Towards Consistent Text-Driven Multi-View Fashion Editing

César Díaz Blanco
University of Tübingen
Tübingen, Germany
cesar.diaz-blanco@student.uni-tuebingen.de

Abstract

Text-based fashion editing methods have advanced in recent years with the advent of diffusion models. While successful, these are limited to edit a single view and using the same prompt on images from different perspectives will not generate the same edits. In this work, we disentangle the task of editing the clothing of a person while keeping it consistent among views by: (1) editing the front-facing image; (2) editing the remaining views using the generated image from step (1). In this manner, we condition subsequent generations not only on the ambiguous text prompt but also on the initial edited image which contains rich cues of the generated clothing, the person's identity and pose. This method allows edits on an arbitrary number views and is model-agnostic enabling its use with current and future state-of-the-art methods. Extensive experiments on a setup which takes in 20 images and uses `FLUX.1 Kontext [dev]` *for the first step and* `FLUX.1 Fill [dev]` *for the second step show that our method is able to generate consistent edits while preserving the person's pose and identity.*
Code is released in this repository.

1. Introduction

Latent diffusion-based [17] models have revolutionized image generation [12] and image editing models [14], [9]. Focused in the domain of fashion, image editing models can realistically modify garments on a person given an image and a text instruction. However, these models are fundamentally limited to operating on a single view and are known to struggle with precise edits altering parts of an image which shouldn't have changed [14].

Added to this, if the same text prompt is applied independently to different views of the same person the results will be inconsistent. This lack of 3D consistency is a major barrier to applying these models to 3D-aware applications, such as 3D clothing generation while preserving human pose and achieving precise and consistent 3D fashion editing on existing garments.

A related task, Virtual Try-On (VTON) seeks to transfer a garment onto a person has shown advances with methods like CatVTON [1] successfully transferring garments onto images of persons in in-the-wild scenarios. Related to our goal, MV-VTON [21] focuses not only on front-view virtual try-on but also on back-view achieving multi-view consistency. Video VTON has also emerged as the task of editing-in clothing in a consistent manner for an image sequence showing the same person in different poses; for this task, ClothFormer [5] achieves good results for front and back views, but struggles in intermediate views.

Nevertheless, VTON focuses on adding a garment from an image. The complex task of modifying or removing an existing garment in a multi-view consistent manner—especially when it requires plausibly hallucinating occluded parts, like the shirt underneath a removed jacket—remains a significant and unsolved challenge. Furthermore, the field lacks established benchmarks and evaluation metrics to specifically assess such fashion edits.

In this work, we propose a model-agnostic generation strategy that disentangles the task of applying edits to an arbitrary number of multi-view images given a text instruction. Our method consists of two stages:

1. Root-Anchor Editing: We select the image closest to a front-facing image (the "root-anchor") and apply the text-based edit using an editing model, in our case `FLUX.1 Kontext [dev]` [9]. This initial edit captures essential details such as the garment length and style if applicable, skin color, and the appearance of previously occluded garments.
2. Guided Inpainting: We propagate this edit to all other views using an inpainting model, `FLUX.1 Fill [dev]` [8] in our case. We condition the inpainting of each new view not only on the text prompt but also on the already-edited root-anchor image. This provides strong visual guidance, ensuring the edits remain consistent.

Our main contributions are:

1. A novel two-stage, model-agnostic strategy for text-driven, multi-view consistent editing.
2. A method which successfully removes garments in a

multi-view consistent manner, as tested on the 4D-DRESS [22] dataset using 20 views for each scan.

3. An evaluation suite for the garment-removal task to test 3D consistency, success rate and quality of removal as judged by a VLM, and consistency before and after the generation on regions which shouldn't have been altered.

2. Related Work

Our work sits at the intersection of text-based fashion editing, virtual try-on, and multi-view consistent generation.

2.1. Text-Based Virtual Try-On (VTON)

Virtual Try-On (VTON) aims to realistically place a new garment onto a target person. Traditional VTON methods often relied on 3D meshes or complex warping networks. More recently, diffusion-based models have shown impressive results and now dominate the field. A notable example is CatVTON [1], which demonstrated that a simple spatial concatenation of the garment and human images gives high-fidelity results while simplifying the architecture and training when compared to other VTON methods.

Recent extensions to the task include ensuring multi-view consistency with works like MV-VTON [21] which adds a back-view of the garment to edit-in. MV-VTON does self and cross-attention between the latent features of the human and garment images to interpolate between the two input garment images. ClothFormer [5] tackles a similar problem: fit a target garment to a person with spatio-temporal consistency. Their method is composed of stages to obtain clothing segmentation maps, warped clothing, optical flow maps, and finally a transformer module which takes in the warped clothing and the original video sequence and outputs the video of the person wearing the target garment.

Our work draws inspiration from CatVTON's concatenation-based conditioning. However, instead of using a clean image of a garment as the condition, our method uses a previously generated image as the visual guide. Opposed to the mentioned VTON methods which rely on training, our work aims to be as simple as possible and focuses on leveraging state-of-the-art editing and inpainting models. Another reason for our approach's focus on inference is the lack of datasets which provide an image sequence of humans wearing different clothing while preserving the identity and pose of the person and camera pose.

Lastly, VTON methods report metrics which require ground truth data or a ground truth distribution to compare to. CatVTON and MV-VTON report SSIM, FID, KID, and LPIPS metrics on paired data as well as FID and KID on unpaired data. For our task, there is no ground truth to compare to or the distribution of generated images is considerably different. For instance, when removing a shirt to show the torso of a man, the resulting image is not similar to any of the 4D-DRESS [22] scans which show clothed humans. Thus, in our work, we prompt a VLM to assess removal quality and compute the PSNR and SSIM metrics on regions which shouldn't have been edited.

2.2. Multi-View Consistent Generation

Generating 3D-consistent images from text prompts has been a major goal in recent computer vision research. DreamFusion [16] introduced Score Distillation Sampling (SDS) to optimize a 3D representation, in their case NeRF [11], using a 2D diffusion model as a prior.

More recent methods, such as MV-Adapter [4], aim to adapt existing 2D diffusion models to generate consistent multi-view images directly. MV-Adapter introduces a lightweight module that processes multi-view noise latents, enabling a pre-trained model to produce consistent images from different camera angles.

While powerful for generating objects from scratch, these methods are not designed for the task of editing a real, high-fidelity person. As our experiments show in Fig. 5, methods like MV-Adapter struggle to preserve the identity and pose of the person, as the underlying VAE fails to capture high-frequency details. Furthermore, the output views only considers six different camera poses; a problem which is more generalized to multi-view generation models which have limited camera control.

Given the myriad choices when designing multi-view consistent generation methods, until recent, there were no metrics tailored to compare methods which generate images at different camera poses with different styles and image quality. We found MVGBench [24] to be a helpful ally as it provides an evaluation suite that fairly compares different multi-view generation models for object reconstruction.

3. Background

Our method is model-agnostic, but its success relies on leveraging powerful, pre-trained image models for editing and inpainting. For our experiments, we selected models from the `FLUX.1` family due to their state-of-the-art performance and open-source availability.

3.1. FLUX.1

`FLUX.1 Kontext [dev]` [9] and `FLUX.1 Fill [dev]` [8] are rectified flow-matching-based models. In general, `FLUX.1` models use a convolutional autoencoder to encode images into latent features with 16 channels and decreased spatial resolution. Given the increased number of channels, `FLUX.1`'s autoencoder achieves a 31.1 mean PSNR reconstruction quality on 4096 ImageNet [2] images, outperforming SD3's [3] quality of 29.6 mean PSNR and SDXL's [15] 25.9 mean PSNR.

`FLUX.1` models initially encode text and image tokens separately through double stream diffusion transformer blocks obtaining strong features which are mixed with attention over the concatenated tokens. Later on, single stream diffusion transformer blocks further process the concatenated tokens. The latter single stream blocks fuse the attention and MLP layers to perform large matrix-vector multiplications which are more efficient than many smaller matrix-vector multiplications. Finally, the image tokens are decoded to get the generated image. For both of the models we use there is an input image whose tokens are encoded by time equal to 0 and pixel coordinates.

