Controlling Virtual Try-on Pipeline Through Rendering Policies

Kedan Li kedan@revery.ai
Jeffrey Zhang jeff@revery.ai
Shao-Yu Chang shaoyuc3@illinois.edu
David Forsyth daf@illinois.edu

Abstract

This paper shows how to impose rendering policies on a virtual try-on (VTON) pipeline. Our rendering policies are lightweight procedural descriptions of how the pipeline should render outfits or render particular types of garments. Our policies are procedural expressions describing offsets to the control points for each set of garment types. The policies are easily authored and are generalizable to any outfit composed of garments of similar types. We describe a VTON pipeline that accepts our policies to modify garment drapes and produce high-quality try-on images with garment attributes preserved.

Layered outfits are a particular challenge to VTON systems because learning to coordinate warps between multiple garments so that nothing sticks out is difficult. Our rendering policies offer a lightweight and effective procedure to achieve this coordination, while also allowing precise manipulation of drape. Drape describes the way in which a garment is worn (for example, a shirt could be tucked or untucked).

Quantitative and qualitative evaluations demonstrate that our method allows effective manipulation of drape and produces significant measurable improvements in rendering quality for complicated layering interactions.

1. Introduction

This paper shows how to impose *rendering policies* on a virtual try-on (VTON) pipeline. Our rendering policies are lightweight procedural descriptions of how the pipeline should render outfits or particular types of garment. They are easily authored, and generalize very well. We demonstrate two applications of such polices: changing the drape of garment types, and polishing renderings of layered outfits.

Drape describes the way in which a garment is worn (for example, a shirt could be tucked or untucked). While systems for editing fashion renderings exist [4, 7, 9, 20, 28, 33, 41], these systems modify both drape *and* garment. In contrast, our policies allow the same garment to be draped in different ways. Each *row* of Figure 1 shows images of different garments worn with the same drape and synthesized by our system. The alternatives could be discrete (a jacket could be worn open or closed) or continuous (a skirt worn at different points on the waist or hips) and are often mixed (tucked shirts largely look the same, but there are many ways to wear a shirt untucked). It is straightforward for vendors to author complicated policies that apply to types or particular instances of garments, and draw on various

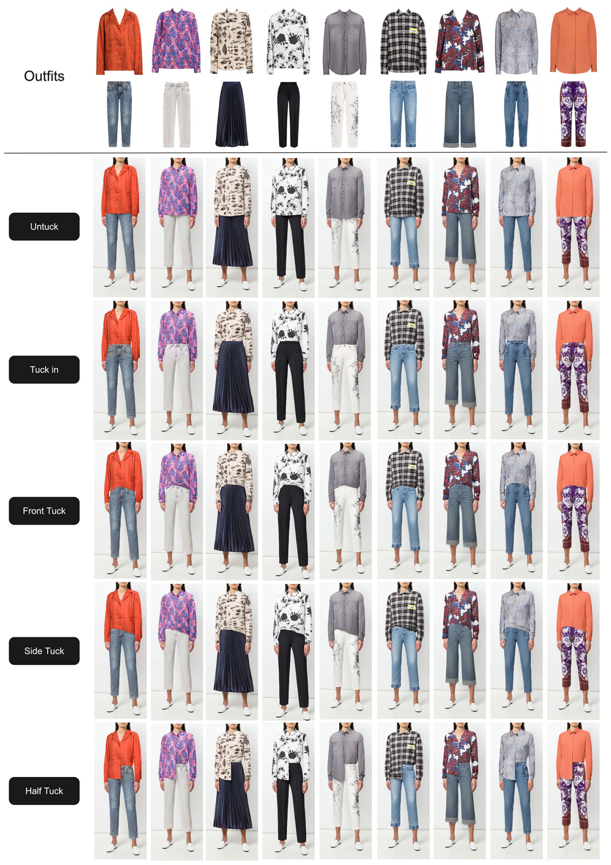

Figure 1. Our method produces high-quality images of people wearing a provided outfit, while allowing the garments to be worn in different ways. The figure shows our method draping the same shirt untucked or tucked in many different styles. Each *column* shows images of the same garment worn in different ways and each *row* shows different outfits worn in the same style/drape. As shown, the same kind of drape renders well on all shirts and the identity of the garments is unaltered when worn in different ways.

metadata (eg. “On Thursdays, red outerwear is worn open, except when paired with skirts”).

Layered outfits are a particular challenge to VTON systems. VTON systems for commerce must accurately represent each specific garment (otherwise a purchaser might return their purchase) [5,6,8,12,13,15,18,20,21,25,27,30,34, 36, 37, 40, 41]. To preserve essential texture details, VTON systems warp garments onto the target image, using warps that are learned. But learning to coordinate warps between multiple garments so that nothing sticks out is difficult (Figure 7). As that figure shows, our rendering policies offer a lightweight and effective procedure to achieve this coordination. Relatively straightforward rendering policies can accurately synthesize outfits with complex layering while prior work produces obvious artifacts (e.g., misalignments, garment beneath sticking out, etc..) as shown in Figure 6.

We describe a VTON pipeline that accepts our policies in Section 3. This pipeline is shaped by important constraints. Garment identity must be preserved, so garments are warped onto a target image. Policies must be grounded in garment semantics, so warper control points have a semantic meaning. For some garments, there is more than one type of warp possible (for example, a coat could be worn open or closed), so the warper must accept discrete parameters to specify which is intended. Once all garments have been warped onto the target image, an adversarially trained renderer polishes it.

Our policies are procedural expressions describing offsets to the control points for each of a set of garment types, possibly in terms of other control points on the same garment or other garments in the outfit. Because each of our control points marks the location of an anchor that is meaningful for the specific garment type (e.g., left shoulder, right inner sleeve, etc..), the policies are meaningful for any outfit composed of garments of the relevant types. For example, one can apply a slightly tilted tuck to all shirts with a policy that moves the waistline control point slightly higher. Rendering policies can be authored interactively: the examples in Figure 1 and 2 were obtained by editing the shirt in one outfit interactively resulting in a policy, then applying the rendering policy to the other shirts. The policies visibly **generalize**. Policies naturally accept continuous parameters (interpolation in Figure 5).

