques infeasible. To our

knowledge, no other method provides realtime interactive try-on of Gaussian garments under these settings.

4.1 Performance

When playing back animations from the AMASS dataset [40], our online method is able to achieve interactive framerates. With spherical harmonics disabled, we attain framerates of 20-24 fps. With spherical harmonics enabled, our performance is slightly reduced, as we must perform singular value decomposition for each simulation triangle, but we are able to maintain framerates of 18-21 fps. We report more detailed numbers in Table 1. While not as fast as the naïve method that simply displays all Gaussians, our method produces much better visual results, while maintaining interactive framerates. Notice that our performance relative to naïve increases when spherical harmonics are on, as we are able to evaluate the singular value decomposition for only seen triangles, whereas the naïve method must for all triangles. In Table 2, we provide information about the size of each garment. Our implementation is written in GPU-oriented Python, utilizing PyTorch and Warp—further performance gains may be achieved through reimplementation in C++ and CUDA. Our mesh-visibility augmented Gaussian optimization takes 20-30 minutes on our machine per garment, depending on garment complexity. For reference, standard 3DGS take approximately 5-8 minutes to optimize.

Figure 5: We present many examples of different garment combinations with 2-3 layers, in various layering orders and poses.

Figure 6: By using physics simulation, our method can achieve poses that are impossible for skinning-based approaches, such as this fur vest sliding off.

4.2 Qualitative Evaluation

In Fig. 5, we provide visual results of our method for various combinations of 7 different garments. In addition to the layer of garments, we also visualize Gaussian particles of the SMPL-X model itself. We provide examples of our method of a user wearing 1-to-3 garments, in various layering orders. We test our method on a variety of mocap sequences found in the AMASS database [40]. In Fig. 6, we show a result where the outer fur vest slides off of the body, a scenario that cannot be handled by skinning-based approaches [74]. We refer the reader to our supplemental material for videos of our method in action, displayed at their real-time framerate. Our method is able to consistently output high-quality renderings of physically simulated clothed avatars at interactive framerates. Even when large penetrations are present, as in Fig. 7, we still can recover high-quality visual results.

4.3 Interactive Try-On

In our video, we demonstrate our method through interactive virtual try-on of garments. Our simple Python implementation produces realistic try-ons of multilayer garments from webcam video. We use the ExPose [8] implementation from MMHuman3D [41] on a second thread for tracking, and SMPL-X is updated with the new joint information. The tracking performance is not perfect, which results in occlusion artifacts, but we note that this is not the focus of our work. We segment visible garments in a simple process by using a black background as well as a black SMPL body to handle occlusions, and then composite resulting visible garments onto the

Table 1: We evaluate the performance of our method against a naïve method of displaying all Gaussians. We provide FPS for our method with and without spherical harmonics (SHS) enabled, and we evaluate on the "Bad Books" (B), "Pigeons" (P), and "Singlet" (S) garments in this order.

Garms.	SHS	Method	FPS
B	Y	Ours	21
B	Y	Naïve	31
B	N	Ours	24
B	N	Naïve	45
B,P	Y	Ours	19
B,P	Y	Naïve	26
B,P	N	Ours	21
B,P	N	Naïve	39
B,P,S	Y	Ours	18
B,P,S	Y	Naïve	24
B,P,S	N	Ours	20
B,P,S	N	Naïve	34

Table 2: Details on the size of our learned Gaussians (G) and their corresponding meshes. We refer to each garment using a simple visual identifier: "Bad Books" and "Migrant" refer to the garments with the corresponding text visible, and "Pigeons" refers to the shirt with 5 pigeons.

Garment	#G	#V	#F
SMPL-X	1.2M	10k	20k
"Bad Books"	250k	2k	3.5k
"Migrant"	350k	2k	3.5k
"Pigeons"	500k	2k	3.5k
Plaid	250k	2k	4k
Hoodie	400k	2.5k	5k
Singlet	200k	1k	2k
Fur	400k	1k	2k

Figure 7: Even when simulation results in large penetrations (left), we are often still able to recover high-quality visual results (center). A naïve approach results in large penetrations, in addition to other artifacts (right).

user's live video. Our interactive try-on runs at 12-16 fps—thread synchronization is a bottleneck, as compared to our examples via mocap sequences. We use a 1080P Logitech Brio 100 webcam in testing.