`FLUX.1 Kontext [dev]` is designed for in-context image generation and editing. Given an image and a natural-language instruction, it synthesizes content that seamlessly fits into the new scene. It excels at understanding a text instruction in relation to an input image. The model is trained with a rectified flow–matching loss:

$$\mathcal{L}_\theta = \mathbb{E}_{t\sim p(t),\, x,y,c}\big[\, \|v_\theta(z_t, t, y, c) - (\varepsilon - x)\|_2^2 \big], \quad (1)$$

where x is the target image, y is a context image (or $\varnothing$), c is a natural-language instruction, z_t is the linearly interpolated latent between x and noise $\varepsilon \sim \mathcal{N}(0, 1)$, and $p(t; \mu, \sigma = 1.0)$ with μ depending on the image resolution.

We leverage `FLUX.1 Kontext [dev]` for the first stage of our pipeline: generating the high-quality 'root-anchor' image that serves as the guidance for all subsequent edits.

`FLUX.1 Fill [dev]` is an inpainting model sharing the architecture of the `FLUX.1` family and focused on editing images based on a text description and a mask. While the training objective is not known as there is no technical report on this model, in the inference code the transformer module takes in the masked image, the mask, and the text prompt. That is, for tasks like ours where the masked regions could be helpful for guiding the inpainting (to preserve inner garments or human pose), these are not available during the denoising process.

In our second stage, we use `FLUX.1 Fill` to inpaint the garment regions in the non-anchor views guided by the concatenated root-anchor image which provides a strong visual condition to ensure consistency.

3.2. MV-Adapter

As a point of comparison, we evaluate against MV-Adapter [4]. MV-Adapter is a set of plug-and-play modules which add multi-view attention layers to the layers of a diffusion model's denoising U-Net [18]. They also introduce a condition guider which encodes camera information. In our case, since we use their image-to-image version of the model, additional cross-attention layers enable image-conditioned generation.

While designed for generation-from-scratch, we adapt it for our editing task by providing the root-anchor image as a condition. However, as noted in our quantitative results in Section 5.5, we found that their method using SDXL struggles to encode the fine-grained details of a real person, leading to significant degradation of identity and pose as seen in Fig. 5. MV-Adapter is a powerful addition for U-Net based generation-models, however, recent state-of-the-art image generation models opt for a transformer-based denoising model as is the case for the `FLUX.1` family.

4. Methods

Given N input images $\{\boldsymbol{I}^i\}_{i=1}^N, \boldsymbol{I}^i \in \mathbb{R}^{H\times W\times 3}$ showing the same person from different perspectives in a static setting, their corresponding N segmentation maps $\{\boldsymbol{S}^i\}_{i=1}^N, \boldsymbol{S}^i \in \mathbb{R}^{H\times W\times 3}$ following the 4D-DRESS [22] labels and palette, the text instruction P_{edit} to edit garments, the text prompt P_{inpaint}, and the set of garments $\boldsymbol{l}$ that are to be edited following 4D-DRESS' categorization, our goal is to apply the edit specified by P_{edit} on all N images while preserving the person's identity, pose, and garments other than those in $\boldsymbol{l}$, thus obtaining the edited images $\{\boldsymbol{G}^i\}_{i=1}^N, \boldsymbol{G}^i \in \mathbb{R}^{H\times W\times 3}$. The text prompt P_{inpaint} is a substring of the text instruction P_{edit}, the only difference is that P_{edit} describes how the model should edit the image while P_{inpaint} describes what is inpainted. For instance, for P_{edit} "remove the jacket to show the yellow shirt underneath", P_{inpaint} would be "yellow shirt".

As shown in Fig. 1, our method consists of two stages. First, as explained in section 4.1, we select the front-facing image and edit it with the instruction P_{edit} to get the root-anchor image which will guide edits in other views. Second, we propagate the edits to all other views with an inpainting model; an in-depth explanation is in section 4.2. In section 4.3 we introduce two algorithms to determine the order of generation and which views will guide the generation.

In the first stage, any model capable of editing images based on text instructions can be used; we choose `FLUX.1 Kontext [dev]` [9] for this task since it achieves impressive results and its weights have been open-sourced. In the second stage, any inpainting model can be used; again we opt for a Black Forest Labs alternative: `FLUX.1 Fill [dev]` [8]. In general, inpainting models require a mask specifying the region to be inpainted. We compute these from the labels $\boldsymbol{l}$ which specify the garments to be inpainted and from the segmentation maps $\{\boldsymbol{S}^i\}_{i=1}^N$.

During evaluation to test consistency, we fit a 3DGS [6] representation to the output $\{\boldsymbol{G}^i\}_{i=1}^N$ images which requires the camera poses. Since our method preserves the camera pose, we use the projection matrices $\{\mathbf{P}_i\}_{i=1}^N, \mathbf{P}_i \in \mathbb{R}^{3\times 4}$ corresponding to the input images $\{\boldsymbol{I}^i\}_{i=1}^N$.

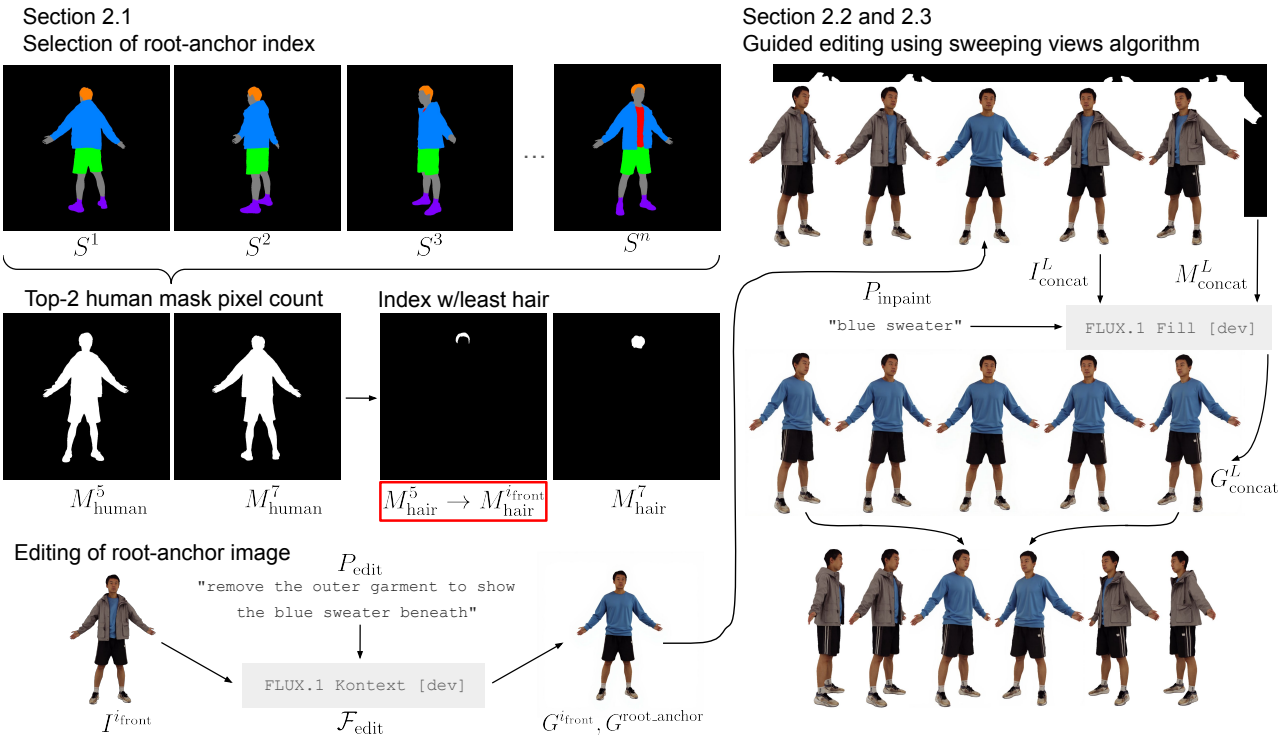


Figure 1. Overview of our method. We first select the front-facing image out of all images using a simple heuristic: the image with least hair from the two images with most pixels showing a human. We edit the front-facing image with `FLUX.1 Kontext [dev]`. Then, following the sweeping views algorithm, we use the edited image as guidance for the two contiguous views to the left and right. Once those images are edited, we use the image most to the left and right as guidance for the next batch of images. This process continues until all images have been edited.