Contributions: We describe the first outfit virtual try-on method that allows reliable control of how a garment is worn on a body without changing the identity of the garment. We demonstrate our method can synthesize images of the same garment worn in different and natural ways. Furthermore, our control is instance independent: one can make edits on a specific garment and apply the same edit to all the other garments of the same category successfully. We show that our method also improves the rendering quality of outfits through better coordination between garments.

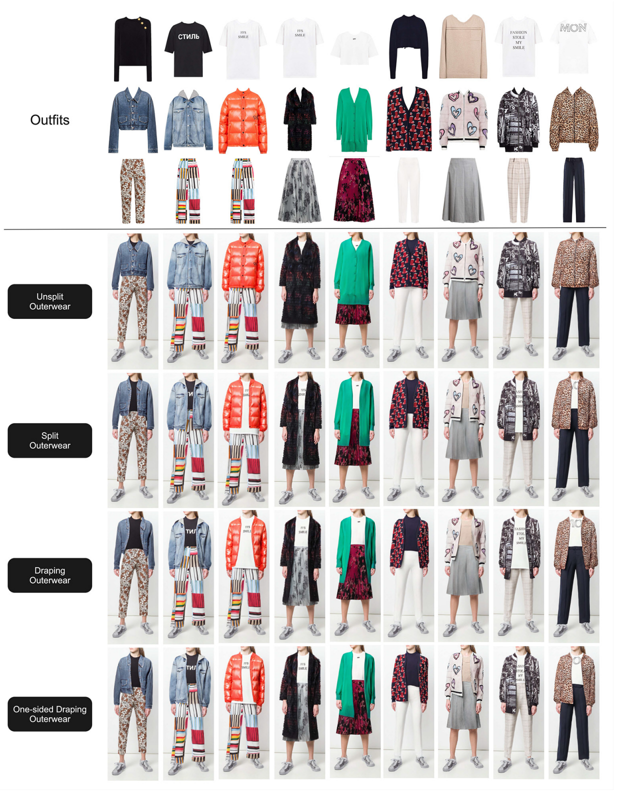

Figure 2. The figure shows a sequence of outfits with outerwear styled differently. We could wear the same jacket as split (unzip or unbuttoned) or non-split (zipped or buttoned) and render it with different drapes. Note that we could produce the same kind of drape for outerwear that are very different (some are knee length while others are cropped). Also, our method is able to preserve very complex patterns in some of the jackets.

2. Related Work

Image-based virtual try-on methods produce an image of a person wearing a reference garment. The main challenge lies in producing high-quality images while faithfully preserving the garment’s identity. Modern image generation networks trained with adversaries can produce high-quality images but have difficulty preserving the exact geometrical patterns (such as logos, prints, etc..) on the garment. Thus, most state-of-the-art VTON pipelines learn a differentiable warper to align the garment onto the person, thus preserving the geometrical patterns and details [2, 5, 6, 8, 10, 12, 13, 15, 18, 20, 21, 25–27, 29, 34, 36, 37, 40, 41]. The warped garment images are provided to an image generator network trained with adversaries to synthesize an image of a person wearing the garment. However, VTON methods generally do not provide a way to change how garments are worn. Yan *et al.* [24] proposed a method using semantically associated landmarks to guide virtual try-on to achieve better rendering quality. However, the type of editing the method support modifies the garment attributes (Figure 1 and 10) rather than how the garment is worn. In contrast, our pipeline can produce high-quality try-on with garment detail preserved and, meanwhile, control the drape of the garment. ze **Image warping processes** apply a spatial transformation to an image and thus can preserve the 2D patterns on a garment. The implementation of the warper varies and recent methods have converged to learning a network that predicts per-pixel appearance flow [6, 15, 18]. Prior warping methods do not provide a convenient way to control the warp (that allows editing the drape of the garment). The warper is usually guided by semantic layouts (pixel maps that define the region of the garment and the body), body pose key points, DensePose (a body pose representation with 3D priors [1,31]), or a combination of these [6,13,15,18,26,27,29,36,37,40]. These representations are either difficult to alter or do not directly impact the drape of the garment. In contrast, our method enables intuitive control of the garment drape using a set of control points embedded with garment semantics. We use rendering policies to modify the control points and our rendering pipeline drapes the garments according to the edits.

Multi-garment virtual try-on is more challenging than the single garment version because the framework needs to appropriately manage the layering between multiple garments and the body. O-VITON [30] first synthesizes a semantic layout to outline the garment interactions and then broadcasts the feature encoding vectors based on the layout. This formulation can manage the interactions between garments well. However, feature encoding vectors cannot preserve structural patterns, resulting in loss of details during rendering ([30] Figure 1, row 3, skirt prints wrong). Meanwhile, OVNet [27] proposed an iterative method of constructing the outfit, swapping one garment at each generation step. This framework is able to preserve the attributes but does not have a solution to coordinate the garments in an outfit. We show that mis-coordination can result in ugly renderings in Figure 6. In contrast, our method applies procedural edits to predicted control points to obtain significant improvement in rendering layered outfits.

Image editing with fashion context yields compelling results in editing the garment drapes but tends to not accurately preserve garment appearance. For instance, SwapNet [33] transfers the style of the garments from one person to another without preserving the visual details ([33] Figure 10 last row logo blurred); Cui *et al.* [7] proposed a recurrent generation pipeline to sequentially dress garments on a person. The method is capable of modifying how garments is worn (e.g., tuck vs. untuck), but lost garment details during the process of encoding and decoding visual featured ([7] Figure 8 row 2 and 3, prints altered); Chen *et al.* [4] and Sukar*et al.* [35] synthesize try-on image of different viewpoints controlled through body poses, but cannot specifically control the drape nor preserve garment identity ([4] Figure 1, 4th outfit white stripes blurred); Liu *et al.* [11] distort the length and shape of the garments through control points to fit on underwear models. However, the method cannot layer multiple garments to form an outfit. Other methods manipulate the garment designs. Some allow editing the garment on a person image by providing simple text description or a set of key words [22, 32, 39, 41]. Others enable edits to the shape, appearance and color through inpainting methods [9, 16, 38]. Fashion++ [20] proposed a framework to perform minimal edit to a specific fashion item to maximize the fashionability of an outfit. Baldrati *et al.* [3] proposed a method to edit fashion image through diffusion model. In contrast, our method can freely change how garments are depicted without changing the garment properties.

