POSE TRANSFER USING A SINGLE SPATIAL TRANSFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we address the problem of pose transfer. The goal is to generate a source image in a new target pose. The pose is already provided by a set of spatial landmarks. The transfer function is directly estimated from the difference between the landmarks given in the new target pose and the landmarks of the source image. Existing methods perform the task using two specialized networks, one to move the patches of the source sample and the other one to generate the new patches that are not visible in the source image. Contrary to these strategies, we develop an end-to-end trainable neural network that learns to estimate both these visible and invisible parts using a simple warping module. In other words, we propose a flow estimation method that not only displaces the patches to their new locations but also generates new pixels that are not visible in the source image, all in an unsupervised manner without the need for a ground-truth flow map. In this way, moving the patches and introducing new parts are unified into a single network, ensuring that an overall solution is achieved for these two mutual tasks. Additionally, this method avoids the need for a human observer to determine a trade-off between the performance of the two separated networks, thus avoiding a cartoonish addition of the new parts to the visible areas. Extensive experiments demonstrate the superiority of our method over state-of-the-art algorithms. We conduct our experiments on two well-known datasets: Deepfashion and Market1501.

1 INTRODUCTION

Pose transfer is one of the fundamental problems in computer vision with many applications in 3D reconstruction, image animation, and virtual reality. The main objective is to generate a source image in a new target pose. This pose is provided by a set of spatial keypoints.

The transfer function is directly estimated from the difference between the given keypoints in the target pose and the keypoints of the source sample. Existing methods propose to estimate this function using three consecutive modules. The first one receives a source image along with the source pose and the target pose and estimates a flow map that displaces the patches of the source image to their new location in the target pose. The second one receives this warped image along with the target pose and learns to generate the remaining patches that are invisible in the source sample but are newly introduced in the target pose. The final image is generated by applying a refinement module on the concatenation of these so-called *visible* and *invisible patches*.

Despite promising results, these methods cannot be trained in an end-to-end way. This is due to the concatenation of the features in the refinement module. As a spatial transformer, a warping module (the first one) will only learn when the spatial transformation is the single factor that influences the loss function Jaderberg et al. (2015). However, this is violated by concatenating the target pose to the warping features. On the other hand, this concatenation is required to add the invisible parts to the displaced patches. This means that the warping module must be trained separately from the rest of the network, which comes with its own disadvantages: (1): it needs a human observer to validate a trade-off between the performance of the warping module and the performance of the remaining parts of the network. This is required as the warping aims to fit two images (the source and the target images) which, due to the invisible parts of the samples, are not fully comparable. This leads to an ill-posed problem with no ideal solution, so that the module never converges to the global minimum.

Contrary to these methods, we propose a single-stage end-to-end trainable neural network where a single warping module is responsible for both displacing the patches and generating the parts that are newly introduced in the target pose. This ensures a single global solution is achieved for these two complementary tasks, which in turn avoids a cartoonish addition of the invisible patches to the visible parts[1].

The warping module of the existing pose transfer networks is in fact a deep neural network that estimates the parameters of an affine transformation. This way, the network can easily handle any long displacement of the samples. However, it reduces the learnable parameters of the flow estimator to a few numbers. This hinders the application of this method for estimating a complex deformation.

To address this problem, we introduce a constrained spatial transformation. It is a pixel-wise flow estimator in which the sampling locations are individually estimated for each pixel of the target image. In this way, the spatial transformation is endowed with enough parameters, required for estimating a complex transformation of the samples. The flow is directly estimated from the correlation of the pose maps. This allows us to accurately encode any short or long-range displacements which can not be achieved by a simple convolutional network.

Our transformation not only displaces the visible patches (to their new position in the target pose) but also introduces the invisible parts to the warped version of the sample (by sampling from the most similar areas of the existing patches in the source image). To do so, we alternate between two different tasks. (1) we enforce the flow estimator to estimate an affine function (of the displacement) if the target pose is a linear transformation of the source pose. (2) we enforce the flow estimator to create a warped image that best fits the target sample.

In the following, we list the reasons why these two tasks can serve our goals in (a) recognizing the visibility of the pixels in an unsupervised way, (b) keeping the neighborhood of the pixels in the visible areas, and (3) introducing the invisible parts to the warped image, all by estimating a single flow map:

- An affine transformation is a *linear function* that preserves the neighborhood of pixels
- A region is *visible* (in both the source and the target pose) only when it is transferred by a *linear function*.
- The warped image is the only feature map that is used for generating the target image. We do not benefit from any extra *guiding* map. This encourages the network to include all the visible and the invisible parts into this single warped image.
- We enforce the flow function to be an affine transformation where the entire target pose is a linear transformation of the entire source pose (all the pixels are linearly transferred). However, as a pixel-wise transformation, the flow function learns to hold this affinity even by parts. Therefore, a *local* part of the flow map can be estimated by an affine transformation if it is linearly displaced[2] to the target pose. This allows the flow estimation to keep the locality of the pixels just for the visible areas (whose transformations are mostly linear) and to use an unconstrained transformation for the remaining parts.

2 Related work

Pose transfer: There are two different approaches for generating a novel pose of a human in an RGB image: (i) parametric methods Thies et al. (2016); Corona et al. (2021) that learn to fit a 3D model on the RGB sample and then render the model from a novel view point with a different pose. They have been so far applied to face Thies et al. (2016) and body Corona et al. (2021). Despite the memory advantage and remarkable control over the pose of the sample, these methods suffer from certain disadvantages including the difficulty of fitting a 3D model on a 2D image. Moreover, they have great difficulty with the lack of fine-grained details in the generated samples. This arises from

[1]In a two-stage model, the warping does not contribute to the adversarial training of the network and therefore to the photorealism of the generated samples. This causes a cartoonish addition of the invisible parts to the visible areas.

[2]and thus is visible in both the source and the target image

the generic 3D model that is shared between all the RGB samples. (ii) non-parametric models Liu et al. (2021); Tang et al. (2020); Zhang et al. (2021); Lv et al. (2021); Tang & Sebe (2021). These methods are also known as exemplar-based techniques. The main idea is to remove the need for a predefined 3D model. Rather, they propose to reconstruct the new images using a deep neural network that is guided by the target pose of the samples. The pose can be represented by a set of keypoints Ren et al. (2020); Tang et al. (2020); Zhang et al. (2021), edge maps Ren et al. (2020), or segmentation maps Liu et al. (2021); Zhang et al. (2021). Unlike the parametric models, these methods are proven to be more effective in generating the fine-grained textures of the samples. Despite promising results, they have some difficulties to generate the textures that are not among the training samples. This hinders the scalability of these methods for the vast majority of human garments. The challenge can be well addressed by introducing a warping module in the latent space of these networks. This module is supervised by an estimation of the flow map between the pose of the source and the target samples Ren et al. (2020); Siarohin et al. (2019b;a; 2018); Zhang et al. (2021). However, these methods have significant difficulties with an accurate estimation of the flow maps, especially between two completely different poses.

Flow estimation: Unsupervised flow estimation Ren et al. (2017); Sabour et al. (2021); Stone et al. (2021); Luo et al. (2021) is one of the most difficult tasks in computer vision, especially when estimated from two very sparse sets of keypoints. The task is usually simplified by reducing the flow function to a simple parametric model like first-order affine transformation Siarohin et al. (2018; 2019b;a) or thin spline transformation Zhao & Zhang (2022). In this case, the new position of each pixel is determined by applying a simple algebraic operation on its current coordinates. There are some other strategies to estimate the flow field from the UV maps Albahar et al. (2021); Sarkar et al. (2021), or 3D meshes Li et al. (2019). Compared to the keypoints, they provide a fine description of the custom shape, which facilitates the estimation of the flow maps. However, these representations are not easily accessible for an unseen pose that does not exist among the training samples. By contrast, keypoints can be easily provided by drawing a few dots on an empty scene which is highly convenient for many applications, making it the *de facto* standard of the pose transfer networks.