4.1. Selection and editing of root-anchor image

To select the image including the person facing forward we use the segmentation maps and a simple heuristic: choose the top-2 images with higher pixel count for the human silhouette. The mask including pixels with label $j \in \mathcal{L}$ where $\mathcal{L} = \{\text{skin, hair, shoes, inner, lower, outer}\}$ and its total count are defined as follows:

$$\boldsymbol{M}_j^i(x,y) = \begin{cases} 1 & \text{if } \boldsymbol{S}^i(x,y) = C_j \\ 0 & \text{otherwise} \end{cases}, K_j^i = \sum_{x=1}^{W}\sum_{y=1}^{H} M_j^i(x,y)$$

Where C_j is the color corresponding to the 4D-DRESS palette. The total count of the human silhouette for image $\boldsymbol{I}^i$ is thus $K^i_{\text{human}} = \sum_{j\in\mathcal{L}} K^i_j$. The views with the top-2 count will be either the front-facing human or the back-facing human. To select from these two images which make up the indices set $\boldsymbol{I}_{\text{top2}}$, we choose the one with the lowest hair pixel count:

$$i_{\text{front}} = \arg\min_{i \in \boldsymbol{I}_{\text{top2}}} K^i_{\text{hair}}$$

Next, we use $\mathcal{F}_{\text{edit}}$ which edits images based on text instructions. We use $\boldsymbol{I}^{i_{\text{front}}}$ and instruction P_{edit} as the inputs:

$$\boldsymbol{G}^{\text{root\_anchor}} = \boldsymbol{G}^{i_{\text{front}}} = \mathcal{F}_{\text{edit}}(\boldsymbol{I}^{i_{\text{front}}}, P_{\text{edit}})$$

The output is the first edited image, $\boldsymbol{G}^{\text{root\_anchor}}$, which will serve as the guidance image for other views.

4.2. Guided editing with inpainting

Inspired by CatVTON [1], a virtual try-on diffusion model that transfers garments to a target human by concatenating the garment and human images along spatial dimensions then using this as the input image to a diffusion model, we decide to concatenate a set of input images $\{\boldsymbol{I}^i\}_{i\in L}$ to be edited and guidance image $\boldsymbol{G}^{\text{root\_anchor}}$ along the spatial dimensions to use as input to an inpainting model.

In general, inpainting models fill specific regions of an image $\boldsymbol{I}$. Such regions are defined by a mask $\boldsymbol{M}$ and guided by prompt P_{inpaint}. For our method, given a list of indices L containing both: images which have not been edited yet and images which have been edited and will guide the other views, we create the set $\{\boldsymbol{I}^i_{\text{inpaint}}\}_{i\in L}$ as follows:

$$\boldsymbol{I}^i_{\text{inpaint}} = \begin{cases} \boldsymbol{G}^i & \text{if } \boldsymbol{G}^i \text{ exists} \\ \boldsymbol{I}^i & \text{otherwise} \end{cases}$$

For the mask since guidance images shouldn't be edited, their masks are all zeros. Then, for the remaining images we compute the masks from the segmentation maps $\{\boldsymbol{S}^i\}_{i=1}^N$ and set of garments $\boldsymbol{l}$:

$$\boldsymbol{M}^i_{\text{inpaint}}(x,y) = \begin{cases} 0 & \text{if } \boldsymbol{G}^i \text{ exists} \\ \bigvee_{j \in \boldsymbol{l}} \boldsymbol{M}^i_j(x,y) & \text{otherwise} \end{cases}$$

The concatenation process seeks to minimize the empty regions in the final image by cropping the input images to exactly include the human bounding box. The cropped images and masks are concatenated first along rows and then along columns with padding between the images. The optimal concatenation is computed using the linear partition algorithm as described in section 8.5 of the Algorithm Design Manual [19] on the width of the cropped images; specifically we use this Stack Overflow implementation [13] of the algorithm. The number of desired rows is that whose final concatenated image most closely matches the ratio, r, between image width and height, a parameter which is useful as different inpainting models are trained at different resolutions and aspect ratios.

During cropping and concatenation, the bounding box and top left corners coordinates of each image are saved to be able to undo the concatenation and save each edited image with the person using the same canvas space as in the original image.

More specifically, the concatenation process returns the concatenateds image $\boldsymbol{I}_{\text{concat}}$, mask $\boldsymbol{M}_{\text{concat}}$, and coordinates information C^L:

$$\text{Concat}(L, \{\boldsymbol{I}^i_{\text{inpaint}}\}_{i\in L}, \{\boldsymbol{M}^i_{\text{inpaint}}\}_{i\in L}, r) = \boldsymbol{I}^L_{\text{concat}}, \boldsymbol{M}^L_{\text{concat}}, C^L$$

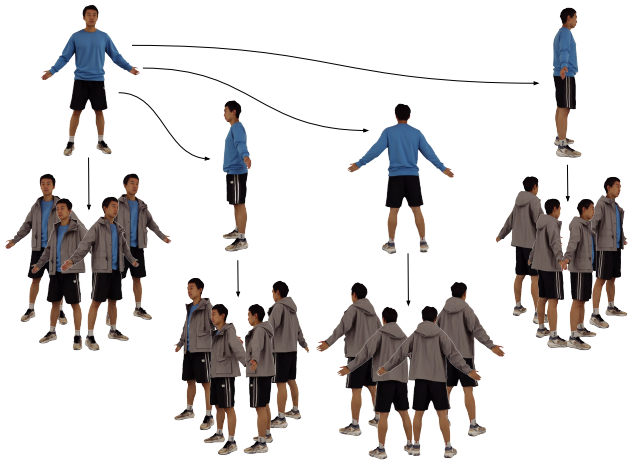


Figure 2. Directed graph showing the evenly spaced views partitioning with 4 anchors and 20 images to edit. The root-anchor image is shown on the top left corner, it guides generation of the left, back, and right views of the human. Then each of the anchors guides the generation of its neighbors. The edges stem from the guidance image and point to the images it guides inpainting for.

The inpainting model's output is the image with the concatenated edited images:

$$\mathcal{F}_{\text{inpaint}}(\boldsymbol{I}^L_{\text{concat}}, \boldsymbol{M}^L_{\text{concat}}, P_{\text{inpaint}}) = \boldsymbol{G}^L_{\text{concat}}$$

Finally, we undo the concatenation of the images and get the output edited images $\{\boldsymbol{G}^i\}_{i\in L}$:

$$\{\boldsymbol{G}^i\}_{i\in L} = \text{Deconcat}(\boldsymbol{G}^L_{\text{concat}}, C^L)$$

4.3. Partitioning algorithms

Since our method seeks to preserve identity and pose we don't rescale the images. This leads to big concatenated images testing the limits of inpainting models as quality decreases with bigger resolutions even if memory is sufficient.

To circumvent this limit and generate an arbitrary number of views, we use the root-anchor image to generate at most 6 other views. These can be used as guidance to edit another 6 views, and so on. We introduce two different strategies to decide which view to use as guidance at each step. These strategies assume that the indices are already ordered. In our experiments we use the 4D-DRESS dataset which has a circular trajectory increasing the azimuth angle.