3. A Controllable VTON Pipeline

A rendering policy in our pipeline is a procedural description of how an outfit should be worn. The policy can be changed during inference to achieve different drapes of the same garment and different interactions between garments. In this section, we describe an outfit virtual try-on pipeline that is controllable through rendering policies.

A typical VTON pipeline starts with predicting the semantic layout to determine where the garment and body are positioned; then, a warp is predicted to align the garment onto the body; finally, an image generator takes in the warped garment, the layout, and other features to synthesize the final image. To control the output of this pipeline, we need to be able to control how the warp.

To control a VTON pipeline, we use a set of garment control points K, which are key positions that mark the drapes of a garment on a person. The control points can be easily manipulated by simple procedures (describing offsets between points) and can be used to guide the warper. We learn a Control Point Regressor R_c to predict the control points K from neutral garment image features A and body pose key points b_p. Then, we train the warper W and the semantic layout generator G_L to make predictions conditioned on the control points K, as outlined in Figure 4. We show that the garments behave according to changes made to the control points in Figure 5, which is important for our rendering policies to function properly.

The advantage of this setup is that the rendering policy can move these control points around during inference to alter the garment accordingly. The way the information flows in the pipeline makes it natural for the rest of the pipeline to follow the control points. Some garments have "modes" (e.g. open vs. closed outerwear) that require a different set of control points. We train Control Point Regressor R_c to take different style variables Z to predict different "modes".

During inference, we first run the control points regressor to produce a set of control points K for each and every garment in the outfit (Figure 3). Then, we apply a rendering

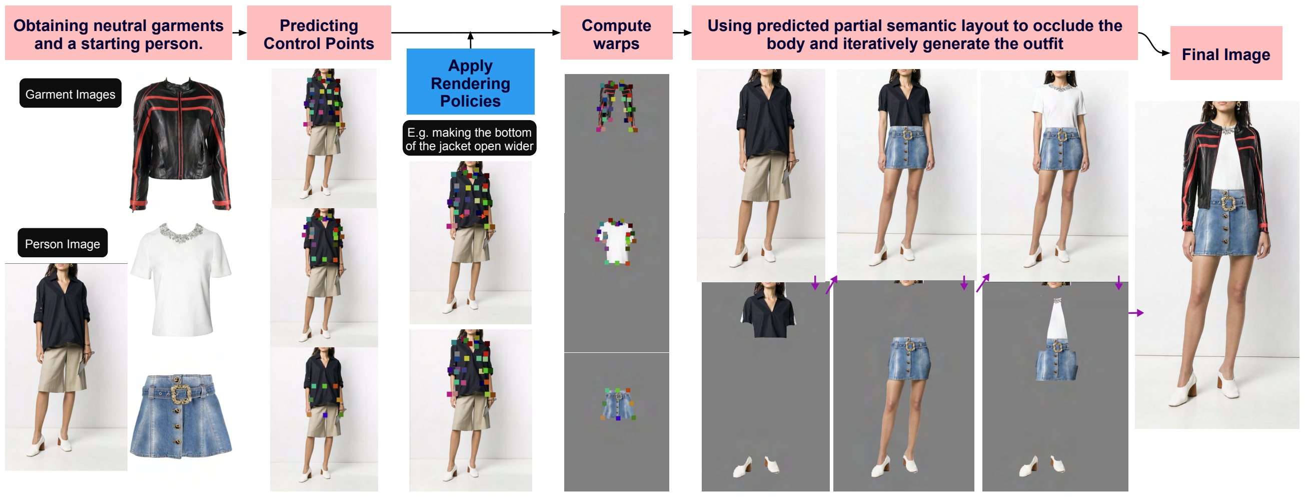
Obtaining neutral garments and a starting person.
Garment Images
Person Image
Predicting Control Points
Apply Rendering Policies
E.g. making the bottom of the jacket open wider
Compute warps
Using predicted partial semantic layout to occlude the body and iteratively generate the outfit
Final Image

Figure 3. This figure shows an overview of our outfit generation pipeline. We first predict the control points of each garment based on a neutral garment image and the body pose of the person. Then, we can modify the drape of each garment by applying a certain rendering policy to adjust the control points. Subsequently, we predict each garment warp based on the edited control points. Finally, we generate each garment on the person sequentially based on the layering (garments beneath are generated first). During the process, the predicted semantic layout is used to occlude the specific image region we are generating for each garment.

policy to modify the control points of all garments. Having all the garment control points available is advantageous because certain types of style edits require coordinating multiple items in an outfit. Then, we iteratively generate each garment on the person by applying W, G_L, and the image generator G_I. The inference pipeline after computing the warp resembles the iterative inference process of Li *et al*. [27]. Please find the complete set of notations and definitions in the Appendix.

3.1. The Control Points

K is a set of control points which are 2D coordinates on the image of the person. Each control point has a specific semantic meaning (e.g., left shoulder, right inner sleeve, etc..). Using control points to guide the warp and the layout is desirable because it is intuitive to understand how these edits would change the garment. For example, increasing the distance of the control points between two sides of a jacket will widen the split of the jacket, as in Figure 5. It is also important that the control points embed garment semantics. Having such a property allows us to make edits on an instance of garment and apply the edits to other garments of the same category.