Unlike these methods, we propose a nonparametric strategy for unsupervised flow estimation which handles any highly complex deformation of the samples. Our model offers the first strategy that exploits the warping flow to generate the invisible parts of the samples. This allows to estimate the *visible* patches and generate the *invisible* ones in a single warping module, thus facilitating the end-to-end training of the flow estimation in a pose transfer network, which has not yet been investigated in the literature.

3 Our method

Our main goal is to transfer the pose of a person, in an image, to a given target pose. To do so, we propose a fully warping-based strategy where a warped version of the source image is the only feature map that is used to generate the sample in its target pose. In this way, estimating the flow map and generating the target sample are both included in a single network that can be trained in an end-to-end way. The flow (warping) map is estimated from the collection of the source pose and the target pose of the sample. Each pose is an empty image where the location of body joints is indicated by a set of small Gaussian envelopes.

The typical use of a warping module is to displace the patches of an image to their target position. However, unlike this strategy, we aim to develop a warping method that not only displaces the patches but also generates the parts that are not visible in the source image but newly introduced in its target pose. In this way, we avoid the need for an extra *guiding map* to indicate the location of the *invisible parts*, which is important for the end-to-end training of the network. Indeed, this approach prevents the network to be confused whether a decrease in the loss function comes from the displacement of the patches or results independently from the *guiding map*.

Figure 1 shows the overall pipeline of our method. It includes two different modules: (1) a warping module $\mathcal{W}$ that learns to estimate a flow map between the pose of the source sample and the target pose, and (2) a refinement network $\mathcal{R}$ that receives the warped version of the source image and converts it to a photorealistic sample.

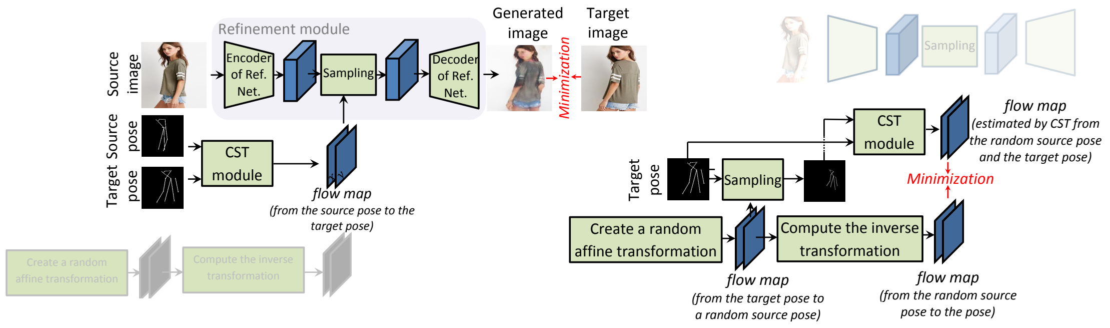
Refinement module
Source image
Encoder of Ref. Net.
Sampling
Decoder of Ref. Net.
Generated image
Target image
Minimization
Source pose
Target pose
CST module
flow map
(from the source pose to the target pose)
Create a random affine transformation
Compute the inverse transformation
Sampling
CST module
flow map
(estimated by CST from the random source pose and the target pose)
Target pose
Sampling
Minimization
Create a random affine transformation
Compute the inverse transformation
flow map
(from the target pose to a random source pose)
flow map
(from the random source pose to the pose)

Figure 1: Overall pipeline of our method, trained by alternating between two minimization tasks, depicted on the left and right sides of the figure.

3.1 Warping Module

3.1.1 Flow Estimation

Given a source pose $p_s \in R^{h\times w}$ and a target pose $p_t \in R^{h\times w}$, our warping module, $\mathcal{W}$, estimates a flow map, $f_{s,t} \in R^{2\times h\times w} = \mathcal{W}(p_s, p_t)$, which is later used to spatially transfer[3] the source image $I_s \in R^{3\times h\times w}$ to its target pose. Our estimation consists of 7 different steps. It finally outputs a flow map that is able to generate the unseen parts of the source sample I_s while displacing its visible parts to their new locations in the target pose:

1. Projecting p_s and p_t to a feature space using a single convolutional network: $p_s \to z_s$, $p_t \to z_t$[4]
2. Creating a correlation tensor from z_s and z_t using the pairwise similarity of their pixels: $(z_s, z_t) \to C$
3. Flagging a single pixel of z_s as the corresponding point for each pixel of z_t. To do so, we multiply Q by an attention tensor γ: $Q_\gamma \to \gamma C$
4. Estimating the flow map by applying a set of convolutional layers on Q_γ: $Q_\gamma \to f_{s,t}$
5. Applying $f_{s,t}$ to warp the source image. Then, we use a refinement network to estimate the target sample from this warped image. The estimation is encouraged by minimizing the distance between the output of the refinement network and the target sample.
6. Enforcing steps 1 to 4 to estimate an affine function if p_t is a linear transformation of p_s.
7. Alternating between the last two steps. In this way, the network leans to keep the vicinity of pixels in those parts whose movements from p_s to p_t are linear (i.e., can be approximated by an affine transformation) and use a non-constrained estimation for the remaining parts.

Creating a guided correlation map: (this corresponds to steps 1 to 3). The flow is directly estimated from the correlation of the source pose p_s and the target pose p_t. By correlation, we aim to create a one-to-one correspondence between each target pixel and its corresponding point in the source sample. This comes with two advantages: (1) each point in the target pose is compared with all the pixels of the source pose. Thus, the network can easily handle any large displacements of samples; (2) the network assigns each pixel to its relevant area in the source sample, even for those pixels that correspond to an invisible part of the source image. In this case, the sampling point is selected from an area with the closest texture.

To do so, both the source and the target poses are first projected to a feature space: $p_s \xrightarrow{\mathcal{M}} z_s \in \mathbb{R}^{m\times h_1\times w_1}$, $p_t \xrightarrow{\mathcal{M}} z_t \in \mathbb{R}^{m\times h_1\times w_1}$, where $\mathcal{M}$ is a fully convolutional network, h_1 and w_1 are the spatial dimension of the feature maps, and m is the dimensionality of the features. Thus, we

[3]Here, the spatial transformation is a sampling operation which samples a source image in the pixel locations of the flow map $f_{s,t}$

[4]z_s and z_t are three dimensional feature maps

have two dense feature maps which are later used for creating a *dense* correspondence between the source and the target poses of the sample. The correlation is computed based on the pixel-wise scalar product of the feature maps which is represented as follows:

$$C(i, j, , v, u) = \sum_{\rho=1}^{m} z_s(i, j)[\rho] z_t(v, u)[\rho] \tag{1}$$

where $z_s(i, j) \in \mathbb{R}^m$ and $z_t(v, u) \in \mathbb{R}^m$. $z_s(i, j)[\rho]$ indicates the ρ-th element of the feature vector in the spatial location (i, j) of the feature map z_s. Then, we assign each pixel in the target pose to only one pixel in the source sample. To do so, C is filtered so as to keep the maximum similarity for each target pixel, (v, u), and to mitigate the values of the other locations. The mitigation is performed using an attention mechanism that gives more importance to the location of the maximum similarities, while suppressing the values in other locations: $Q_\gamma = \gamma C$. However, as dot product is a collective measure of the angle and the magnitude, its results are unbounded, causing it to be sensitive to the magnitude of the generated sample. To avoid this, we normalize the features in each spatial location, (u, v), of Q_γ:

$$Q_\gamma(i, j, , v, u) = \frac{Q_\gamma(i, j, v, u)}{\sqrt{\sum_{i,j=1}^{h_1, w_1} \left(Q_\gamma(i, j, v, u)\right)^2}} \tag{2}$$

Estimating a flow map from a correlation tensor: (this corresponds to step 4) The flow is then estimated by applying a set of convolutional layers on the refined correlation Q_γ. To do so, we first rasterise Q_γ so that all the (i, j) locations are vectorized for each (v, u) point. The resulting tensor is then passed to the convolutional layers: $f_{s,t} = \mathcal{C}_l(Q_\gamma)$, where $\mathcal{C}_l$ stands for the convolutional filtering. Thus, the flow at each target position is estimated from the correlation of this point with all the positions of the source pose. This ensures any long-range displacements are captured during the process of flow estimation.