Algorithm 1 Recursive Evenly Spaced Partitioning

Input: I_0: Root-anchor index. $\mathcal{I}$: Sorted view indices list.
Parameters: k (number of anchors to select), N (total views), M (maximum group size).
Output: $\mathcal{P}$: A partition of $\mathcal{I}$ specifying the generation order for the view indices.

function EVENLYSPACEDPARTITION($I_0, \mathcal{I}$)
  Step 1: Select k evenly strided anchors $\mathcal{A}$ from $\mathcal{I}$
  Select $\mathcal{A}$ by taking k indices from $\mathcal{I}$ starting at I_0 with a stride of round($|\mathcal{I}|/k$).
  Step 2: Assign each index to its closest anchor in $\mathcal{A}$
  Init a map of indices to lists $\mathcal{G} \leftarrow \{a : [] \text{ for } a \in \mathcal{A}\}$
  for each index $i \in \mathcal{I}$ **do**
    $d(i,a) \leftarrow \min(|i-a|, N - |i-a|)$
    $a^* \leftarrow \arg\min_{a\in\mathcal{A}} d(i,a)$
    Append i to $\mathcal{G}[a^*]$
  end for
  Step 3: Recursively partition groups larger than M
  $\mathcal{P}_{final} \leftarrow \emptyset$
  for each anchor index $a \in \mathcal{A}$ **do**
    $\mathcal{G}_a \leftarrow \mathcal{G}[a]$
    if $|\mathcal{G}_a| > M$ **then**
      $\mathcal{P}_{sub} \leftarrow$ EVENLYSPACEDPARTITION($a, \mathcal{G}_a$)
      Append all lists from $\mathcal{P}_{sub}$ to $\mathcal{P}_{final}$
    else
      Append $\mathcal{G}_a$ to $\mathcal{P}_{final}$
    end if
  end for
  return $\mathcal{P}_{final}$
end function

Evenly spaced views partitioning. This strategy is akin to a sorted tree structure as the anchor image guides views which have the same distance between consecutive views. Then, the newly generated images serve as guidance for the next batch of images which are also equally spaced. This process continues in a recursive way until all images have been generated. The input to the algorithm are the indices to generate in a sorted list and the root-anchor index. The algorithm is described in Algorithm 1. An example of the partitioned list in action is shown in Fig. 2 where the rendered images and edges pointing from the guidance images to the guided images are shown.

Sweeping views partitioning. This strategy focuses on consistency as it generates the views in both directions starting from the root-anchor image. The root-anchor image serves as guidance to the two views to the left and to the right. Then the image most to the right serves as guidance to the next two images to its right; the left image is used in the same way. This process follows until the last image is generated. The input is the same as for the evenly spaced views partitioning algorithm. The algorithm is described in Algorithm 2. An example of the partitioned list in action is shown in Fig. 3 where each row depicts the indices of each list in the partitioning.

Algorithm 2 Sweeping Views Partitioning

Input: I_0: Root-anchor index. N: Total number of views.
Output: $\mathcal{P}$: A partition of $\mathcal{I}$ specifying the generation order for the view indices.

function SWEEPPARTITION(I_0, N)
  Initialize current anchors $\mathcal{A}_{\text{curr}} \leftarrow [I_0, I_0]$
  Initialize completed set $\mathcal{C} \leftarrow \{I_0\}$
  Initialize partition $\mathcal{P} \leftarrow \emptyset$
  while $|\mathcal{C}| < N$ **do**
    Get the left and right anchors:
      $a_0 \leftarrow \mathcal{A}_{\text{curr}}[0], a_1 \leftarrow \mathcal{A}_{\text{curr}}[1]$
    Get the views to the left of the left anchor
      $\mathcal{L}_{\text{left}} := [a_0 - 2, a_0 - 1, a_0] \pmod N$
    Get the views to the right of the right anchor
      $\mathcal{L}_{\text{right}} := [a_1, a_1 + 1, a_1 + 2] \pmod N$
    Append $\mathcal{L}_{\text{left}} + \mathcal{L}_{\text{right}}$ to $\mathcal{P}$
    Update left and right anchors to new boundaries
      $\mathcal{A}_{\text{curr}} \leftarrow [\mathcal{I}_{\text{curr}}[0], \mathcal{I}_{\text{curr}}[-1]]$
    Update the set of completed indices.
      $\mathcal{C} \leftarrow \mathcal{C} \cup \text{set}(\mathcal{I}_{curr})$
  end while
  return $\mathcal{P}$
end function

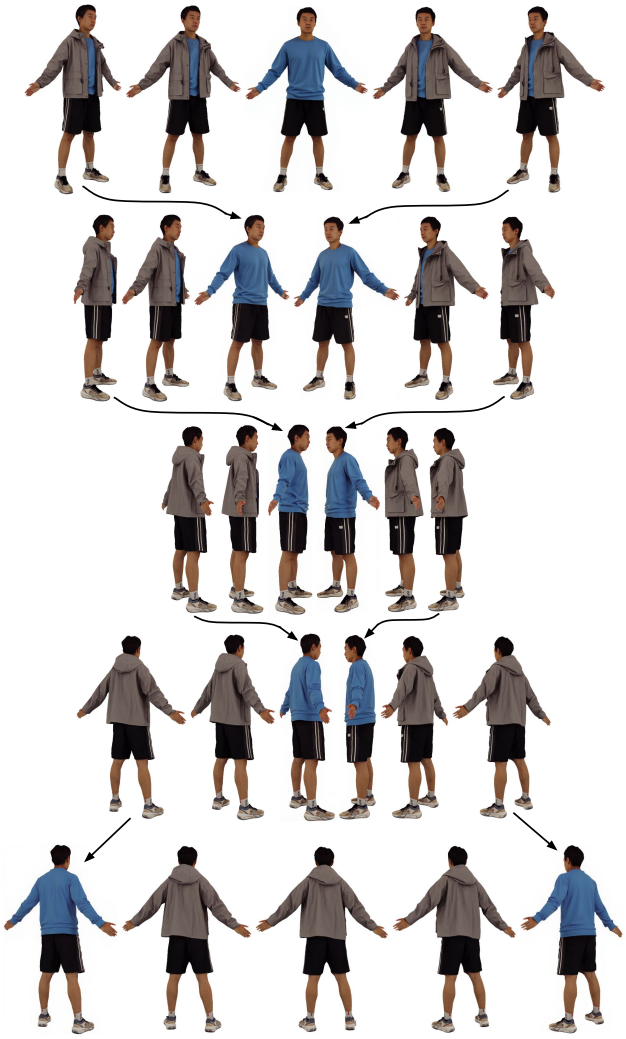


Figure 3. Input images, per row, to the inpainting model when using the sweeping views partitioning with 20 images. Each row contains one or two edited images which will guide inpainting for the other images in that same row. The arrows point from the image before editing to the image after editing, showing the guidance propagation from the front-view to the back-view.

5. Experiments

5.1. Datasets

Our experiments are conducted on the 4D-DRESS dataset [22] which contains 82 scans, out of which 42 are "Outer" scans where the person is wearing upper inner and outer garments. Outer garment examples include a jacket, sweater, or any other garment that serves as an outer layer. The remaining 40 scans don't contain outer layers as labeled in 4D-DRESS.

We render the textured meshes and segmented meshes from $N = 20$ views in a circular trajectory with radius $R = 1.5$ and with circular camera positions for view i:

$$\mathbf{c}_i = \left(R \sin\left(\frac{2\pi i}{N}\right), 0, R\cos\left(\frac{2\pi i}{N}\right)\right)$$

These renders make up the images $\{\boldsymbol{I}^i\}_{i=1}^N$ and segmentation maps $\{\boldsymbol{S}^i\}_{i=1}^N$. The camera poses $\{\boldsymbol{P}^i\}_{i=1}^N$ are also saved for later evaluation.

5.2. Tasks

We focus on two tasks on the upper body garments: (1) removing the outer garment for the 42 "Outer" scans but keeping the inner garment and (2) removing the inner garment for all scans. For the second task, we edit the images obtained from the first task, if available, as to keep consistency between the two tasks in: skin color, body pose, and garment color, style, and shape.

For the first task, the instruction P_{edit} is either "remove the outer garment" or "remove the *white jacket*" followed by "to show the *yellow shirt* underneath" where the italicized text changes across scans. The prompt P_{inpaint} is a substring of P_{edit} and is a short description of the inner garment specifying the color and type of garment. We test $\boldsymbol{l} = \{\text{inner, outer}\}$ and $\boldsymbol{l} = \{\text{inner}\}$ as the set of garments to be edited.

For the second task, for scans showing men, the instruction P_{edit} is "remove the upper garment to show a man's chest" and "remove the upper garment to show a sports bra" for women. The prompt P_{inpaint} is "a man's torso" for men which is a more general description than chest which woudn't work for back-facing men and "sports bra" for women. The set of garments to be edited are $\boldsymbol{l} = \{\text{inner}\}$.

5.3. Implementation details

As mentioned before, we opted for two models from Black Forest Labs: `FLUX.1 Kontext [dev]` [9] for the first stage and `FLUX.1 Fill [dev]` [8] for the second stage as they both achieve state-of-the-art generation quality and their weights are openly available. We use the `diffusers` library and run all experiments on an A100 GPU.