The DeepFashion dataset already contains garment semantic key points annotation on the human body [14]. We took their pre-trained network and predicted the key points for every model image b in our training dataset. The original key point annotation has redundancy (e.g., there are separate sets of collar key points for t-shirt and shirt). We merge the key points with identical semantic meaning and obtain a list of 49 unique control points K (details in the Appendix).

3.2. Training Procedure

The system consists of the Control Points Regressor R_c, the Semantic Layout Generator G_L, the Warper W, and the Image Generator G_I, as shown in Figure 4.

The Control Points Regressor R_c takes in the body pose representation b_p, the neutral garment features A and the style variables Z, and outputs the garment control points $K\prime = R_c\left(A, b_p, Z\right)$. The style variable $Z = [z_1, z_2, ..., z_n]$ is a control parameter where each index is set to control a discrete style (e.g., z_1 is set to 0 for outerwear to be open and 1 it to be closed). Because the styles to control by Z have to be labeled in the training data, we recommend only using style parameters Z to control styles that are commonly observed in the training data or require different sets of control points (e.g. closed vs. open jacket). For this paper, we learned two distinct styles through Z, tuck vs. untuck and outerwear open versus closed.

R_c consists of a ResNet32 [17] connected to a fully-connected layer. Z is broadcasted to a 2D plane and concatenated with other inputs before feeding them into R_c. The output of the fully-connected layer is reshaped into $N \times n \times 2$ where N is the batch size and n is the number of control points. The network is trained using a $\mathcal{L}_1$ loss and $\mathcal{L}_2$ loss computed between K and $K\prime$. In addition, we compute a structural consistency loss $\mathcal{L}_s$ that penalized errors in pair-wise distance between the control points. This is important to prevent the structure of the garment from being misrepresented or distorted due to minor displacement of each individual control point. We compute a matrix of

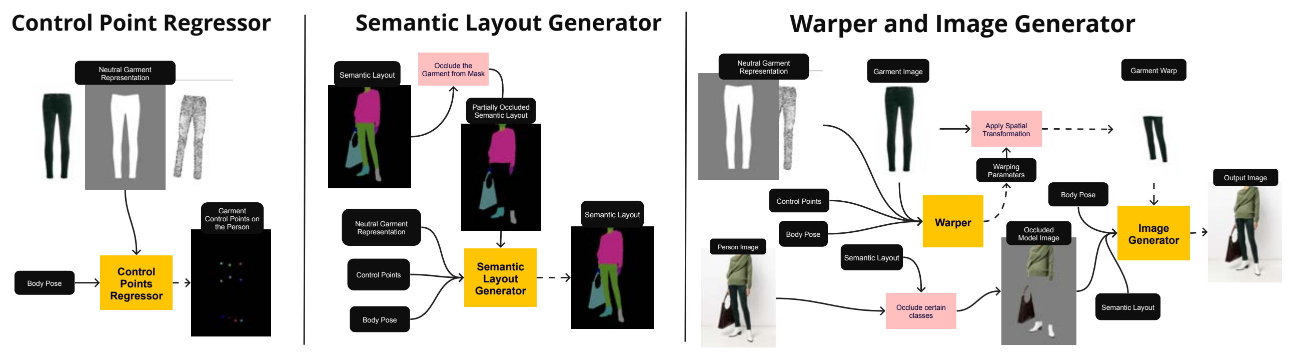

Figure 4. The figure shows an overview of the training process. The Control Point Regressor R_c first is trained to predict the control points based on body pose and neutral garment features. The Semantic Layout Generator G_L is trained to reconstruct the partially occluded layout using the control points and other features. The Warper W predicts a warp to align the neutral garment onto the body based on the control points and the body pose. The predicted warp and other features are fed into the Image Generator G_I to synthesize the output image. The Warper is trained jointly with the Image Generator. The dashed lines indicate the path of backpropagation during training.

the distance between each pair of points

$$D = \begin{bmatrix} d(k_1,k_1) & d(k_1,k_2) & \dots & d(k_1,k_n) \\ d(k_2,k_1) & d(k_2,k_2) & \dots & d(k_2,k_n) \\ \vdots & \vdots & \vdots & \vdots \\ d(k_n,k_1) & d(k_n,k_2) & \dots & d(k_n,k_n) \end{bmatrix}$$

where $d(k_i,k_j) = \sqrt{|x_i - x_j|^2 + |y_i - y_j|^2}$, and train the network to minimize the structural consistency loss $\mathcal{L}_s = ||D - D'||$. The total training loss for R_c can be written as $\mathcal{L}_{R_c} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_s$ where λ_1, λ_2 and λ_3 are the weights for each component.

Note that the network predicts a value for every control point but not every control point is relevant for every type of garment (e.g., tops don't need trouser leg control points). Thus, when computing the training loss, we mask out the irrelevant control points based on the garment's category.

The **Semantic Completion Generator** G_L predicts the human parsing indicating the pixel region of garments and body parts on the generated try-on image. The formulation of G_L largely follows other VTON pipelines [6, 21, 27, 37], except that it is conditioned on the control points. To train G_L, we obtain the neutral garment features A, the model's body pose b_p, the garment control points K, and the partially occluded semantic layout mask $\hat{b}_m$. K is plotted into a 2D map where each channel contains a single control point and is then concatenated with the other inputs. G_L's training objective is to reconstruct the original semantic layout b_m from $\hat{b}_m$. The occluded mask $\hat{b}_m$ is obtained by replacing parts of b_m by background class through an occlusion function $\hat{b}_m = f_o\,(b_m, a_t)$. The occlusion function f_o operates based on the garment category a_t. f_o is different per garment category and guided by the following rules: (1) f_o always replaces the region of the specified garment category a_t; (2) f_o also replaces the category of skin classes that are directly connected with a_t. For example, when a_t is a top, f_o removes the arm layouts and the neckline layout, but not the legs layout. G_L is trained through pixel-wise