4.4 Ablation Study

We compare against alternative approaches for rendering multilayer Gaussian garments. In particular, we consider alternative methods that attempt to fix an existing cloth simulation state without affecting physics simulation, while maintaining real-time performance. First, we discuss the ablation results displayed in Fig. 10. In the *naïve* approach (B), we simply display all Gaussians skinned to all garments. In *Single View Processes* (C), we perform our method, but only use a single virtual camera, aligned with the user-chosen viewpoint. In the *Single View Visible* method (D), we again rasterize single virtual viewpoint, and display Gaussians skinned to visible triangles with no further processing. We find that our proposed method (A) generates results with far fewer artifacts as compared to these ablations. In the naïve approach (B) and unprocessed triangle visibility method (D), large penetrations can easily occur, as nothing is done to resolve these states. In our method with a single viewpoint (C), errors remain—side penetrations, similar to those shown in Fig. 3, remain unresolved.

In Fig. 8, we consider naïve methods in which one updates the simulation mesh to a penetration-free state per-frame before skinning and displaying all Gaussian particles. As in our method, these comparisons do not affect the physics simulation, only updating vertices for rendering purposes. In both (A) and (B), we assign layered garments to different level sets of the SMPL-X body. In (A), we project vertices that are inside their assigned level set to the surface of that level set. In (B), we project all vertices to be exactly on their assigned level sets. In either case, issues remain, and our method (C) yields better results. When level set thresholds for (A) and (B) are tuned to prevent visible artifacts on the front of the garment, large penetrations and instabilities begin to occur between the arm and side of the body.

In Fig. 9, we provide results with cloth-cloth collisions *turned off* (A), where penetrations may be even greater. In this case, our method (C) still resolves most collisions, though some poses may be unrecoverable if garments deform in very different ways.

Figure 8: We compare our method (C) against two methods of quickly adjusting triangles in order to try and avoid penetrations. In both (A) and (B), we assign layered garments to different level sets of the body. In (A), if a vertex is inside its level set, we project it to this level set before rendering. Large visible penetrations begin to occur when the arm is down, while small penetrations still remain on the front of the body. In (B), we project all vertices to be exactly on their level sets after running the physics step. Again, issues occur when garments hang closer to the arm than the body.

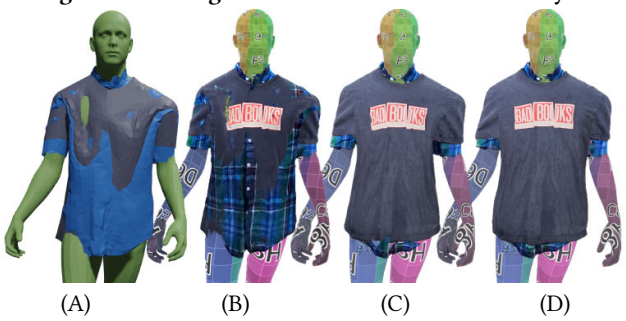

Figure 9: Even with cloth-cloth collision disabled (A,B,C) in our mesh simulation (A), our method (C) is still able to improve over the naïve method (B), so long as the garment depth difference is within our tolerance. We provide the same scene with our method and cloth-cloth collisions enabled for reference (D). Note that the outer garment is pushed out by the inner garment.

In Fig. 11, we provide a comparison of Gaussians optimized with and without our mesh-visibility augmentation described in §3.2. Without using our augmentation, Gaussian optimization often results in garments that rely on Gaussians skinned to both the front and back of a garment. While these results often appear realistic in a traditional Gaussian renderer, they appear patchy in our system due to our online method of rendering Gaussians only if their simulation triangle is visible. Our augmentation technique removes this issue. We similarly provide a comparison of Gaussians before and after applying our entire second optimization pass in Fig. 12. While alternate methods, such as GS2Mesh [67], may be used in place of our first optimization pass, we similarly find that the resulting Gaussians are not simulation ready, and transparency and patchiness issues remain without our second optimization pass.