Finding the visibility of pixels in an unsupervised way: (this corresponds to steps 5 to 7) Up to this point, we have a non-parametric flow estimator, where the sampling locations are individually estimated for each pixel of the target sample. However, such estimation has no constraint to keep the vicinity of the pixels[5], which is important when reconstructing the textures that are visible in both the source and the target poses of the sample. In contrast, binding the estimation to keep the vicinity for the entire 2D space restricts the ability of the flow map to introduce any novel region that is not present in the source sample, especially for the regions with a novel shape, as the novel shapes can not be reconstructed by duplicating the visible parts of the sample.

To avoid this problem, we enforce the flow estimator to keep the vicinity of the pixels just in the visible areas. We approximate these regions with the areas that are displaced using a linear (affine) transformation. Then, for the remaining areas, the flow is estimated without applying any constraint on the pixel-wise estimator. This way, the vicinity of pixels, preserved by the affine transformation, is just applied to the areas that are visible in both the source and the target poses of the sample.

We implement this strategy in an unsupervised manner. To do so, we make steps 1 to 4 to alternate between two different tasks: (1) learning an affine transformation if the target pose is a linear transformation of the source pose, (2) estimating a warping map that best reconstructs the target image from the warped version of the source image.

For the first task, we first generate a random affine transformation using a few random parameters:

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} (-1)^{a_1} + a_2 & a_3 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}) \tag{3}$$

where $a_1, ..., a_5$ and b_1, b_2 are all random variables. If we sample the pixels of the target pose p_t using the generated affine transformation, it provides us with a novel pose p_{st} whose displacement is linear with respect to the target pose. This pose is considered as the source pose of task (1). In practice, we restrict the random values $a_1, ..., b_2$ so that the resulting transformation does not make a big difference in the verticality of the human skeletons.

[5]Here, we consider the vicinity in a large neighborhood which is more related to preserving the overall integrity of patches in the source sample

Then, we generate the inverse flow of this transformation using the following equation:

$$\begin{bmatrix} v \\ u \end{bmatrix} = \Big(\begin{bmatrix} (-1)^{a_1} + a_2 & a_3 \\ a_4 & a_5 \end{bmatrix} \Big)^{-1} \Big(\begin{bmatrix} i \\ j \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \Big) \tag{4}$$

This transformation is the flow map that transfers the source sample to the target pose. We require the target pose to be real. This avoids learning irrelevant mappings to a huge number of unrealistic poses that never happen in the real world. This is the reason for calculating the inverse flow and applying it on a randomly generated source pose. We refer to the flow map made by this affine function as $s_f = \{(v, u) | v \in \{1, ..., h_1\}, u \in \{1, .., w_1\}\}$.

Then, we minimize the distance between $f_{s,t}$ and s_f when the warping module receives p_{st} and p_t as its input samples. In this way, we enforce the pixel-wise flow map $f_{s,t}$ to be a linear function and therefore to keep the integrity of the neighboring pixels.

Next, we need to avoid this integrity to be applied to the areas whose displacements are not linear (the pixels that are invisible in the source image). To do so, we simply feed the module with the original p_s and p_t, estimate the pixel-wise flow map $f_{s,t}$, and then utilize it to warp the source sample. This warped image is then used to reconstruct the target sample. Since we do not use any *guiding map* (in concatenation with our warped image), this reconstruction requires the network to include all the invisible areas in the warped image. This is feasible due to the pixel-wise estimation of the flow map that does not apply the same function to the entire space. Therefore, a point can be sampled from any part of the source image regardless of the sampling location for its neighboring pixels. In this way, the pixels can take any arbitrary shapes, which is necessary for a complete match between the warped image and the target sample.

Finally, we only need to enforce the flow estimation to differentiate between the visible and the invisible parts of the sample and to restrict the affinity of the transformation to the visible areas. This is performed by alternating between task (1) and task (2). For task (1), we already enforced $f_{s,t}$ to be an affine function when the entire p_t is a linear transformation of the entire p_s. However, by nature $f_{s,t}$ is a pixel-wise transformation. When we alternate between two tasks, one of which is a pixel-wise transformation and the other one is the same function for the whole pixels, the network learns to apply the holistic one in a local manner. Therefore, it learns to apply the affine transformation on the local neighborhoods whose displacements[6] are linear (*visible* pixels). It means, the network learns to keep the vicinity of pixels in the local neighborhoods that are *visible* even if they are surrounded by the *invisible* areas.

3.2 REFINEMENT NETWORK

Even if the warping module can ideally generate the invisible parts of a sample, it has no prior information regarding the background of the images. In addition, warping has significant difficulties with generating a realistic face or hands. Therefore, the refinement module $\mathcal{R}$ is used to inpaint these regions and generate a photorealistic sample from the warped image. To do so, we benefit from a few convolutional layers without any skip connection. The module receives the source image along with the warping map generated by the flow estimation module. It first relies on a convolutional encoder to project the input image into a low-dimensional feature space, $\theta_s = E_{ref}(I_s)$, then utilizes the flow map to warp this resulting feature map, $\eta_s = \mathcal{T}(\theta_s, f_{s,t})$, where $\mathcal{T}$ is the sampling operation. Next, we benefit from a convolutional decoder to reproject the warped feature map η_s to the original dimension, $y_o = D_{ref}(\eta_s)$. y_o is the image that estimates the target sample. Finally, $f_{s,t}$ is used for direct sampling from the pixels of the source image which directly transfers the image to its target pose.

The encoder-decoder configuration allows adding more learnable parameters to the refinement process, which increases the overall performance of the network in generating fine-grained textures. In practice, the module is responsible for generating realistic faces and hands, as well as correcting the lines that are distorted due to inaccurate flow estimation of the warping module.

[6]here, we mean the transformation between the source and the target pose of the sample

3.3 Learning model

For training, we use three loss functions, a perceptual loss $\mathcal{L}_{per}$, an affine preserving loss $\mathcal{L}_{aff}$, and a generative loss $\mathcal{L}_g$. In Albahar et al. (2021), the authors propose to use an identity loss to preserve the identity of facial parts. However, our experiments showed that this could be misleading when the facial parts are occluded in either the source or the target images. In addition, Zhou et al. (2022) proposes to use a contextual loss to measure the similarity of non-aligned regions between the generated sample and the ground truth target image. However, this violates our goal to best fit the source image to the target pose without any non-aligned regions. $\mathcal{L}_{per}$ is utilized to ensure a pixel-level similarity between the generated image y_o and the target sample I_t, which is calculated as follows:

$$\mathcal{L}_{per}(y_o, I_t) = \sum \|\phi_i(y_o) - \phi_i(I_t)\|_1 \tag{5}$$

where $\phi_i(\S)$ is the i-th feature map of a pre-trained network (here we benefit from VGG-19 Simonyan & Zisserman (2014) pre-trained on Imagenet Deng et al. (2009)) when it is applied on $\S$. Inspired by Siarohin et al. (2019b), we use four different resolutions of y_o and I_t as input of our pre-trained model ϕ which ensures the similarity of the samples at different scales.