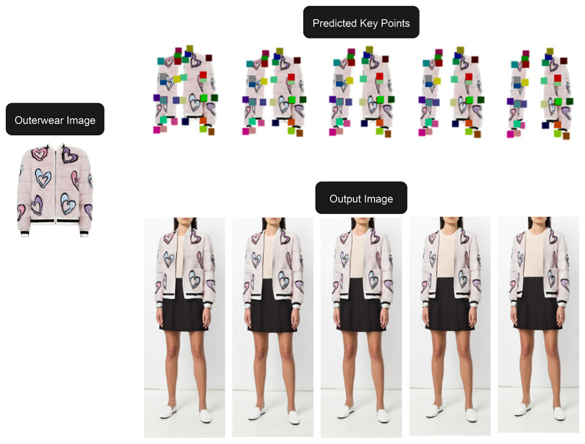

Figure 5. We perform an interpolation of the control points to show that our rendering pipeline can effectively control how garment drapes. In this example, we move the control points by a small offset every step to gradually open the outerwear and observe that the drape of the outerwear closely follows the control point in every plot. Results demonstrate that the control points are highly effective in controlling the garment drapes.

Cross-Entropy loss and adopts a U-Net architecture following prior arts [27].

The Garment Warper W aligns A onto the person following the control points. The Image Generator Network G_I takes in the warped garment A^w, the partially occluded semantic layout $\hat{b}_m$, and other features to produce the final output image b'. W takes in the the body pose b_p, the neutral garment features A and the control points K, and outputs the transformation parameters $\theta = W\,(b_p, A, K)$. We compute the spatial transformation through θ and obtain the warped garment features $A^w = a^w, a^w_m, a^w_c, a^w_e$ (a^w is the warped garment image; a^w_m is the warped garment mask; a^w_c is the warped garment cropped mask; a^w_e is the warped garment edge map.). More details about the warper training are available in the Appendix.

The Image Generator G_I produces the final try-on image $b' = G_I(A^w, b_p, b_o, b_m^g)$ based on the warped garment features A^w, the body pose representation b_p, the occluded person image b_o and the input semantic mask $b_m^g = b_m \odot (1 - b_g)$ without the to try-garment's layout b_g. See the Appendix for how b_o is obtained. The Image Generator G_I architecture is a U-Net as the skip connections provide an easy way to copy the provided input image. The learning objective is to produce an image that resembles the ground truth model and appears realistic. We train G_I with an $\mathcal{L}_1$ loss and an $\mathcal{L}_{perc}$ perceptual loss [23] computed between b' and b. In addition, we train G_I with an adversarial loss $\mathcal{L}_{adv}$ to encourage realism of the output image following prior arts [5, 6, 12, 13, 13, 15, 18, 27, 36, 37].

The combined training loss for W and G_I becomes $\mathcal{L}_{G_f} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_{perc} + \lambda_3\mathcal{L}_{adv} + \lambda_4\mathcal{L}_w$ where λ_1, λ_2, λ_3 and λ_4 are the weights the loss.

4. Authoring Rendering Policies

A rendering policy consists of procedural expressions describing offsets to the control points for each set of garment types, possibly in terms of other control points on the same garment or other garments in the outfit. To create a procedure, one starts with an outfit rendered using the predicted control points K'. One edits K' to achieve a certain drape and reproduces the render after each edit until one is satisfied with the drape. Then, one looks at the manual edits and comes up with a set of rules that modifies the control points' coordinates to achieve the same effect on the instance. The rules often come in the form of relative position between control points on the same garment or different garments in an outfit. For example, one could shift the vertical coordinate of every control point on a skirt upward by a constant to wear the skirt at a higher waistline position. While authoring a procedure, one would test the building procedure on a few different outfit examples. Once the procedure is finalized, it can be applied to any other garment of the same category and render the outfit in the specified style. A rendering policy that achieves certain drape may contain many procedures. We show that the procedures generalize well.

4.1. Performing Style Edits

Rendering policy can be used to simulate different styles of wearing an outfit, In Figure 3, we create a style that opens the lower part of the jacket wider – we first predict the control points of the jacket setting Z as "open outerwear"; then we apply a procedure that moves the control points on each side of the split away from each other by a constant to simulate a wider split on the lower part. When computing the warp for the jacket, we compute the warp for each side separately (see Appendix for more details). Other types of edits require coordinating multiple garments in an outfit. For example, to achieve the front tuck in Figure 1 row 4, we use a procedure that sets the middle torso control point of the shirt to be at the same height as the waistline control point of the bottom. Every style in Figure 1 and 2 are created using another variation of such procedures. For each style, we constructed the policy by testing on less than 3 examples. Results in Figure 1 and 2 show examples of these policies generalizing well to other outfits (not the ones we tested on).

4.2. Coordinating Garments in Complex Outfits

Users sometimes will layer multiple garments in an outfit that creates complicated garment interactions. This presents a serious challenge to a VTON pipeline because minor failures of coordination between garments can produce ugly results as in Figure 6. These errors occur because the control points for individual garments are predicted separately. Our rendering policies are well-adapted to prevent these minor failures and they generalize well.

A common type of error is that the garment beneath sticks out. For example, skirts sometimes unnaturally stick out of the coat as in Figure 7. An example correction procedure to address this type of error is as follows: (1) draw a line using the two control points on each side of the coat; (2) check if the skirt's control points lie within the region bounded by the two lines; (3) if some control points do not, shift the horizontal coordinate of these control points such that they are between the two lines; (4) rendering the skirt with the corrected control points, and the skirt does not stick out as in Figure 7. Other errors can be addressed by different procedures of a similar type. All the procedures are stacked together, resulting in a rendering policy that coordinates layered garments.

The steps for building a complete rendering policy for garment coordination is as follows: (1) one starts testing with random outfits and identifies failures; (2) one looks at the common failures and creates correction procedures to address them (like the jacket-skirt example above); (3) one renders the outfit with the updated procedure applied and keep iterating the rules until the error does not occur; (4) one start building correction procedure for the next type of failure. (5) All the correction procedures are stacked together to form the rendering policy. Experiments in Section 5.1 show that a policy built through examining a few examples can significantly improve the rendering quality of general outfits.