(A) Naïve (B) Single View Process (C) Single View Visible (D) Ours

Figure 10: We compare our method (D) against a naïve approach rendering all Gaussians (A), our method only using a single viewpoint (B), and a method using all visible triangles with no processing from a single viewpoint (C). In the naïve approach, even when simulation garments are not in collision, Gaussians may still penetrate. In our method with only a single viewpoint (B), most visual artifacts are resolved, but artifacts remain where normal vector tolerances fail. In (C), Gaussians are displayed incorrectly due to penetrating simulation garments.

Figure 11: We compare Gaussians optimized with (right) and without (left) our mesh-visibility augmentation in our second optimization pass. Without our mesh-visibility augmentation, Gaussian optimization may result in front-facing portions of a garment relying on Gaussians on the back of the garment. This results in spotty final renders when only Gaussians on front faces are rendered—notice how individual Gaussians are distinguishable as blotches under close inspection. This is further exacerbated under deformations.

Figure 12: We compare Gaussians optimized with our second optimization (right) and with only our first optimization (left). By including our second optimization pass, we both ensure Gaussians are well aligned to the mesh surface, and form a full opacity cover of the mesh. In the circled section, Gaussians fail to form a full cover—the black background used in our visualizer is visible through the Gaussians.

Finally, in Fig. 14 we provide a comparison of our chosen number of cameras ($n = 17$) against other evenly spaced camera designs. We find that with fewer cameras, many artifacts are present, and there is little benefit to introducing more.

Figure 13: When simulation garments have large penetrations (Yellow), restricted visibility (Blue), or large boundary triangles (Red), our method may leave minor artifacts while still greatly improving against naïve methods.

5 CONCLUSION & FUTURE WORK

We presented a novel interactive system for capturing and driving multilayer Gaussian garments for dressing 3D human avatars using low-cost hardware. Our capture method allows us to optimize the Gaussian garments from static poses using a single phone camera. Our system then combines physical simulation and Gaussian garments with a screenspace layer resolution method to synthesize both realistic and interactive 3D avatar dressing results. By using physically based simulation, our system is able to achieve dynamic multilayer try-on, allowing for try-on settings that are impossible for similar methods based on skinning.

Our work is not without limitations. In our method, we discard an entire triangle if it is found to be violating, which may cause artifacts if triangles are large (Fig. 13, right). On the converse, simulation triangles must be large enough to be seen by our rasterization step, or else they may not display their Gaussians. An approach that accounts for barycentric coordinates could remedy this dichotomy, but we leave this as future work. Similarly, if an inner garment triangle is partially obscured but not intersecting, its Gaussians may incorrectly appear on top of the outer garment, depending on opacity and distance to the triangle. Enforcing surface-aligned Gaussians may improve this case, at the cost of rendering quality.

(A) 5 Cameras (B) 9 Cameras (C) 9 Cameras (Horizontal) (D) 17 Cameras (Ours) (E) 33 Cameras (Diagonals)

Figure 14: We display the effect of the number of virtual cameras used in our pipeline. In addition to the "head on" view, we place cameras at X and Y rotations of (A): $\frac{\pi}{2}$; (B): $\frac{\pi}{4}, \frac{3\pi}{4}$; (D): $\frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$. In (C) we used the same angles as (D) but omit vertical rotations. In (E) we use the same angles as (D) for X and Y axis rotations, as well as the diagonals between these. The results generated with fewer cameras than ours (D) have more artifacts. On the other hand, there is no noticeable benefit using more cameras.

Also, since we do not have any information about the inside of the clothing, our system shows artifacts when the clothing flips over (*e.g.,* vest opening, as seen under the armpit in Fig. 6). We assume users specify a garment order, which may not change during simulation, though restarting with different garment orders is very fast. Although we rotate the spherical harmonics at runtime, some details, such as fine wrinkles, are baked into our splats, as we use a static capture setup as opposed to an expensive camera array, and we avoid related expensive runtime techniques in order to remain interactive. Finally, extreme penetrations can cause our depth thresholding to fail (Fig. 13, left). For these reasons, we expect the physics simulator to produce plausible results, though they may still be far from perfect, especially with self-collisions.