We also benefit from a generative loss, which encourages the photo-realism of the generated samples. To do so, both the warping and the refinement module are considered as the generator of our network $\mathcal{G} = \{\mathcal{W}, \mathcal{R}\}$ which competes against a discriminator $\mathcal{D}$. Our discriminator is conditioned on the target pose which means that both the generated and the target samples are first concatenated to the target pose and then passed to the discriminator. Our generative loss is defined as follows:

$$\mathcal{L}_g(y_o, I_t) = \mathbb{E}\big[\big(1 - \mathcal{D}(y_o, p_d)\big)\big] + \mathbb{E}\big[\mathcal{D}(I_t, p_d)\big] \tag{6}$$

where $y_o = \mathcal{G}(I_s, p_s, p_d)$.

Additionally, we use an affine preserving loss that enforces the network to preserve the linearity of the warping function in the regions where the target pose p_d is a linear transformation of the source pose.

$$\mathcal{L}_{aff}(p_t) = \|\mathcal{W}(p_{st}, p_t) - s_f\|_1 \tag{7}$$

where $\mathcal{W}(p_{st}, p_t)$ is the pixel-wise flow map estimated by the warping module. The overall loss function is defined as a weighted sum of $\mathcal{L}_{per}$, $\mathcal{L}_g$, and $\mathcal{L}_{aff}$, where λ_1 and λ_2 are empirically determined to ensure the best quality of the generated samples.

$$\mathcal{L}_t = \mathcal{L}_{per} + \lambda_1 \mathcal{L}_g + \lambda_2 \mathcal{L}_{aff} \tag{8}$$

3.4 Experiments

This section evaluates the performance of our method on two real-world datasets: Deepfashion Liu et al. (2016) and Market1505 Zheng et al. (2015). Deepfashion is a fashion-style dataset that includes 52,712 images, mostly captured indoors against a white background. Images are of the size 256×256, all provided in JPG format. We use the same split of data provided by Zhu et al. (2019), which includes 101,966 pairs of training samples and 8,570 test pairs. Each pair includes two images of the same person captured in different poses. The pose of each image is already extracted using OpenPose Cao et al. (2017).

3.4.1 Evaluation metrics

The evaluation is based on 3 different metrics: Structural Similarity Index Measure (SSIM) Wang et al. (2004), Learned Perceptual Image Patch Similarity (LPIPS) Zhang et al. (2018), and l_1-norm.

3.4.2 Quantitative analysis

We compare our method with GFLA Ren et al. (2020), FOMM Siarohin et al. (2019b), TPSMM Zhao & Zhang (2022), and MRAA Siarohin et al. (2021). Similar to our method, they all provide an estimation of the warping flow in an unsupervised manner. FOMM, TPSMM, and MRAA propose to estimate the keypoints of RGB images using an internal module. However, for a fair comparison, we modify this strategy so that the flow map is directly estimated from two sets of given keypoints. We evaluate the performance of each method for estimating an accurate flow map which is the main

Table 1: Comparison with the state-of-the-art on estimating the most accurate flow maps. The samples are generated by direct sampling from the source images and compared with the target samples.

	Sz	Ours	FOMM	MRAA	TPSMM	GFLA32	GFLA64
l_1-norm	32	**13.11**	13.26	13.79	13.66	14.18	14.01
l_1-norm	256	**15.52**	15.97	15.58	16.12	28.13	20.25
SSIM	256	0.608	0.604	0.607	**0.617**	0.554	0.615
LPIPS(Alex)	256	**0.39**	0.47	0.42	0.45	0.52	0.48
LPIPS(VGG)	256	**0.35**	0.42	0.38	0.41	0.49	0.44

Table 2: Ablation study on the depth of the $\mathcal{C}_l$. It projects the 4096-channel Q_γ to a 2-channel flow map $f_{s,t}$

Method	l_1-norm	LPIPS(Alex)	LPIPS(VGG)
1-layer	33.31	0.61	0.58
2-layer	27.92	0.59	0.51
3-layer	21.17	0.47	0.44
4-layer	16.14	0.42	0.39
5-layer	15.96	0.40	0.35
6-layer	15.52	0.39	0.35

Table 3: Ablation study on the refinement module. We conduct two sets of experiments with and without the refinement module.

Method	l_1-norm	SSIM	LPIPS(Alex)	LPIPS(VGG)
w/o	42.13	0.504	0.84	0.92
Full	15.52	0.608	0.39	0.35

Table 4: Ablation study on $\mathcal{L}_{aff}$. We conduct experiments on the samples of size 256×256

Method	l_1-norm	SSIM	LPIPS(Alex)	LPIPS(VGG)
w/o $\mathcal{L}_{aff}$	42.13	0.504	0.84	0.92
Full	15.52	0.608	0.39	0.35

contribution of our method and also the main difference between these competing algorithms. To do so, we first estimate a flow map using each of these networks (applied on a pair of the source and target poses). The map is then rescaled to 256×256 pixel and then utilized to warp the source image. Rescaling is performed by using the bilinear interpolation. Finally, the similarity of the warped image and the target sample is reported in terms of three different measures SSIM, LPIPS, and l_1-norm..

The results are listed in Table 1. The evaluation is provided in two different scales 32×32 and 256×256. For 32×32, we first downsample each image to 32×32 and then upsample it to the original dimension. All the competing algorithms use the same split of the dataset. This allows for a fair comparison between the results. For training the competing methods, we follow the same learning rate and the epoch number suggested in the original papers or the official implementations.

As can be seen, there are two versions of GFLA in two different scales, 32×32 and 64×64 which refer to the two flow maps estimated by this method. These maps are separately estimated in the first stage of this method and individually contribute in two distinct layers of its second-stage network. However, for FOMM, TPSMM, and MRAA, they provide a set of individual flow maps, each estimated using a parametric model. But these maps are later accumulated into a single flow map where each pixel of the final map is selected from one of these parametric estimations[7]. From the table, our method outperforms the state-of-the-art on three out of four evaluation metrics and on two scales which means the superiority of our method both in reconstructing the global shapes of the target samples and in generating textures in their correct position.

3.4.3 VISUALIZATION

We further visualize a few samples of warped images generated by our method and each of these competing algorithms. The results are shown in Fig. 2. As can be seen, none of the competing methods are able to introduce the invisible parts into the warped images. GFLA suffers from some unwanted distortions even for displacing the visible patches to their new locations. This happens because its flow estimator makes no provision for covering the invisible parts of the sample. In contrast, FOMM, TSPM, and MRAA can ideally displace the visible patches but due to their parametric estimation, they fail to estimate a correct flow map between two complex poses. As can be seen, their attempt for introducing the invisible parts is limited to stretching or duplicating a part of images. Unlike these strategies, our method can estimate a highly complex flow function that not only displaces the visible parts to their new location but also introduces any novel parts that are indicated by the target pose, all by warping from the visible patches of the source sample.

[7]A definition of the parametric flow estimation is provided in Section 1

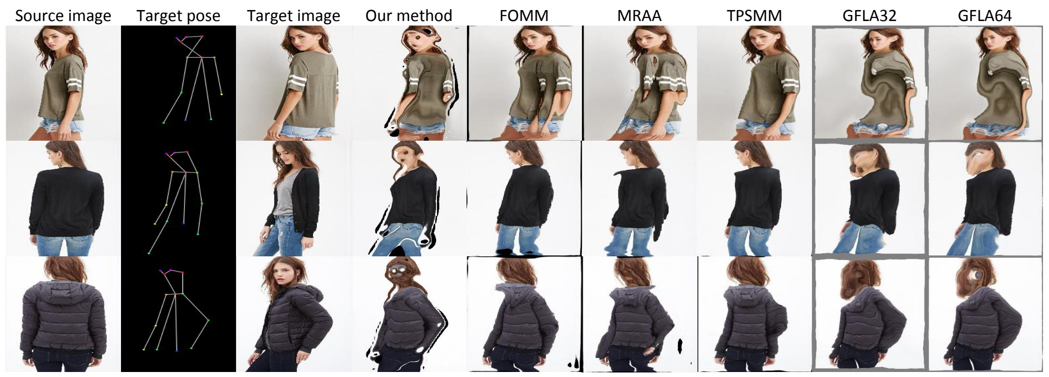

Figure 2: Visual comparison of our method and the state-of-the-art techniques (**these images are generated by direct sampling of pixels from a source image and not using a convolutional network or a generative model**).