5. Experiments

We train our network on the OVNet dataset [27], which contains garments of different categories worn in different styles. See the Appendix for details on data preparations.

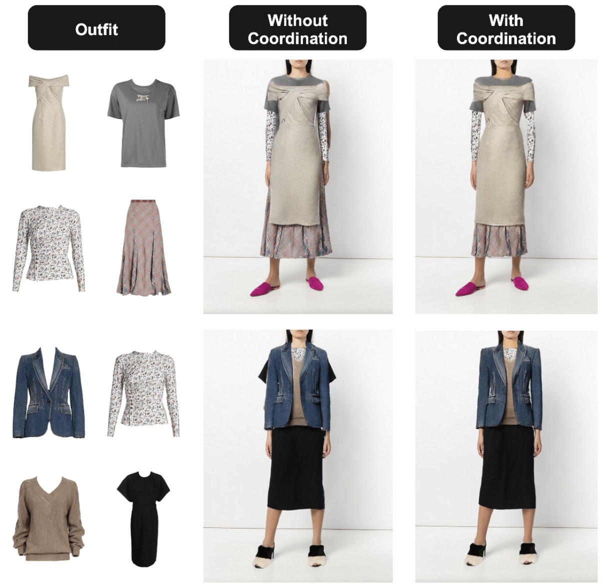

Figure 6. The figure compares outfits rendered with or without using the rendering policies that coordinate the garments. When each garment warp is predicted individually, we observe minor errors that cause obvious artifacts (e.g., sleeves sticking out and overlapping). Our method fixes the problem by applying a rendering policy that coordinates the relative positions of garments. Note that the same rendering policy is used for both examples.

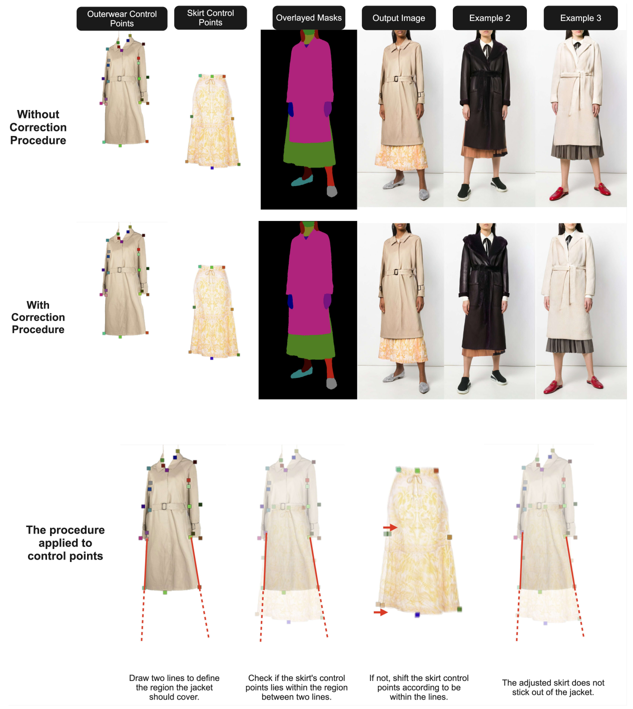
Outerwear Control Points
Skirt Control Points
Overlayed Masks
Output Image
Example 2
Example 3
Without Correction Procedure
With Correction Procedure
The procedure applied to control points
Draw two lines to define the region the jacket should cover.
Check if the skirt's control points lies within the region between two lines.
If not, shift the skirt control points according to be within the lines.
The adjusted skirt does not stick out of the jacket.

Figure 7. An example procedure that coordinates the control points between jackets and skirts to avoid the skirt sticking out. In the first row, the skirt sticks out of the long jacket, as they were predicted separately. On the second row, a correction procedure is applied and the jackets cover the skirts naturally. The correction procedure moves the skirt's control points to align with those of the outerwear. The bottom row illustrates how the correction procedure works. This procedure is easy to come up with and generalizes to any outerwear-skirt pair. See details in section 4.2.

5.1. Quantitative Evaluations

Quantitative evaluation results in Table 1and 2 show that our method can produce images with quality that match the state-of-the-art outfit visualization methods while having better control on garment drapes. Note that our base rendering pipeline (without rendering policy) also outperforms the other baseline methods, suggesting that our proposed pipeline is compelling (See Table 2).

Can our pipeline render different styles that people recognize? To show that our method can generate different styles, we sample real model images wearing outfits of 8 different styles (10k each) and compute FID [19] score against 10k generated images conditioned to exhibit the same style. Table 1 shows the results for each style-specific subset. Our method yields a smaller FID [19] score than the baseline for every style, meaning that our method produces images that better capture these individual styles. Note that the baseline method does not describe a reliable way to control tuck vs. untuck other than changing the order of the top and bottom (as in [27] Fig 3). Thus, we adopt such a procedure when running the baseline.

Do rendering policies generalize? We designed our rendering policies to be generalizable – meaning one can author the policy through testing on a few examples and the policy applies to any other outfits containing the same type of garments. To validate, we went through 20 outfit examples and come up with a correction policy (Section 4.2) that contains 27 correction procedures. Then, we apply this policy to randomly sampled outfits (from 2 to 7 garments per outfit; 10k for each set). We compute the FID against a set of 10k real photographs of models wearing outfits. Results in Table 2 show that applying the rendering policy yields a consistently better outcome than without policy. This shows that the rendering policy created through our procedure generalized well. In fact, the rendering policy is able to process complex outfits like the ones in Figure 6 and the small adjustments to control points remove the original artifacts.

Note that the FID score increases as the number of garments in the outfit increases because all rendering pipelines tend to make more errors with more complex outfits.