We render images at 1024x1024 resolution as `FLUX.1 Kontext [dev]` is known to perform well at this resolution. During concatenation, we aim for up to 6 concatenated images and an aspect ratio $r = 16/9$. A higher count of concatenated images results in reduced generation quality even if the concatenated image fits in GPU memory.

In practice, `FLUX.1 Fill [dev]` is sensitive to the mask shape and size: if the mask is too tight and pixels corresponding to the garment to edit are not masked out, the generation will fail and generate the garment it was supposed to edit. While a dilation of the mask solves this issue, it also masks out hands which the model fails to reconstruct. Thus, we first dilate the mask with a square 5x5 kernel for 3 steps. Then, we erode the skin mask for 2 steps with a 3x3 ellipse-shaped kernel and mask out the valid pixels in this eroded skin mask from the dilated garment mask.

5.4. Metrics

Our metrics test 3D consistency between rendered images through the cPSNR, cSSIM, and cLPIPS metrics from MVGBench [24]; editing success rate and quality by prompting a VLM on the generated images; consistency before and after the generation through PSNR, SSIM on the inner garment which shouldn't have changed in the first task; and IoU on regions like face, hands, and other regions whose area on the image shouldn't change.

We also considered using human pose estimation models to inspect whether human poses changed after the generation; however we decided against using this metric given the results from STAGE [7] which finds severe degradation in the performance of pose estimators as they are sensitive to gender, ethnicity, age, and other attributes.

3D consistency metric. MVGBench [24] correctly identifies that consistency must be measured between the generated images when there's no ground truth in the generation task. They propose fitting two 3DGS representations using two non-overlapping subsets of the generated images and computing PSNR, SSIM, and LPIPS metrics between renders of the two 3DGS representations using the same camera pose. In our experiments, the first subset contains the even-index views and the other the odd-index views. Then for each set, the 10 camera poses which were not used to fit the 3DGS are used to render and compute the metrics which we call cPSNR, cSSIM, and cLPIPS.

Editing success rate and quality. For this metric we use the open-weights Gemma-3 4B VLM model [20]. While inference is fast, evaluating on all views for both tasks and scans is expensive so we limit to views 0, 4, 8, 12, and 16. An example of the prompt to check whether the generated image removed the garment is as follows: "Focusing only on the upper body garment and not the shoes or pants: Answer yes if the person in the image is only wearing a *white polo shirt*. Answer no if the person is wearing a jacket, blazer, or outer garment on top of the *white polo shirt*." where the italicized text changes across scans.

In practice, `FLUX.1 Fill [dev]` may generate images that contain remains of the removed garment, especially in regions like neck and wrists. Thus, we prompt Gemma on the quality of the removal as follows: "The person in the image had their jacket, blazer, or outer garment removed and thus should only be wearing a *white polo shirt*. Your task is to rate the quality of the removal. On a scale from 1 to 10, rate the removal. 10 means that the removal was successful and the person is only wearing a *white polo shirt*. 1 means that the removal was unsuccessful and the

person is wearing a jacket, blazer, or outer garment on top of the *white polo shirt*.'"

Consistency before and after the generation. Since one of our goals is to preserve the person's identity, pose, and garments which are not to be edited, we need a metric to ensure our method attains this. We do so by reporting the PSNR and SSIM of the regions which are not to be edited before and after the generation.

More specifically, for the first task we compute PSNR and SSIM for the region specified by the "inner" garment mask. We also compute the IoU metric on the hair, shoes, and the lower garment and skin below the newly generated inner garment regions. We use a SegFormer [23] model fine-tuned on the ATR dataset [10] to compare the segmentation map before and after the generation. This last metric is meant to measure whether the camera and human pose change before and after the generation.

5.5. Quantitative Comparison

We conduct the quantitative comparison between our method with evenly spaced partitioning, our method with sweeping partitioning, `FLUX.1 Fill [dev]` with 6 concatenated images, and the recent MV-Adapter [4] for its ease-of-use, impressive performance, and open weights. For the last model's available weights, it can only generate up to 6 views from azimuth angles 0, 45, 90, 180, 270, 315. We use these same camera poses for the method `FLUX.1 Fill [dev]` when concatenating 6 images.

Editing success rate and quality using different masks. Before comparing our method and the two partitioning strategies, we need to determine what garments are to be edited in the first task, either choosing $\boldsymbol{l}_{\text{outer}} = \{\text{outer}\}$ or $\boldsymbol{l}_{\text{upper}} = \{\text{inner, outer}\}$. Since the "Outer" scans of 4D-DRESS show partially open jackets, vests, and other outer garments, the first task should preserve the inner garment. Thus, we initially chose $\boldsymbol{l}_{\text{outer}}$. However, as seen in Table 1, the VLM metrics measuring editing success rate and quality are lower for both cases when comparing the generations using $\boldsymbol{l}_{\text{outer}}$ against $\boldsymbol{l}_{\text{upper}}$. We believe that the mask acts as a prior for the generation, thus, for instance, if the mask resembles the outline of a jacket, it will ignore the text prompt and generate a jacket. All following comparisons for the first task which require masks use $\boldsymbol{l}_{\text{upper}}$.

3D consistency metric. We conducted the 3D consistency metrics on our method using the sweeping views partitioning and the evenly spaced views partitioning aiming for 4, 5, and 6 anchors. Results are shown in Table 2. Among these, the sweeping views partitioning algorithm outperforms the evenly spaced algorithm since the guidance image is an immediately contiguous image. Figure 4 third row, images 2 and 3 show a consistency failure case. The pitfall of the evenly spaced algorithm is when left and right images of the person guide views of the person facing slightly to the front or back: the left and right views don't provide any information about the garment seen from the front nor back.

We also report this metric for the original dataset from the same 20 views which serves as the maximum baseline. The sweeping views partitioning approach scratches a 30 PSNR value which is not too far from the high-quality dataset captures at 36.7 PSNR. Finally, we report the 4D-DRESS dataset with each image shifted one pixel on any left, right, up, or down directions at random. This metric alleviates our concerns that the images may be shifted by `FLUX.1 Fill [dev]` after the generation since the PSNR score is lower than the sweeping views approach.

We also tried to test the MV-Adapter and `FLUX.1 Fill [dev]` with 6 views results on this metric but the scarcity of views doesn't allow the 3DGS to converge to a meaningful representation from which to render images.

Table 1. Editing Success Rate and Quality when using outer mask and inner and outer mask

Partitioning	**Sweeping**		**Evenly Spaced w/4 anchors**	
Metric	**Outer**	**Inner+Outer**	**Outer**	**Inner+Outer**
Outer Removal Success Rate	0.6381	**0.7143**	0.7952	**0.819**
Outer Removal Quality	9.3333	**9.4619**	9.6429	**9.7**

Table 2. Comparison of 3D Consistency Metrics

Method	**Sweeping**	**Evenly Spaced (4 anchors)**	**Evenly Spaced (5 anchors)**	**Evenly Spaced (6 anchors)**	**4D-DRESS**	**4D-DRESS (+1px shift)**
cPSNR	**29.3475**	28.5451	28.4448	28.1745	36.7200	29.2274
cSSIM	**0.9705**	0.9674	0.9666	0.9651	0.9915	0.9799
cLPIPS	**0.0216**	0.0242	0.0254	0.0265	0.0045	0.0185

Figure 4. Removal of outer garment for scan 00134_Outer from the 4D-DRESS dataset using our method.
P_{edit}: remove the blue cardigan to show the brown shirt underneath
P_{inpaint}: brown shirt