3.4.4 Ablation study

We evaluate the efficiency of each block in our proposed algorithm. To do so, we consider two variants of our method: (1) our model without $\mathcal{L}_{aff}$, and (2) our model without a refinement network. All the experiments are conducted on the same split of the dataset as in Section 3.1.

The results are reported in Tables 3 and 4. Without $\mathcal{L}_{aff}$, the model simply estimates the flow map using a pixel-wise transformation without any encouragement for preserving the vicinity of pixels. In this way, the model still generates the overall pose of the samples, however it fails to generate a faithful appearance to the source images. The results in the Tables validate this assumption, where the performance of *model w/o* $\mathcal{L}_{aff}$ is significantly lower than the full model This proves the effectiveness of $\mathcal{L}_{aff}$ in preserving the fidelity of the textures, especially for the *visible* parts whose displacements are usually linear, even though $\mathcal{L}_{aff}$ has no prior knowledge about the visibility of the pixels.

Without a refinement module, the model directly computes the flow map by minimizing the distance between the warped image and the target sample. In this way, all the background estimations are performed using the flow estimation module. This causes an overfitting issue when we fit too closely to the details of the background regions. This can be verified by the result in Table 4, where a refinement module significantly boosts the performance of our method on all the metrics.

We also conduct another experiment to evaluate the performance of the model when we use a projection network C_l with different number of layers. The experiment is performed using six different layers. The depth of the layers is respectively 128, 128, 96, 64, 32, and 2. As can be seen in Table 2, the best performance is achieved when we use $\mathcal{C}_l$ with five to six hidden layers which are considered in the full model of our method.

4 Conclusion

We introduced an end-to-end model for generating a novel human pose using a single warping module. This module not only displaces the pixels but also generates the pixels that are invisible in the source image. This is required if the warped version of the source image is the only feature map that is used for generating a novel pose of the sample. To do so, we proposed a non-parametric flow estimator which is a pixel-wise transformation but learns to estimate an affine transformation for the regions that are visible in both the source and the target samples. In this way, the visible parts are transferred using a linear function that allows the vicinity of their textures to be preserved when transferring from the source to the target pose. The visibility of pixels is captured in a completely unsupervised manner. Our method does not use any priors about the dense position of the samples, neither in the source nor in the target pose. We evaluated the performance of our flow estimation module compared to a set of state-of-the-art algorithms. We also conduct another experiment on a low-resolution image database to analyze the application of our method for boosting the performance of the re-identification methods.

REFERENCES

Badour Albahar, Jingwan Lu, Jimei Yang, Zhixin Shu, Eli Shechtman, and Jia-Bin Huang. Pose with style: Detail-preserving pose-guided image synthesis with conditional stylegan. *ACM Transactions on Graphics (TOG)*, 40(6):1–11, 2021.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7291–7299, 2017.

Enric Corona, Albert Pumarola, Guillem Alenya, Gerard Pons-Moll, and Francesc Moreno-Noguer. Smplicit: Topology-aware generative model for clothed people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11875–11885, 2021.

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*, pp. 248–255. Ieee, 2009.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Yining Li, Chen Huang, and Chen Change Loy. Dense intrinsic appearance flow for human pose transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3693–3702, 2019.

Meichen Liu, Xin Yan, Chenhui Wang, and Kejun Wang. Segmentation mask-guided person image generation. *Applied Intelligence*, 51(2):1161–1176, 2021.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1096–1104, 2016.

Kunming Luo, Chuan Wang, Shuaicheng Liu, Haoqiang Fan, Jue Wang, and Jian Sun. Upflow: Upsampling pyramid for unsupervised optical flow learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1045–1054, 2021.

Zhengyao Lv, Xiaoming Li, Xin Li, Fu Li, Tianwei Lin, Dongliang He, and Wangmeng Zuo. Learning semantic person image generation by region-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10806–10815, 2021.

Yurui Ren, Xiaoming Yu, Junming Chen, Thomas H Li, and Ge Li. Deep image spatial transformation for person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7690–7699, 2020.

Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. *Advances in neural information processing systems*, 31, 2018.

Sara Sabour, Andrea Tagliasacchi, Soroosh Yazdani, Geoffrey Hinton, and David J Fleet. Unsupervised part representation by flow capsules. In *International Conference on Machine Learning*, pp. 9213–9223. PMLR, 2021.

Karthik Abinav Sankararaman, Soham De, Zheng Xu, W Ronny Huang, and Tom Goldstein. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In *International conference on machine learning*, pp. 8469–8479. PMLR, 2020.

Kripasindhu Sarkar, Vladislav Golyanik, Lingjie Liu, and Christian Theobalt. Style and pose control for image synthesis of humans from a single monocular view. *arXiv preprint arXiv:2102.11263*, 2021.

Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuiliere, and Nicu Sebe. Deformable gans for pose-based human image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3408–3416, 2018.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2377–2386, 2019a.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32: 7137–7147, 2019b.

Aliaksandr Siarohin, Oliver J Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13653–13662, 2021.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3887–3896, 2021.

Hao Tang and Nicu Sebe. Total generate: Cycle in cycle generative adversarial networks for generating human faces, hands, bodies, and natural scenes. *arXiv preprint arXiv:2106.10876*, 2021.

Hao Tang, Song Bai, Philip HS Torr, and Nicu Sebe. Bipartite graph reasoning gans for person image generation. *arXiv preprint arXiv:2008.04381*, 2020.

Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2387–2395, 2016.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Jichao Zhang, Aliaksandr Siarohin, Hao Tang, Jingjing Chen, Enver Sangineto, Wei Wang, and Nicu Sebe. Controllable person image synthesis with spatially-adaptive warped normalization. *arXiv preprint arXiv:2105.14739*, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Jian Zhao and Hui Zhang. Thin-plate spline motion model for image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3657–3666, 2022.

Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pp. 1116–1124, 2015.

Xinyue Zhou, Mingyu Yin, Xinyuan Chen, Li Sun, Changxin Gao, and Qingli Li. Cross attention based style distribution for controllable person image synthesis. *arXiv preprint arXiv:2208.00712*, 2022.

Zhen Zhu, Tengteng Huang, Baoguang Shi, Miao Yu, Bofei Wang, and Xiang Bai. Progressive pose attention transfer for person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2347–2356, 2019.

Yang Zou, Xiaodong Yang, Zhiding Yu, BVK Kumar, and Jan Kautz. Joint disentangling and adaptation for cross-domain person re-identification. In *European Conference on Computer Vision*, pp. 87–104. Springer, 2020.

A Implementation details

We implement $\mathcal{M}$ as a fully convolutional network. It benefits from 11 convolutional layers, each headed with a Batch Normalization Ioffe & Szegedy (2015). The last layer outputs a feature map $z_.$ with the size of $35 \times 64 \times 64$ where 35 is the dimension of the features and 64×64 denotes the spatial size of the feature map. Each channel of this map is then normalized using an l_2 norm.

γ is implemented using the Soft Mutual filtering Rocco et al. (2018) which preserves $C(i, j, v, u)$ if $z_s(i, j)$ and $z_t(v, u)$ are the most similar pixels to each other, when comparing each of these pixels with all the pixels of the other feature map.

$$\gamma(i, j, v, u) = \frac{C(i, j, v, u)}{max_{\alpha\beta}C(\alpha, \beta, v, u)} \frac{C(i, j, v, u)}{max_{\alpha\beta}C(i, j, \alpha, \beta)} \tag{9}$$

where $max_{\alpha\beta}C(\alpha, \beta, v, u)$ denotes the maximum similarity between $z_t(v, w)$ and the entire pixels of $z_s(\alpha, \beta)$ when $\alpha \in \{1, ..., h_1\}$ and $\beta \in \{1, ..., w_1\}$.