Human avatars are not the focus of this work, as we focus on capturing static garments (which are then driven dynamically). Therefore, we optimize Gaussians of SMPL-X itself as a baseline character. However, our method is compatible with 3DGS avatars produced by other methods [20, 47, 74], provided that Gaussian particles are skinned near the surface of SMPL-X in order to form a triangle correspondence. Finally, the quality of our webcam-based interactive try-on depends on the quality of tracking. Occlusion artifacts will be reduced if better tracking methods are available.

There are many interesting directions in which our work can be extended. One could consider optimizing the locations of the cameras used for triangle culling, such as skinning them to garments, or SMPL-X itself. Another direction would be to incorporate lighting-aware or ambient-occlusion-aware Gaussians in order to further improve realism. While our design is focused on correcting collisions visually, one could extend our approach to update simulation objects' positions, or extend to more general Gaussian mesh settings.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their thoughtful comments. This work was sponsored in part by the National Science Foundation (#2313076). R. Zesch was supported by the Physical AI Design NSF International Research Experiences for Students (IRES #2433313) project.

REFERENCES

[1] George Baciu and Wingo Sai-Keung Wong. 2004. Image-based collision detection. *Integrated image and graphics technologies* (2004), 75–94.
[2] David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Annual Conference Series (Proc. SIGGRAPH)*. 43–54.
[3] David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling cloth. *ACM Trans. Graph.* 22, 3 (jul 2003), 862–870.
[4] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2022. Neural cloth simulation. *ACM Trans. Graph. (TOG)* 41, 6 (2022), 1–14.
[5] Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches* 10, 1 (2007), 1.
[6] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. 2005. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses*. 3–es.
[7] Thomas Buffet, Damien Rohmer, Loïc Barthe, Laurence Boissieux, and Marie-Paule Cani. 2019. Implicit untangling: a robust solution for modeling layered clothing. *ACM Trans. Graph.* 38, 4, Article 120 (Jul 2019), 12 pages.
[8] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J. Black. 2020. Monocular Expressive Body Regression through Body-Driven Attention. In *European Conference on Computer Vision (ECCV)*. 20–40.
[9] Aiyu Cui, Daniel McKee, and Svetlana Lazebnik. 2021. Dressing in order: Recurrent person image generation for pose transfer, virtual try-on and outfit editing. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14638–14647.
[10] Luca De Luigi, Ren Li, Benoit Guillard, Mathieu Salzmann, and Pascal Fua. 2023. DrapeNet: Garment Generation and Self-Supervised Draping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[11] Christer Ericson. 2004. *Real-time collision detection*. Crc Press.
[12] François Faure, Jérémie Allard, Florent Falipou, and Sébastien Barbier. 2008. Image-based collision detection and response between arbitrary volumetric objects. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* Eurographics Association, 155–162.
[13] Yao Feng, Weiyang Liu, Timo Bolkart, Jinlong Yang, Marc Pollefeys, and Michael J. Black. 2023. Learning Disentangled Avatars with Hybrid 3D Representations. *arXiv* (2023).
[14] Yao Feng, Jinlong Yang, Marc Pollefeys, Michael J Black, and Timo Bolkart. 2022. Capturing and animation of body and clothing from monocular video. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.
[15] Naga K. Govindaraju, Ming C. Lin, and Dinesh Manocha. 2004. Fast and reliable collision culling using graphics hardware. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Hong Kong) *(VRST '04)*. ACM, New