5.2. Qualitative Evaluation

In Figure 1 and 2, we demonstrate a variety of style edits that our system can support. Example edits include untucking or tucking the top in different ways, wearing the outerwear closed or open with different kinds of drapes, wearing bottoms at different waistline heights, etc. In addition,

Table 1. This table compares the FID Score [19] between our method and state-of-the-art outfit visualization. Method 1 (M1) is the original implementation of OVNet [27]. Method 2 (M2) is the OVNet framework combined with the Flow Warper from [6]. In each row, the FID scores are computed against real outfits worn in each individual style. (t is top; b is bottom; o is outerwear) Results show that our method outperforms prior work on all styles.

Gender	Style	M1 ↓	M2 ↓	Ours ↓
Female	dress	15.6	16.2	**14.2**
Female	t+b untuck	26.4	21.3	**19.5**
Female	t+b tuck	22.4	22.6	**20.2**
Female	t+b+o closed	21.2	21.7	**19.4**
Female	t+b+o open	23.9	20.8	**19.0**
Male	t+b	20.3	20.9	**18.8**
Male	t+b+o closed	21.5	19.5	**19.1**
Male	t+b+o open	22.1	18.0	**17.3**

Table 2. We build outfits that consist of a different number of garments and compute FID Score [19] against a dataset of real outfits. The rendering pipeline is more likely to make mistakes as the number of garments in the outfit increases. Results show that our rendering policies largely improve the multi-garment generation performance, especially as the outfits increase in complexity. No. of items indicates the number of garments in the outfits; we increase this number to simulate more complex outfits; Ours(w/o p) is our method without applying the rendering policies.

No. of items	M1 ↓	M2 ↓	Ours(w/o p) ↓	Ours ↓
2	16.5	13.7	13.1	**12.6**
3	19.3	18.7	17.6	**15.9**
4	22.6	21.4	20.1	**17.1**
5	25.2	23.4	22.3	**18.9**
6	29.6	25.9	23.8	**19.7**
7	31.0	26.8	24.1	**20.1**

we show that each of the style edits applies consistently to many garments in the same category and appears consistent even when the garments have different properties (e.g., the length and shape of the outerwear differ). This is made possible by using garment control points that capture category-level garment semantics.

The qualitative examples also show the properties of the garments are preserved after the style edits and are not impacted by the way garments are worn. Pay attention to the red blocks (column 6), the colored hearts (column 7), the street painting (column 8), and the leopard prints (column 9) in Figure 2. These distinct features are preserved when the outerwear is open or altered in different ways.

Figure 6 shows that using control points to coordinate outfits can largely avoid artifacts caused by a lack of coordination between garments in an outfit.

To evaluate how the control points affect the results, we perform interpolation by moving the control points slightly after each step and comparing the differences. As shown in Figure 5, the silhouette of the outerwear exactly follows the control point. The results show that the control points are very effective at controlling the garment drapes.

In addition, qualitative examples show our method can control a specific garment without affecting other garments in the outfit. As shown in Figure 2, the method only intends to edit the outerwear and preserves the look of the other garments. The visible parts of the top and the bottoms remain identical, besides the addition of some shading that is necessary to produce a realistic look.

5.3. User Studies

User studies support the idea that (a) our drape edits do not change the identity of a garment and (b) that they do change the way the garment is worn.

In the first study, we show subjects pairs of outfits rendered from our method and ask the subjects if a garment in the pair of outfits is identical. The questions contain rendered pairs of the same garment worn in different ways and rendered pairs of different garments that were chosen to look visually similar. We asked 44 respondents to answer 22 questions, where 15 were pairs of the same garment and 7 were pairs of different garments. The respondents had an overall accuracy of 85.0%, suggesting the subjects perceived that most style edits did not alter garment identity.

In the second study, we show the subjects a rendered outfit and ask the person to choose the style that best describes the look from a set of options. For example, a subject may be asked to choose whether a top+bottoms outfit is either untucked, fully-tucked, front-tucked, side-tucked, or half-tucked. Subjects are shown examples of each style option. We asked 44 respondents to answer 22 styling questions. The respondents had an overall accuracy of 80.1%, indicating that subjects are able to identify the style in most cases. Details of the study are included in the Appendix.

6. Conclusion & Discussion

We propose the first outfit try-on system that allows garments to be worn in different ways while producing try-on images that preserve state-of-the-art quality. Because our control layer embeds garment semantics, our method allows style edits made on an example garment to be applied to style other garments of the same category. Evaluation results show that the control mechanism can effectively alter the way a garment drapes while preserving its identity. The system provides a powerful tool for users to experiment with outfit drapes and for professionals to style different garments appropriately.

References

[1] Rıza Alp Guler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[2] Shuai Bai, Huiling Zhou, Zhikang Li, Chang Zhou, and Hongxia Yang. Single stage virtual try-on via deformable attention flows. In *Proceedings of the European Conference on Computer Vision*, 2022. 2

[3] Alberto Baldrati, Davide Morelli, Giuseppe Cartella, Marcella Cornia, Marco Bertini, and Rita Cucchiara. Multimodal garment designer: Human-centric latent diffusion models for fashion image editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 3

[4] Chieh-Yun Chen, Ling Lo, Pin-Jui Huang, Hong-Han Shuai, and Wen-Huang Cheng. Fashionmirror: Co-attention feature-remapping virtual try-on with sequential template poses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 3

[5] Seunghwan Choi, Sunghyun Park, Minsoo Lee, and Jaegul Choo. Viton-hd: High-resolution virtual try-on via misalignment-aware normalization. In *CVPR*, 2021. 2, 6

[6] Ayush Chopra, Rishabh Jain, Mayur Hemani, and Balaji Krishnamurthy. Zflow: Gated appearance flow-based virtual try-on with 3d priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2, 3, 5, 6, 8

[7] Aiyu Cui, Daniel McKee, and Svetlana Lazebnik. 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 (ICCV)*, 2021. 1, 3

[8] Haoye Dong, Xiaodan Liang, Ke Gong, Hanjiang Lai, Jia Zhu, and Jian Yin. Soft-gated warping-gan for pose-guided person image synthesis. In *NeurIPS*, 2018. 2