For $\mathcal{C}_l$, we use a 6-layer convolutional network, in which the first layer receives the rasterized tensor of the size $4096 \times 64 \times 64$ and the last one outputs a flow map of size $2 \times 64 \times 64$. The depth of the layers is respectively 128, 128, 96, 64, 32, and 2.

The encoder of the refinement network includes two convolutional layers each followed by a down-sampling operation. The last layer outputs a feature map of size $256 \times 64 \times 64$. The decoder is implemented using 6 residual blocks along with 2 upsampling layers. We use a Bach Normalization after each upsampling layer. The architecture of the Residual blocks is depicted in Fig. 3.

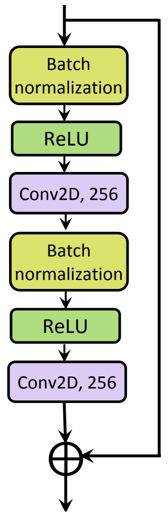

Figure 3: Residual Block of our refinement network.

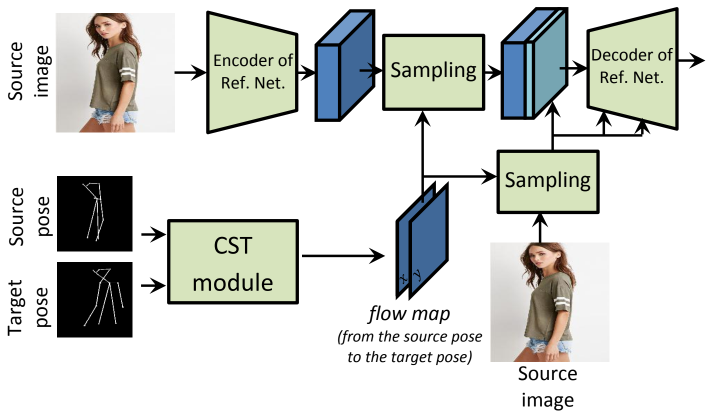

Figure 4: Our model when using skip connection in the refinement module.

B Details of the training process

We train our model for 300 epochs with a batch size of 7. Our regularization parameters λ_1 and λ_2 are respectively set to 0.1 and 0.001. To ensure a stability of training, we first train our model without a generative loss for approximately 100 epochs, using a learning rate of $1e - 6$. Then, we increase the learning rate of the refining decoder to $1e - 4$ while reducing the learning rate of the remaining parts to $1e - 8$.

Table 5: Effectiveness of a flow estimation technique on a person re-identification network.

	mAP	Rank1	Rank5	Rank10
DG-Net++	61.7	82.1	90.2	92.7
Our method+DG-Net++	63.8	83.8	93.0	95.4

Table 6: User study. The results are provided based on two different questions from 6 volunteers.

Method	Q1	Q2
Ours/FOMM	82.7/17.3%	90.6/9.4%
Ours/MRAA	78.9/29.1%	84.7/15.3%
Ours/TPSMM	80.8/19.2%	87.7/12.3%
Ours/GFLA32	75.4/24.6%	77.2/22.8%
Ours/GFLA64	70.8/29.2%	75.6/24.4%

C Application for person re-identification

We also conduct another experiment to evaluate the performance of our method for a recognition task. To do so, we train our model on a paired-image dataset to learn a flow map that generates a novel view of a person. Then, we use this model for augmenting the training set of a re-identification database. This way, we increase the number of poses that each person contribute to the training set. Note that, we do not use the output of the Refinement network, but instead we generate the novel views just by sampling the pixels using the estimated flow map. This allows us to evaluate the performance of our flow estimation model in a task that does not necessarily require a detailed image for the recognition process.

Our experiment is conducted on the Market1501 dataset. It contains 12,936 images of 751 identities as the training samples and 19,732 images of 750 identities as queries. The images are all collected outdoors using 6 different cameras.

For the flow field estimation, we use the paired dataset provided by Zhu et al. (2019). Note that, our model is an algorithm that learns to estimate a flow map from two skeletal poses and therefore is not biased towards the appearance of the training samples. After training the flow estimator, we randomly select 25 poses from the training set and transfer each of the training samples to all these novel poses. All the background regions are replaced with gray. This provides us with 323,400 training samples which cover most of the natural poses that a human takes while walking on the street. Then, we use this new dataset to fine-tune a pre-train re-identification network and then test it on the query samples. Here, we use DG-Net++ Zou et al. (2020) as our baseline model. The results are listed in Table 5. As can be seen, the result clearly demonstrates the effectiveness of the augmentation on the performance of this method. We also provide a visualization of our flow estimation technique on this dataset (Figure 5).

D User study

We ask 6 different people to evaluate the performance of our method compared with the state-of-the-art. The evaluation is based on two different questions: (1) which algorithm better estimates the pose of the samples? (2) which algorithm better generates the invisible parts of the samples. The experiment is conducted on 80 images along with their new poses. The results are listed in Table 6, further verifying the superiority of our method to the state-of-the-art.

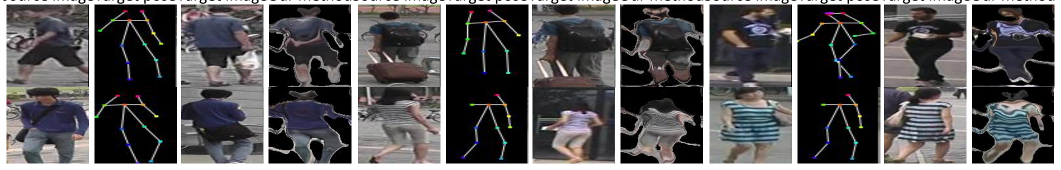

Figure 5: Generating a sample from the Market database by direct sampling from the source image.

E Skip connection

We also conduct an experiment to evaluate the efficiency of skip connections in preserving the fidelity of textures in our method. To do so, we use the structure shown in Figure 4. As can be seen, the flow map (estimated by the warping module) is used to simultaneously sample from the source image and its feature map. Then, the warped version of the source image is hierarchically concatenated to the feature maps of different layers in the decoder of the refinement module. The results of our experiments are shown in Figure 6. Our primary assumption was the effectiveness of this strategy in boosting the fine-grained textures in the generated sample. However, as can be seen, this way the network fails to preserve the integrity of the pixels which is mostly related to the Gradient Confusion (GC) effect Sankararaman et al. (2020), caused by several sources of change for the same concept in a network.

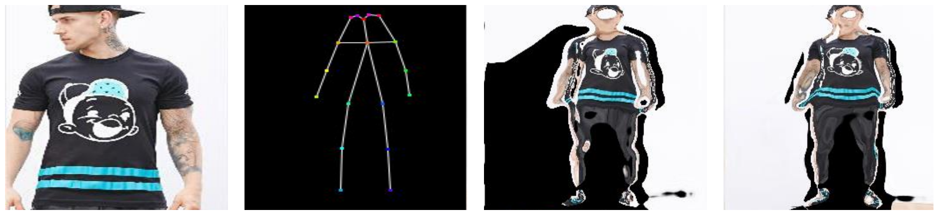
Figure 6: the effectiveness of skip connection, (a) source image, (b) target pose, (c) Generated sample using the flow map estimated by Fig 4 (there is a duplication of the source patches in the generated sample), (d) generated using the flow map of the configuration in Fig. 1.