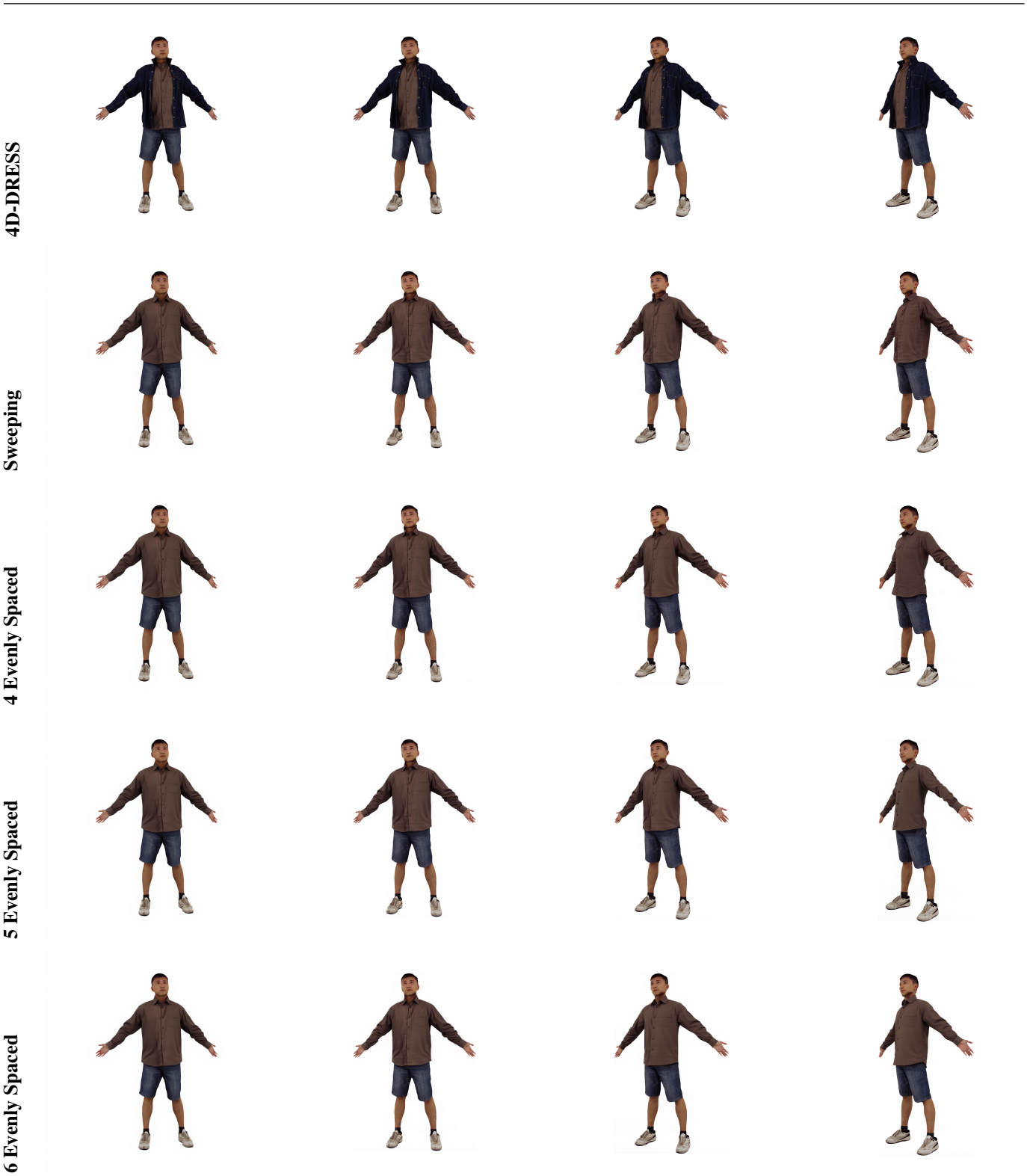


Table 3. Comparison of Editing Success Rate and Quality

Metric	Sweeping	Evenly Spaced (anchors)			Flux Fill (6 views)
		4	5	6	
Outer Removal Success Rate	0.7143	0.8190	**0.8952**	0.8333	0.7698
Outer Removal Quality	9.4619	9.7000	**9.7286**	9.5952	9.5040
Inner Removal Success Rate	0.9317	0.9463	**0.9512**	0.9073	0.9106
Inner Removal Quality	8.8683	8.9293	**8.9610**	8.6927	8.7805

Editing success rate and quality. We report this metric for methods which use the inpainting model `FLUX.1 Fill [dev]` since MV-Adapter just takes in the root-anchor image and has no issue replicating the edits. The evenly spaced views partitioning approach outperforms all other methods for both tasks and metrics. We believe the sweeping partitioning method specifically fails for the first task since it concatenates contiguous views of the person starting with the front image. The masks will thus resemble the outline of the outer garment guiding the generation to the very same garment which is to be removed. For the second task the success rate and quality are not too far apart but since the images generated in the first task are reused, we believe the bad performance on the first task propagates.

Consistency before and after the generation. As before, we test our method with both partitioning algorithms. We also include MV-Adapter and `FLUX.1 Fill [dev]` with 6 views. Results are shown in Table 4.

When it comes to preserving the inner garment while removing the outer garment, `FLUX.1 Fill [dev]` with 6 views outperforms all other methods and the sweeping views approach is in a close second place. We believe both of these methods benefit from using contiguous views as guidance images. Recall that the mask includes all upper garments since removal success rate improved this way. Surely by masking out the inner garment, we sacrifice preserving it.

The IoU metric is similar between all of our methods oscillating between 0.9 and 0.91 which are pretty high considering that the SegFormer model is not perfect. As a baseline, we do inference on the 4D-DRESS dataset with 1-pixel shift for each image in a random direction which yields a 0.93 IoU. Still, we are sure that there is no camera pose change since our method performs well in the cPSNR and cSSIM metrics. Instead, the IoU decrease could be attributed to artifacts in the generation process or slight human pose or identity variation.

Lastly, MV-Adapter performs bad across all metrics: it is not able to recreate the inner garment and changes the human pose, identity, and camera pose. This happens since the MV-Adapter's VAE fails to capture the details of the root-anchor image as shown in Fig. 5.

5.6. Qualitative Comparison

Figure 4 presents the "00134_Outer" scan before and after the first task. Outer garment removal is successful for all configurations of our method. We show images 0-3 to show the failure in consistency for the evenly spaced results. For 4 anchors, the shirt changes the buttons colors between images 2 and 3. For 5 anchors, the shirt has a crease in the middle in image 2 which is not shown in image 3; for 6 anchors the same failure occurs between images 1 and 2.

Figure 5 shows that MV-Adapter fails to encode the root-anchor image and thus also fails to preserve the identity, human pose, and camera pose across views.

Figure 6 shows the results of the second task on "00180_Inner". As in Fig. 4 the most consistent method is

Table 4. Comparison of consistency before and after generation

Region/Task	Metric	Sweeping	Evenly Spaced (anchors)			Flux Fill (6 views)	MV Adapter
			4	5	6		
Inner Garment	PSNR	**16.1183**	15.4676	15.6186	15.7059	**16.4942**	7.2435
	SSIM	**0.4236**	0.4140	0.4116	0.4117	**0.4304**	0.2818
Outer Removal	IoU	0.9013	**0.9049**	0.9038	0.9007	**0.9071**	0.5690
Inner Removal	IoU	0.9034	0.9078	0.9073	**0.9109**	0.9107	0.5798

sweeping views. All configurations of our method successfully remove the outer garment. The evenly spaced method fails to keep the man's chest consistent from image 18- 19.