York, NY, USA, 2–9.
[16] Naga K Govindaraju, Ming C Lin, and Dinesh Manocha. 2005. Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE, 59–66.
[17] Naga K Govindaraju, Stephane Redon, Ming C Lin, and Dinesh Manocha. 2003. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 25–32.
[18] 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* (Denver, CO, USA) *(SIGGRAPH '24)*. ACM, New York, NY, USA, Article 81, 10 pages.
[19] Antoine Guédon and Vincent Lepetit. 2024. Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering. *ECCV* (2024).
[20] Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. 2023. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. *arXiv preprint arXiv:2312.02134* (2023).
[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. https://doi.org/10.1145/3641519.3657428
[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 1–14.
[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4015–4026.
[25] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020).
[26] Lei Lan, Zixuan Lu, Jingyi Long, Chun Yuan, Xuan Li, Xiaowei He, Huamin Wang, Chenfanfu Jiang, and Yin Yang. 2024. Efficient GPU Cloth Simulation with Non-distance Barriers and Subspace Reuse. arXiv:2403.19272 [cs.GR]
[27] Dohae Lee, Hyun Kang, and In-Kwon Lee. 2023. Clothcombo: Modeling inter-cloth interaction for draping multi-layered clothes. *ACM Trans. Graph.* 42, 6 (2023), 1–13.
[28] François Lehericey, Valérie Gouranton, and Bruno Arnaldi. 2015. GPU ray-traced collision detection for cloth simulation. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology* (Beijing, China) *(VRST '15)*. ACM, New York, NY, USA, 47–50.
[29] Jiahui Lei, Yufu Wang, Georgios Pavlakos, Lingjie Liu, and Kostas Daniilidis. 2023. Gart: Gaussian articulated template models. *arXiv preprint arXiv:2311.16099* (2023).
[30] Kedan Li, Jeffrey Zhang, Shao-Yu Chang, and David Forsyth. 2022. Wearing the Same Outfit in Different Ways–A Controllable Virtual Try-on Method. *arXiv preprint arXiv:2211.16989* (2022).
[31] Mengtian Li, Shengxiang Yao, Zhifeng Xie, Keyu Chen, and Yu-Gang Jiang. 2024. Gaussianbody: Clothed human reconstruction via 3d gaussian splatting. *arXiv preprint arXiv:2401.09720* (2024).
[32] Ren Li, Benoît Guillard, and Pascal Fua. 2024. Isp: Multi-layered garment draping with implicit sewing patterns. *Adv Neural Inf Process Syst* 36 (2024).
[33] Xuan Li, Yu Fang, Lei Lan, Huamin Wang, Yin Yang, Minchen Li, and Chenfanfu Jiang. 2023. Subspace-Preconditioned GPU Projective Dynamics with Contact for Cloth Simulation. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. ACM, New York, NY, USA, Article 1, 12 pages.
[34] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. 2024. Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[35] Siyou Lin, Zhe Li, Zhaoqi Su, Zerong Zheng, Hongwen Zhang, and Yebin Liu. 2024. LayGA: Layered Gaussian Avatars for Animatable Clothing Transfer. In *SIGGRAPH Conference Papers*.
[36] Lijuan Liu, Xiangyu Xu, Zhijie Lin, Jiabin Liang, and Shuicheng Yan. 2023. Towards garment sewing pattern reconstruction from a single image. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–15.
[37] Xinqi Liu, Chenming Wu, Jialun Liu, Xing Liu, Chen Zhao, Haocheng Feng, Errui Ding, and Jingdong Wang. 2024. GVA: Reconstructing Vivid 3D Gaussian Avatars from Monocular Videos. *Arxiv* (2024).
[38] Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. NVIDIA GPU Technology Conference (GTC).
[39] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*. 49–54.
[40] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture As Surface Shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
[41] MMHuman3D. 2021. OpenMMLab 3D Human Parametric Model Toolbox and Benchmark.
[42] Matthias Müller. 2022. Ten Minute Physics. https://www.youtube.com/@TenMinutePhysics
[43] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2015. Air meshes for robust collision handling. *ACM Trans. Graph.* 34, 4, Article 133 (Jul 2015), 9 pages.
[44] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.
[45] Karol Myszkowski, Oleg G Okunev, and Tosiyasu L Kunii. 1995. Fast collision detection between complex solids using rasterizing graphics hardware. *The Visual Computer* 11 (1995), 497–511.
[46] Rahul Narain, Armin Samii, and James F O'brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. (TOG)* 31, 6 (2012), 1–10.
[47] Haokai Pang, Heming Zhu, Adam Kortylewski, Christian Theobalt, and Marc Habermann. 2023. Ash: Animatable gaussian splats for efficient and photoreal human rendering. *arXiv preprint arXiv:2312.05941* (2023).
[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Adv Neural Inf Process Syst*. Curran Associates, Inc., 8024–8035.
[49] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 10975–10985.
[50] Sida Peng, Zhen Xu, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. 2024. Animatable implicit neural representations for creating realistic avatars from videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).
[51] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael Black. 2017. ClothCap: Seamless 4D Clothing Capture and Retargeting. *ACM Trans. Graph., (Proc. SIGGRAPH)* 36, 4 (2017). Two first authors contributed equally.
[52] Sergey Prokudin, Michael J. Black, and Javier Romero. 2021. SMPLpix: Neural Avatars from 3D Human Models. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV 2021)*. IEEE, Piscataway, NJ, 1809–1818.
[53] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. 2024. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20299–20309.
[54] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. 2024. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714* (2024).
[55] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).
[56] Boxiang Rong, Artur Grigorev, Wenbo Wang, Michael J Black, Bernhard Thomaszewski, Christina Tsalicoglou, and Otmar Hilliges. 2024. Gaussian garments: Reconstructing simulation-ready clothing with photorealistic appearance from multi-view video. *arXiv preprint arXiv:2409.08189* (2024).
[57] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J. Black. 2021. SCANimate: Weakly Supervised Learning of Skinned Clothed Avatar Networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)*. IEEE, Piscataway, NJ, 2885–2896.
[58] Igor Santesteban, Miguel A Otaduy, and Dan Casas. 2019. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 355–366.
[59] Igor Santesteban, Miguel A Otaduy, and Dan Casas. 2022. SNUG: Self-Supervised Neural Dynamic Garments. *arXiv preprint arXiv:2204.02219* (2022).
[60] Igor Santesteban, Miguel A. Otaduy, Nils Thuerey, and Dan Casas. 2022. ULNeF: Untangled Layered Neural Fields for Mix-and-Match Virtual Try-On. In *Adv Neural Inf Process Syst*.
[61] Yidi Shao, Chen Change Loy, and Bo Dai. 2023. Towards Multi-Layered 3D Garments Animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14361–14370.
[62] Mikio Shinya and Marie-Claire Forgue. 1991. Interference detection through rasterization. *The Journal of Visualization and Computer Animation* 2, 4 (1991), 132–134.
[63] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. 2021. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Adv Neural Inf Process Syst* 34 (2021), 12278–12291.