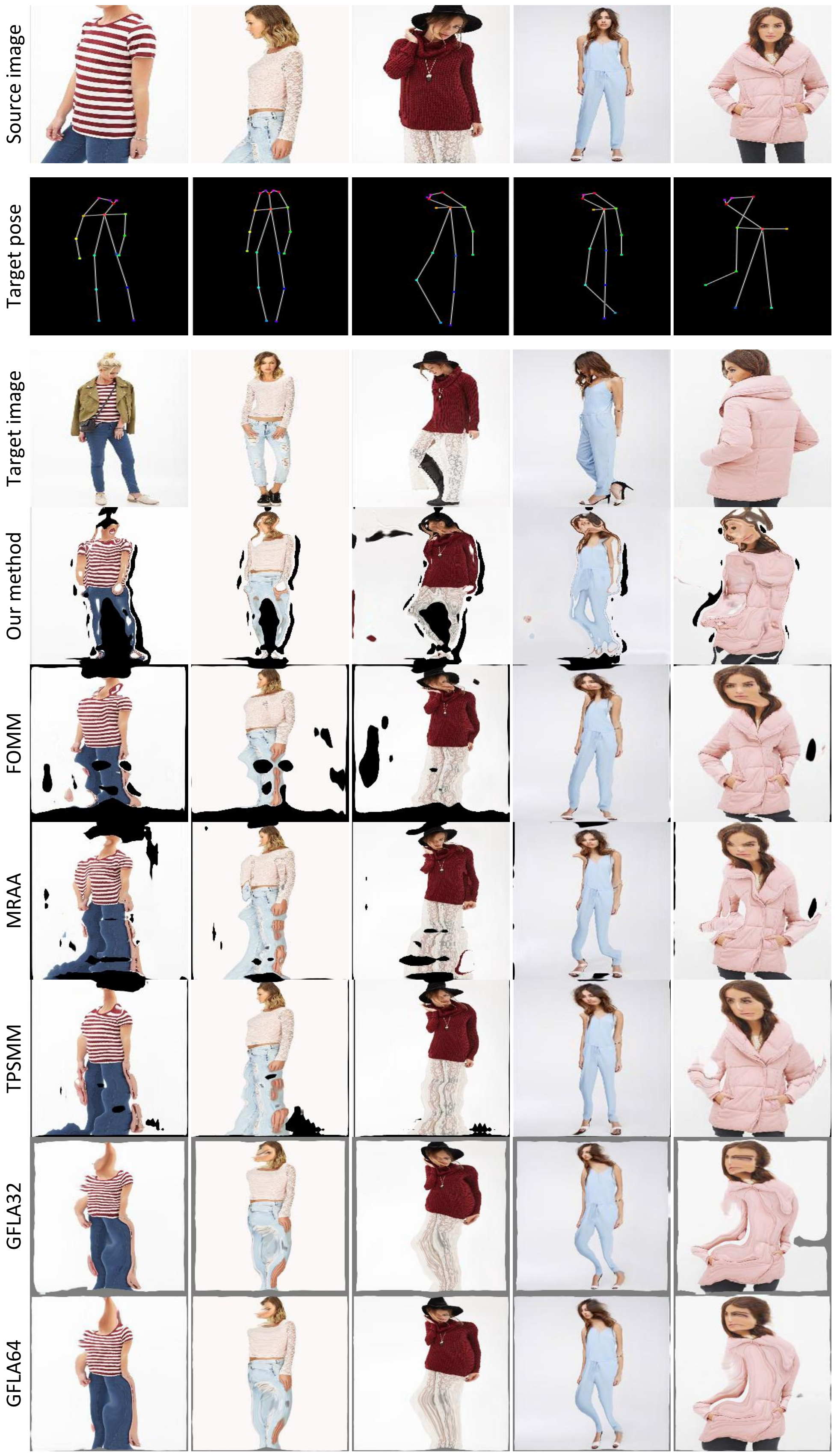

Figure 7: Additional comparison between our method and the state-of-the-art.

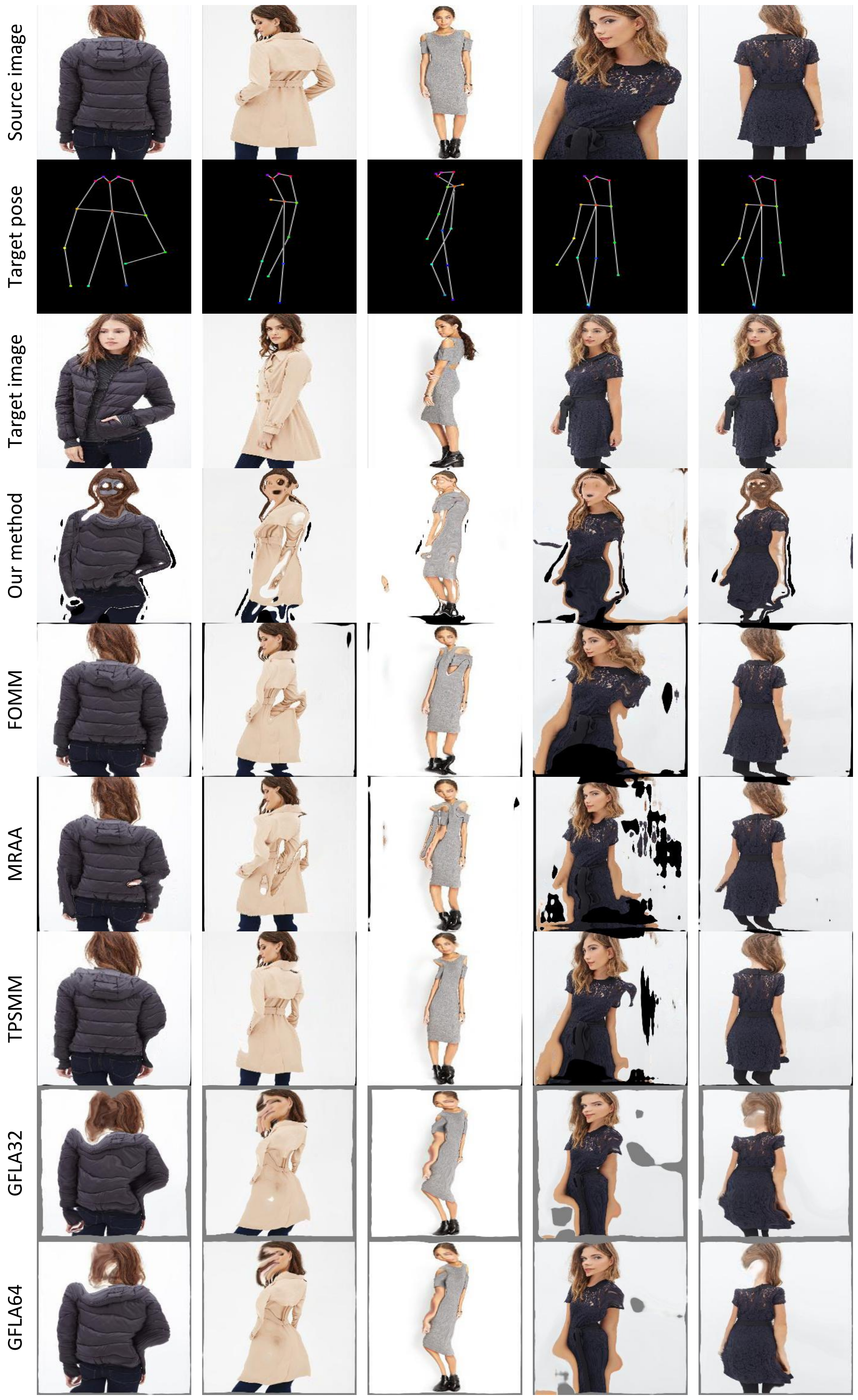

Figure 8: Additional comparison between our method and the state-of-the-art.