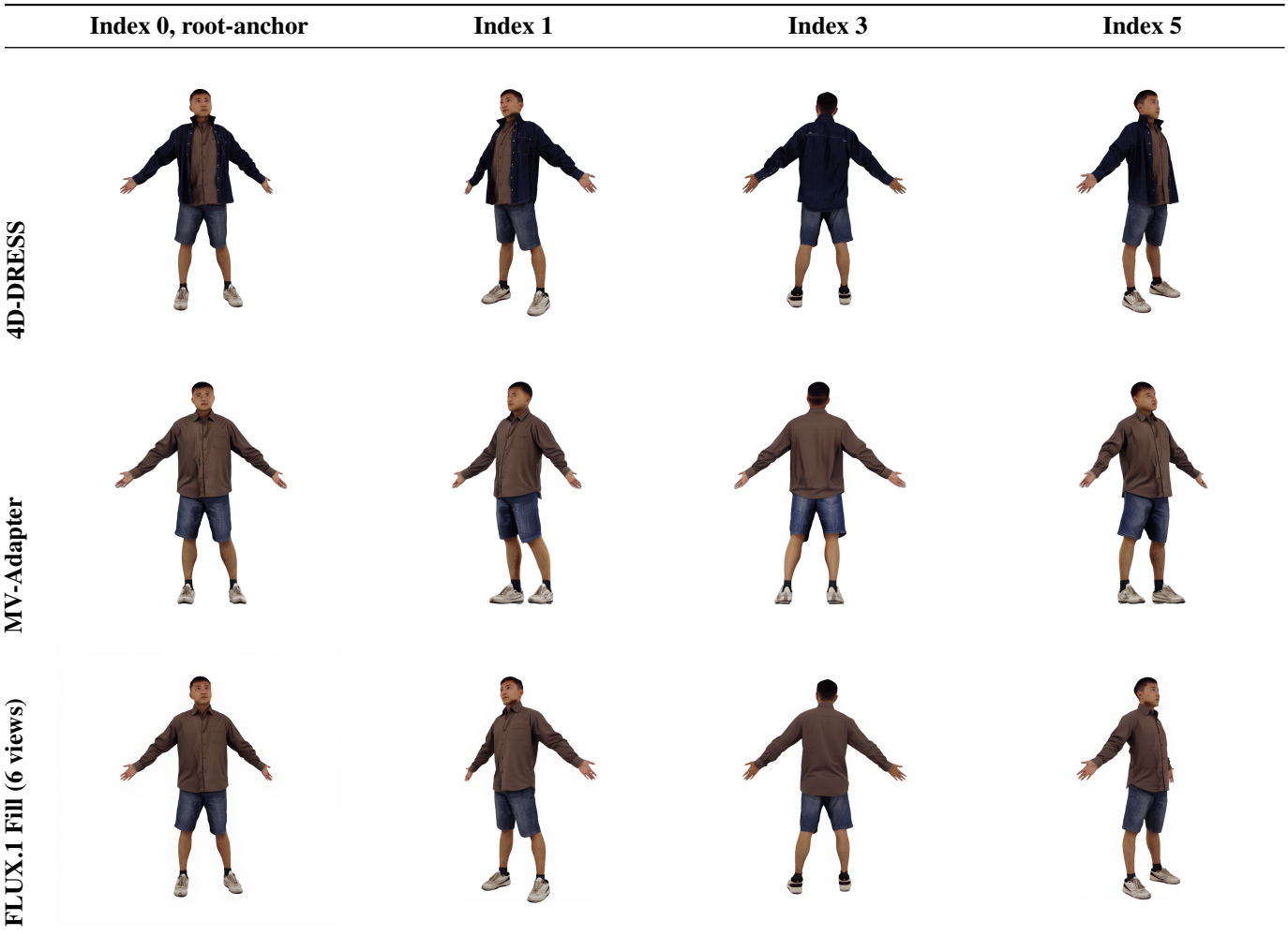
Figure 7 shows the results of the second task on scan "00136_Inner" which shows a woman. As before, the sweeping algorithm shows the most consistency and this time the evenly spaced views algorithm fails in keeping the color of the sports top consistent.

Failure cases of our method. Our method struggles with removing loose or over-sized outer garments. For instance, Fig. 8 shows the failed removal of the over-sized square shirt in the original scan. The sweeping views algorithm fails to properly reconstruct the jean shorts and instead generates a mix of skirt and shorts. We attribute this failure to the mask acting as a prior to the generation. This is most clearly seen in images 7 and 8 which shows shorts that cover the same area as the over-sized shirt.

The evenly spaced results fail in similar ways generating coats and inconsistently long shorts. Furthermore, it presents more consistency problems like in previous figures. For the 4-anchor configuration, the shorts color changes between images 7 and 8.

Lastly, Fig. 9 shows that shadows from the inner garment can steer the generation away from the root-anchor image. Focusing in image 1, all methods present artifacts in the left arm of the man. The reason behind this is that even though the inpainting mask is accurate as it selects every pixel of the blue shirt, the shirt produces a shadow on the arm. This shaded region is not masked out as it is part of the skin so it steers the model to generate a tattoo for the sweeping views method or a dark line in the other methods.

Figure 5. Removal of outer garment for scan 00134_Outer using `FLUX.1 Fill [dev]` with 6 views and MV-Adapter.
P_{edit}: remove the blue cardigan to show the brown shirt underneath
P_{inpaint}: brown shirt



6. Conclusion

In this work we presented a text-driven editing strategy which applies 3D-consistent fashion edits on multi-view images of a person. Our method leverages the impressive performance of image editing models and expands it to multi-view images by concatenating the input images along spatial dimensions. Our method guides diffusion models to apply the edits in a hierarchical way unlocking the number of possible inputs. Furthermore, it only relies on consistent masks of the region to be edited which can be obtained with segmentation models or user input, foregoing human pose estimation or camera pose estimation which may not be consistent between frames. We report metrics on 3D consistency, generation quality, and consistency before and after applying the edits which show our method's capabilities on challenging tasks like removing outer garments while keeping inner garments consistent. These findings underscore the generalization of state-of-the-art diffusion models and their potential to achieve specific tasks when prompted in a structured and precise manner.

References

[1] Zheng Chong, Xiao Dong, Haoxiang Li, Shiyue Zhang, Wenqing Zhang, Xujie Zhang, Hanqing Zhao, and Xiaodan Liang. Catvton: Concatenation is all you need for virtual try-on with diffusion models, 2024. 1, 2, 4

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[3] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. 2024. 2

[4] Zehuan Huang, Yuanchen Guo, Haoran Wang, Ran Yi, Lizhuang Ma, Yan-Pei Cao, and Lu Sheng. Mv-adapter: Multi-view consistent image generation made easy. *arXiv preprint arXiv:2412.03632*, 2024. 2, 3, 8

[5] Jianbin Jiang, Tan Wang, He Yan, and Junhui Liu. Clothformer: Taming video virtual try-on in all module. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 3

[7] Nikita Kister, István Sárándi, Anna Khoreva, and Gerard Pons-Moll. Are pose estimators ready for the open world? stage: Synthetic data generation toolkit for auditing 3d human pose estimators. In *Arxiv*, 2024. 7

[8] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 1, 2, 3, 7

[9] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 1, 2, 3, 7

[10] Xiaodan Liang, Si Liu, Xiaohui Shen, Jianchao Yang, Luoqi Liu, Jian Dong, Liang Lin, and Shuicheng Yan. Deep human parsing with active template regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(12):2402–2414, 2015. 8

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[12] Pamela Mishkin, Lama Ahmad, Miles Brundage, Gretchen Krueger, and Girish Sastry. Dall·e 2 preview - risks and limitations. 2022. 1

[13] n.m. How to understand the dynamic programming solution in linear partitioning. Stack Overflow, 2011. Accessed: 2025-10-17. 5

[14] OpenAI. Introducing 4o image generation. https://openai.com/index/introducing-4o-image-generation/, 2025. Accessed: 2025-10-28. 1

[15] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 2

[16] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 3

[19] Steven S. Skiena. *The Algorithm Design Manual*. Springer, Cham, 3rd edition, 2020. 5

[20] Gemma Team. Gemma 3. 2025. 7

[21] Haoyu Wang, Zhilu Zhang, Donglin Di, Shiliang Zhang, and Wangmeng Zuo. Mv-vton: Multi-view virtual try-on with diffusion models. *arXiv preprint arXiv:2404.17364*, 2024. 1, 2

[22] Wenbo Wang, Hsuan-I Ho, Chen Guo, Boxiang Rong, Artur Grigorev, Jie Song, Juan Jose Zarate, and Otmar Hilliges. 4d-dress: A 4d dataset of real-world human clothing with semantic annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 6

[23] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *CoRR*, abs/2105.15203, 2021. 8

[24] Xianghui Xie, Chuhang Zou, Meher Gitika Karumuri, Jan Eric Lenssen, and Gerard Pons-Moll. Mvgbench: Comprehensive benchmark for multi-view generation models, 2025. 2, 7

Figure 6. Removal of inner garment for scan 00180_Inner using our method.
P_{edit}: remove the top garment to show a slender, thin, and slim man's chest with narrow frame
P_{inpaint}: a man's torso

	Index 17	Index 18	Index 19	Index 0, root-anchor
4D-DRESS				
Sweeping				
4 Evenly Spaced				
5 Evenly Spaced				
6 Evenly Spaced				