[64] Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. (TOG)* 23, 3 (2004), 399–405.
[65] Pascal Volino and Nadia Magnenat-Thalmann. 2006. Resolving surface collisions through intersection contour minimization. *ACM Trans. Graph. (TOG)* 25, 3 (2006), 1154–1159.
[66] Bin Wang, François Faure, and Dinesh K Pai. 2012. Adaptive image-based intersection volume. *ACM Trans. Graph. (TOG)* 31, 4 (2012), 1–9.
[67] Yaniv Wolf, Amit Bracha, and Ron Kimmel. 2024. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views. In *ECCV 2024 Workshop on Wild 3D: 3D Modeling, Reconstruction, and Generation in the Wild*.
[68] Botao Wu, Zhendong Wang, and Huamin Wang. 2022. A GPU-Based Multilevel Additive Schwarz Preconditioner for Cloth and Deformable Body Simulation. *ACM Trans. Graph. (SIGGRAPH)* 41, 4, Article 63 (jul 2022), 14 pages.
[69] Longhua Wu, Botao Wu, Yin Yang, and Huamin Wang. 2020. A safe and fast repulsion method for GPU-based cloth self collisions. *ACM Trans. Graph. (TOG)* 40, 1 (2020), 1–18.
[70] Donglai Xiang, Timur Bagautdinov, Tuur Stuyck, Fabian Prada, Javier Romero, Weipeng Xu, Shunsuke Saito, Jingfan Guo, Breannan Smith, Takaaki Shiratori, et al. 2022. Dressing avatars: Deep photorealistic appearance for physically simulated clothing. *ACM Trans. Graph. (TOG)* 41, 6 (2022), 1–15.
[71] Donglai Xiang, Fabian Prada, Timur Bagautdinov, Weipeng Xu, Yuan Dong, He Wen, Jessica Hodgins, and Chenglei Wu. 2021. Modeling clothing as a separate layer for an animatable human avatar. *ACM Trans. Graph. (TOG)* 40, 6 (2021), 1–15.
[72] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2023. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198* (2023).
[73] Luyang Zhu, Yingwei Li, Nan Liu, Hao Peng, Dawei Yang, and Ira Kemelmacher-Shlizerman. 2024. M&M VTO: Multi-Garment Virtual Try-On and Editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1346–1356.
[74] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. 2023. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581* (2023).