[9] Haoye Dong, Xiaodan Liang, Yixuan Zhang, Xujie Zhang, Xiaohui Shen, Zhenyu Xie, Bowen Wu, and Jian Yin. Fashion editing with adversarial parsing learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 3

[10] Benjamin Fele, Ajda Lampe, Peter Peer, and Vitomir Struc. C-vton: Context-driven image-based virtual try-on network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022. 2

[11] Xin Gao, Zhenjiang Liu, Zunlei Feng, Chengji Shen, Kairi Ou, Haihong Tang, and Mingli Song. Shape controllable virtual try-on for underwear models. 2021. 3

[12] Chongjian Ge, Yibing Song, Yuying Ge, Han Yang, Wei Liu, and Ping Luo. Disentangled cycle consistency for highly-realistic virtual try-on. In *CVPR*, 2021. 2, 6

[13] Yuying Ge, Yibing Song, Ruimao Zhang, Chongjian Ge, Wei Liu, and Ping Luo. Parser-free virtual try-on via distilling appearance flows. In *CVPR*, 2021. 2, 3, 6

[14] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *CVPR*, 2019. 4

[15] Xintong Han, Xiaojun Hu, Weilin Huang, and Matthew R. Scott. Clothflow: A flow-based model for clothed person generation. In *ICCV*, 2019. 2, 3, 6

[16] Xintong Han, Zuxuan Wu, Weilin Huang, Matthew R. Scott, and Larry S. Davis. Compatible and diverse fashion image inpainting. 2019. 3

[17] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016. 4

[18] Sen He, Yi-Zhe Song, and Tao Xiang. Style-based global appearance flow for virtual try-on. In *CVPR*, 2022. 2, 3, 6

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 7, 8

[20] Wei-Lin Hsiao, Isay Katsman, Chao-Yuan Wu, Devi Parikh, and Kristen Grauman. Fashion++: Minimal edits for outfit improvement. In *ICCV*, 2019. 1, 2, 3

[21] Thibaut Issenhuth, J. Mary, and Clément Calauzènes. Do not mask what you do not need to mask: a parser-free virtual try-on. *ECCV*, 2020. 2, 5

[22] Yuming Jiang, Shuai Yang, Haonan Qiu, Wayne Wu, Chen Change Loy, and Ziwei Liu. Text2human: Text-driven controllable human image generation. *ACM Transactions on Graphics (TOG)*, 2022. 3

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 6

[24] Hui Zhang Keyu Yan, Tingwei Gao and Chengjun Xie. Linking garment with person via semantically associated landmarks for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 2

[25] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. A generative model for people in clothing. In *ICCV*, 2017. 2

[26] Sangyun Lee, Gyojung Gu, Sunghyun Park, Seunghwan Choi, and Jaegul Choo. High-resolution virtual try-on with misalignment and occlusion-handled conditions. In *Proceedings of the European Conference on Computer Vision*, 2022. 2, 3

[27] Kedan Li, Min Jin Chong, Jeffrey Zhang, and Jingen Liu. Toward accurate and realistic outfits visualization with attention to details. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 4, 5, 6, 7, 8

[28] Yifang Men, Yiming Mao, Yuning Jiang, Wei-Ying Ma, and Zhouhui Lian. Controllable person image synthesis with attribute-decomposed gan. In *Computer Vision and Pattern Recognition (CVPR), 2020 IEEE Conference on*, 2020. 1

[29] Davide Morelli, Matteo Fincato, Marcella Cornia, Federico Landi, Fabio Cesari, and Rita Cucchiara. Dress Code: High-Resolution Multi-Category Virtual Try-On. In *Proceedings of the European Conference on Computer Vision*, 2022. 2, 3

[30] Assaf Neuberger, Eran Borenstein, Bar Hilleli, Eduard Oks, and Sharon Alpert. Image based virtual try-on network from unpaired data. In *CVPR*, 2020. 2, 3

[31] Natalia Neverova, Riza Alp Güler, and Iasonas Kokkinos. Dense pose transfer. In *ECCV*, 2018. 3

[32] Martin Pernuš, Clinton Fookes, Vitomir Štruc, and Simon Dobrišek. Fice: Text-conditioned fashion image editing with guided gan inversion, 2023. 3

[33] Amit Raj, Patsorn Sangkloy, Huiwen Chang, James Hays, Duygu Ceylan, and Jingwan Lu. Swapnet: Image based garment transfer. In *ECCV*, 2018. 1, 3

[34] I. Rocco, R. Arandjelović, and J. Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, 2017. 2

[35] Kripasindhu Sarkar, Vladislav Golyanik, Lingjie Liu, and Christian Theobalt. Style and pose control for image synthesis of humans from a single monocular view. *ArXiv*, 2021. 3

[36] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, and Liang Lin. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 6

[37] Han Yang, Ruimao Zhang, Xiaobao Guo, Wei Liu, Wangmeng Zuo, and Ping Luo. Towards photo-realistic virtual try-on by adaptively generating↔preserving image content. In *CVPR*, 2020. 2, 3, 5, 6

[38] Li Yu, Yueqi Zhong, and Xin Wang. Inpainting-based virtual try-on network for selective garment transfer. *IEEE Access*, 2019. 3

[39] Kaiduo Zhang, Muyi Sun, Jianxin Sun, Binghao Zhao, Kunbo Zhang, Zhenan Sun, and Tieniu Tan. Humandiffusion: a coarse-to-fine alignment diffusion framework for controllable text-driven person image generation, 2022. 3

[40] Xie Zhenyu, Huang Zaiyu, Dong Xin, Zhao Fuwei, Dong Haoye, Zhang Xijin, Zhu Feida, and Liang Xiaodan. Gp-vton: Towards general purpose virtual try-on via collaborative local-flow global-parsing learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 2, 3

[41] Shizhan Zhu, Sanja Fidler, Raquel Urtasun, Dahua Lin, and Change Loy Chen. Be your own prada: Fashion synthesis with structural coherence. In *CVPR*, 2017. 1, 2, 3