Figure 7. Removal of inner garment for scan 00136_Inner using our method.
P_{edit}: remove the shirt to show sports bra
P_{inpaint}: sports bra

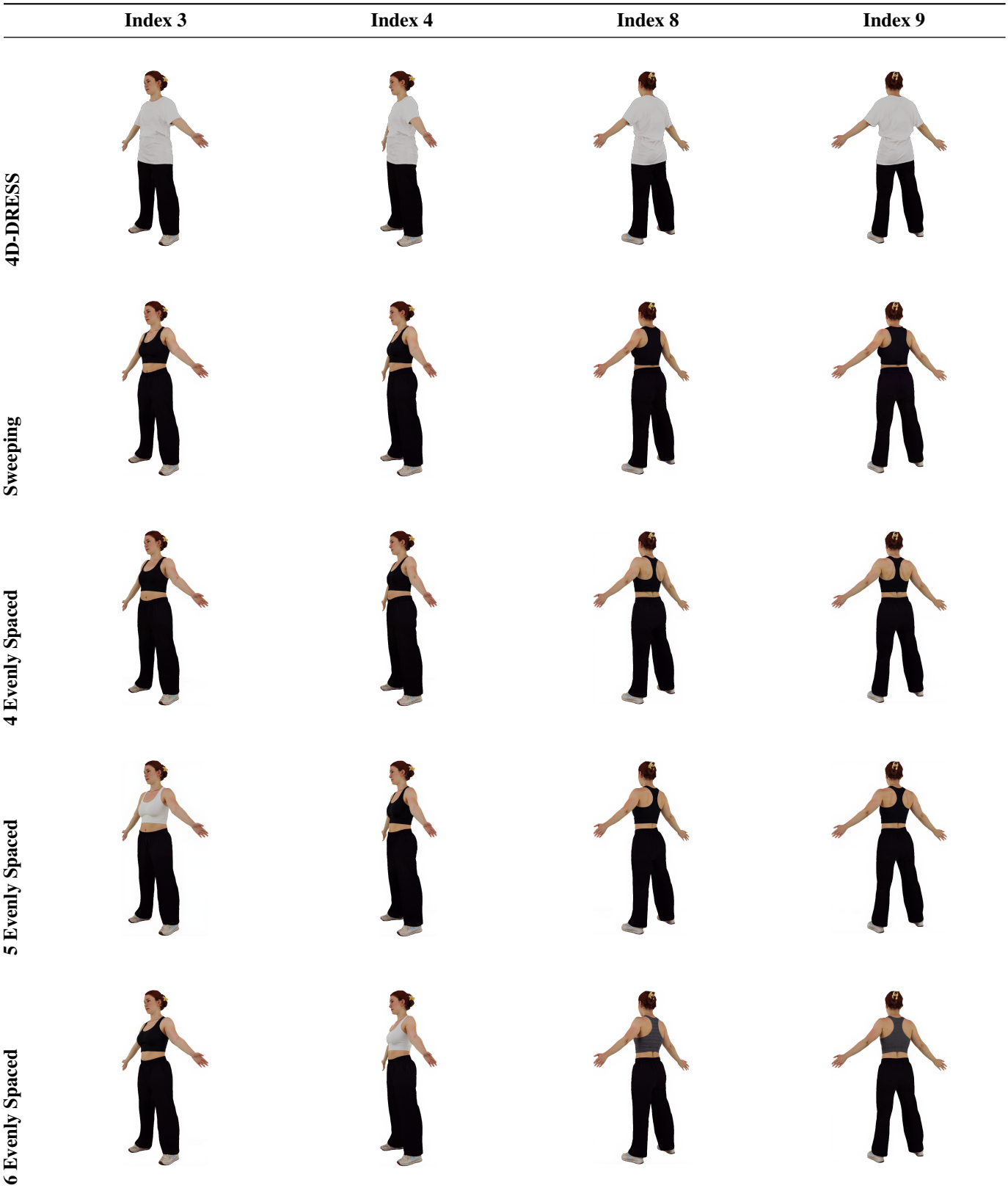


Figure 8. Removal of outer garment for scan 00152_Outer_1 using our method.
P_{edit}: remove the square-pattern shirt to show the yellow shirt underneath
P_{inpaint}: yellow shirt

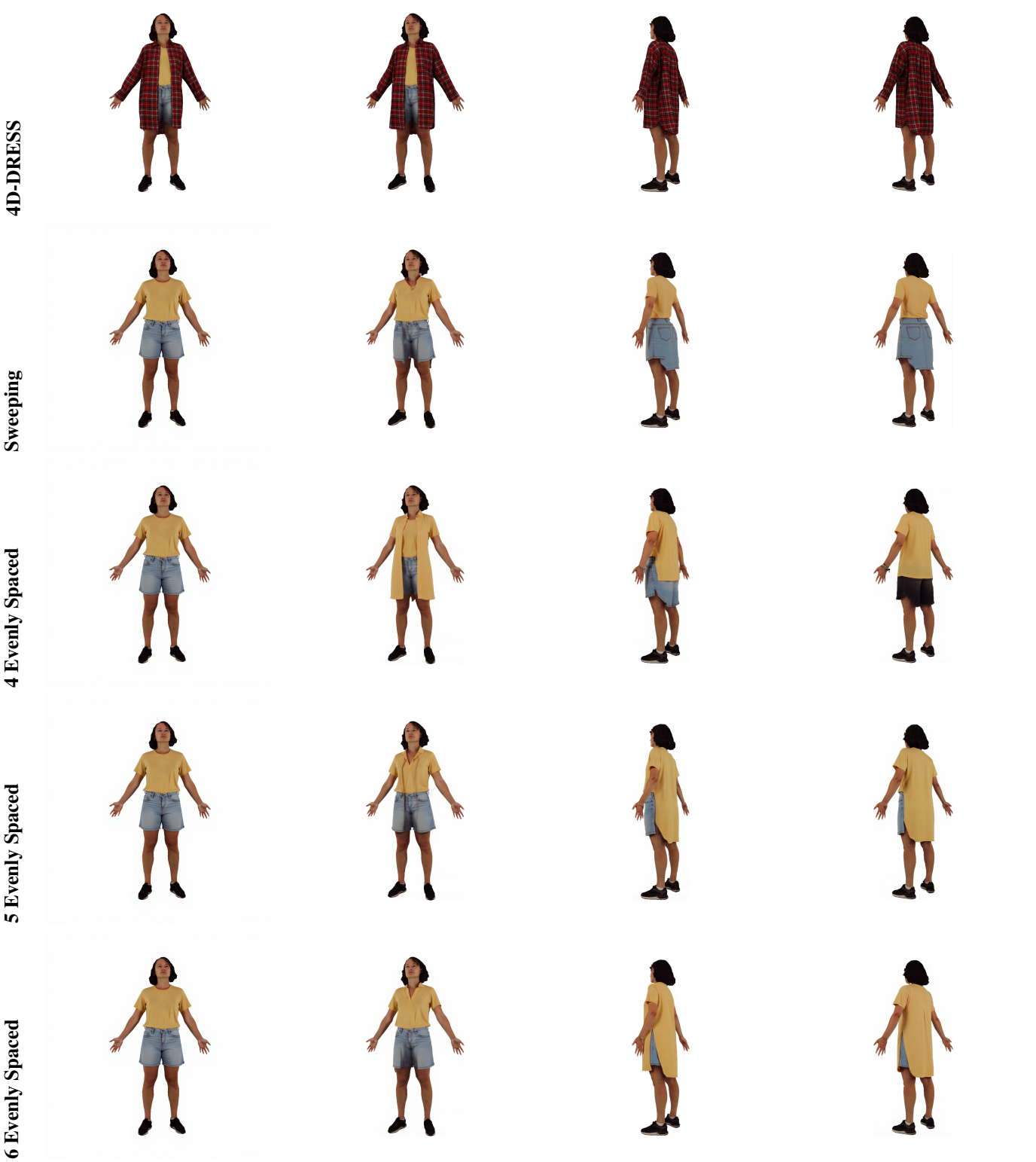


Figure 9. Removal of inner garment for scan 00135_Inner using our method.
P_{edit}: remove the top garment to show a slender, thin, and slim man's chest with narrow frame
P_{inpaint}: a man's torso