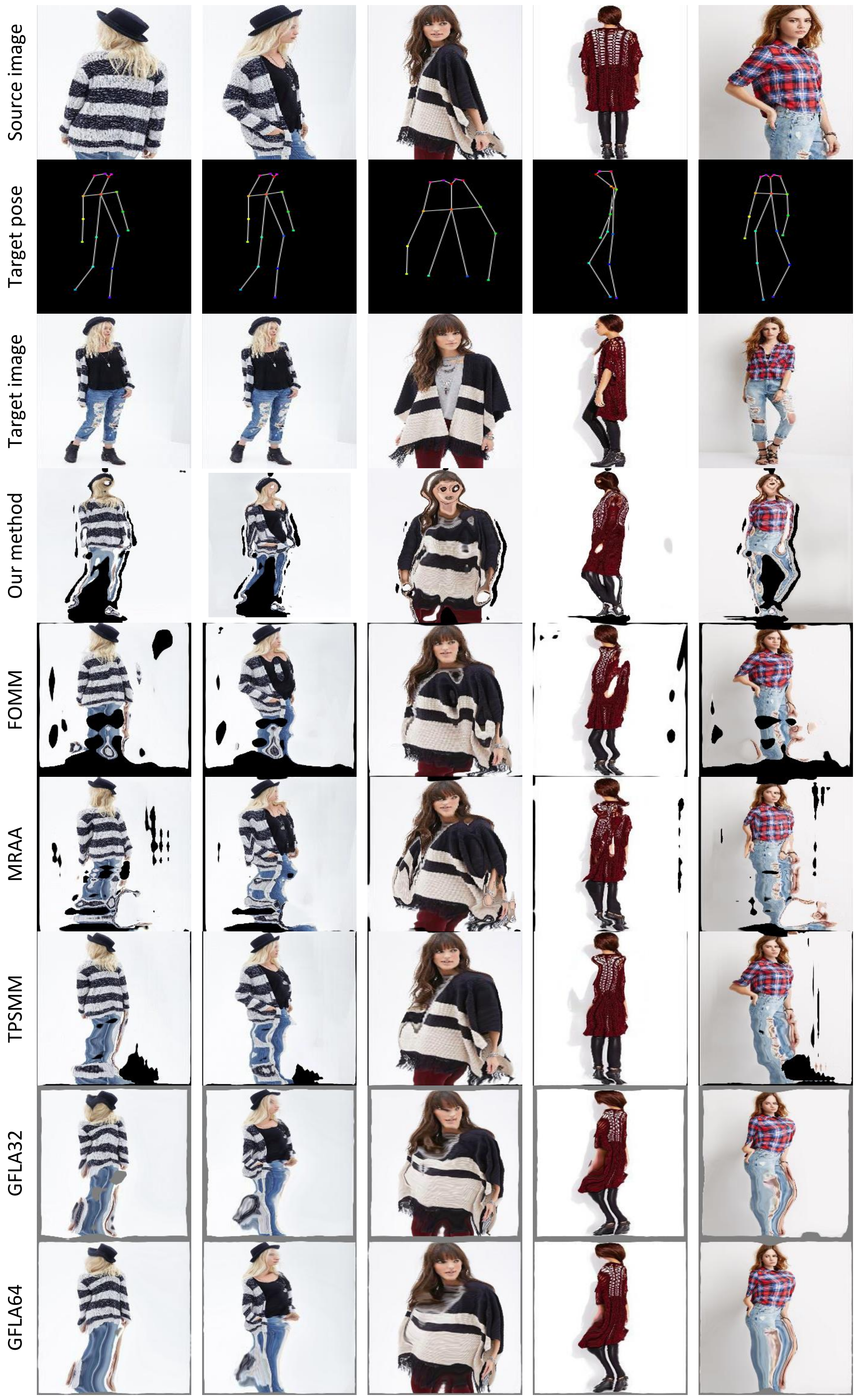

Figure 9: Additional comparison between our method and the state-of-the-art.

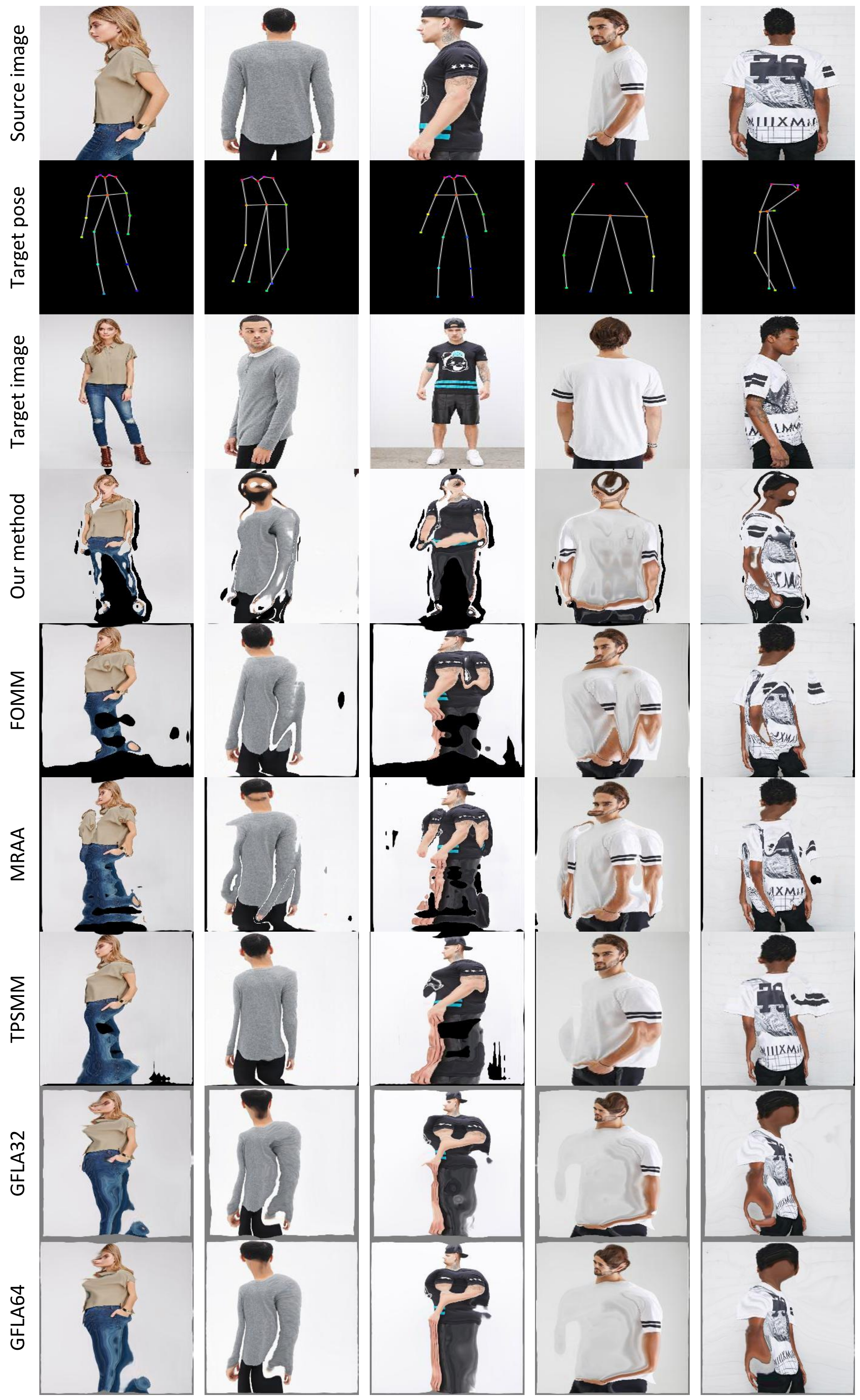

Figure 10: Additional comparison between our method and the state-of-the-art.