Controlled Simultaneous Pose and Appearance Manipulation

Shivam Grover
Bharati Vidyapeeth's College of Engineering
Paschim Vihar, Delhi 110063
shivumgrover@gmail.com

Koteswar Rao
Indraprastha Institute of Information Technology Delhi
Okhla Industrial Estate, Delhi 110020
secondauthor@i2.org

Abstract

Manipulating the geometric structure (pose) in images has been a well-explored area. However, methods for manipulating the appearances are not that well established, mainly because it is difficult to curate a labeled dataset in which for each individual, there are images in different appearances (and pose for our task). In this paper, we propose a novel approach for manipulating the appearance and the pose simultaneously in a controlled manner. We first separate the pose and appearance in the images, giving us control over each of them separately, thus giving the generator access to a much larger and previously unseen augmented dataset with combinations of multiple appearance and pose components. While training, we maintain a three-way cyclic consistency which keeps a check on the pose and appearance of the generated images and makes sure that the identity of the person is retained after the appearance transfer.

1. Introduction

We present a novel approach for simultaneously manipulating the pose and appearance in images of humans conditioned on a set of target pose and appearance. More specifically, given a source image *X*, a target pose *Py* and a target appearance *Ay*, we synthesis a target image of the original person present in *X*, such that his pose is *Py* and appearance is *Ay*. While doing this transfer, we also make sure to retain the identity of the person (such as the facial features, the size of the body, and the patterns and designs on the cloths). To the best of our knowledge this had been an untouched problems.

Existing works in the domain of pose transfer [1][8][9][11][17][14][15] have shown promising results.

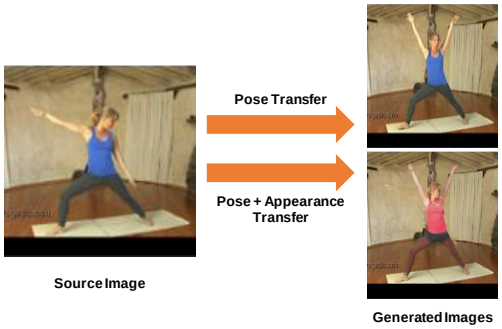

Figure 1. A comparison between pose transfer (top right) and simultaneous pose and appearance transfer (bottom right). Our model is capable of performing simultaneous pose and appearance transfer in a single go.

These methods generally use a supervised approach for training their generator, in which for each individual human (or object) in the dataset, they have multiple images with varying poses (for example different frames of a video in which the subject person is moving around), and by sampling random pairs of poses for each individual, they train their network to go from the source image to the target image conditioned on the target pose. Similar supervised methods exists for appearance transfer as well, but they require multiple images in different appearances for each individual. Such a dataset is much harder to curate than the former dataset for pose transfer. Approaches for appearance transfer that do not require paired data such as [16] use a cyclic-consistency approach. However, such approaches require geometric alignment between the input and the output, hence such models do not work well when for transferring appearance along with geometric deformations (such as pose transfer).

Through this work, we achieve pose and appearance transfer in a single shot (Fig 1). First, we curate a dataset of several videos in which each individual performs various activities. Within each video there is a rich amount of pose variation, however the appearance is constant. So we formulate a representation in which we separate the pose from the appearance. We use a key-points based representation for the pose which is obtained using a state-of-the-art pose-estimator [2]. For representing the appearance, we first obtain the segmentation masks for each body part and then extract their corresponding appearances. This gives us control over pose and appearance components individually and allows us to perform two important things. First, it allows us to remove the original appearance *Ax* from the source image, which allows the model to learn to adopt the provided target appearance *Ay* instead of the original source appearance. Second, by randomly pairing pose and appearance components over different videos in the dataset, the generator model gets access to a large amount of unseen dataset with varying poses and appearances.

To build the network for pose transfer, we use an approach similar to [1] in which one module learns to segment the foreground (different body parts) from the background, and then using affine transforms, the segmented parts are warped transformed and warped to resemble the target pose. This coarse image is sent through the generator module which adds outputs a realistic image with the transferred pose. The model takes as input the source image *X*, the source pose *Px*, the target pose *Py*, the target appearance *Ay*, the affine transformations to go from *Px* to *Py*, and a coarse location of each body part in *X*. As an additional step, we convert the segmented foreground layers into grayscale layers. This essentially subtracts the original appearance *Ax* from the source image, but keeps the identity (such as facial features and patterns on the cloths). We do this to make sure that the model learns to use the target appearance *Ay* provided as input. Finally to cater to the lack of a labelled dataset, we establish a virtual supervision by maintaining a two-way cyclic-consistency which keeps a check on the pose and appearance of the generated images and also helps retain the identity of the person in the source image after the transfer.

Manipulation of pose and appearance in images of humans has several applications, such as generating a large amount of realistic augmented data from a limited dataset, person re-identification, performing realistic animations, and virtual try-on of cloths. While we only show the efficacy of our work on images of humans, it should work for other categories of objects such as faces, animals, furniture, etc. as long as there is a line of separation in the shape and appearance.

2. Related Works

2.1. Image synthesis and manipulation

Image synthesis is one of the major use cases of deep generative networks. More recent state-of-the-arts in this domain exploit Generative Adversarial Networks (GANs) [4] or Variational Auto Encoders (VAEs) [7]. In VAEs ¡insert VAE definition¿. On the other hand, in GANs two networks, namely the discriminator which tries to predict whether an image looks real or not and the generator which tries to generate realistic fakes and fool the discriminator, are trained at the same time and they are made to compete with each other until a state of nash equilibrium is achieved.

Isola *et al*. [6] proposed a general framework for the task of paired (supervised) image-to-image translation based on GANs. Zhu *et al*. [16] later proposed a similar framework but for unpaired (unsupervised) image-to-image translation in which they train two generators at the same time and instead of supervising each generated image for each generator, they instead try to inverse the translation using the other generator and use the original image for supervising the inverted generated image. This way they train the image-to-image translation networks in an unsupervised way by maintaining a cyclic consistency. We make use of a two-way cyclic-consistency in our work as well and it allows us to train our network without any labelled data of the augmented appearance components the generator is trained on. In our case however, we only require a single generator which makes the training much faster than [16].

2.2. Pose Transfer

Supervised learning for pose transfer alone is comparatively easier since curating a dedicated dataset simply involves obtaining videos of people performing daily life activities. Balakrishnan *et al*. [1] use such a method for transferring actions from different videos of sports. Reed *et al*. [12] propose a Generative Adversarial What-Where Network which generates unseen images of humans based on input keypoint locations and text descriptions. They extend their work further in [13] and use an extension of the Pixel Convolutional Neural Network.

2.3. Appearance Transfer

2.4. Pose and appearance transfer

3. Methodology

Our goal is to teach our network how to comprehend the target pose Py and target appearance Ay provided as inputs and generate an image of the person in the source image conditioned on Px and Ay. In the generated image, the identity of the source person should be retained while in harmony with the target appearance.

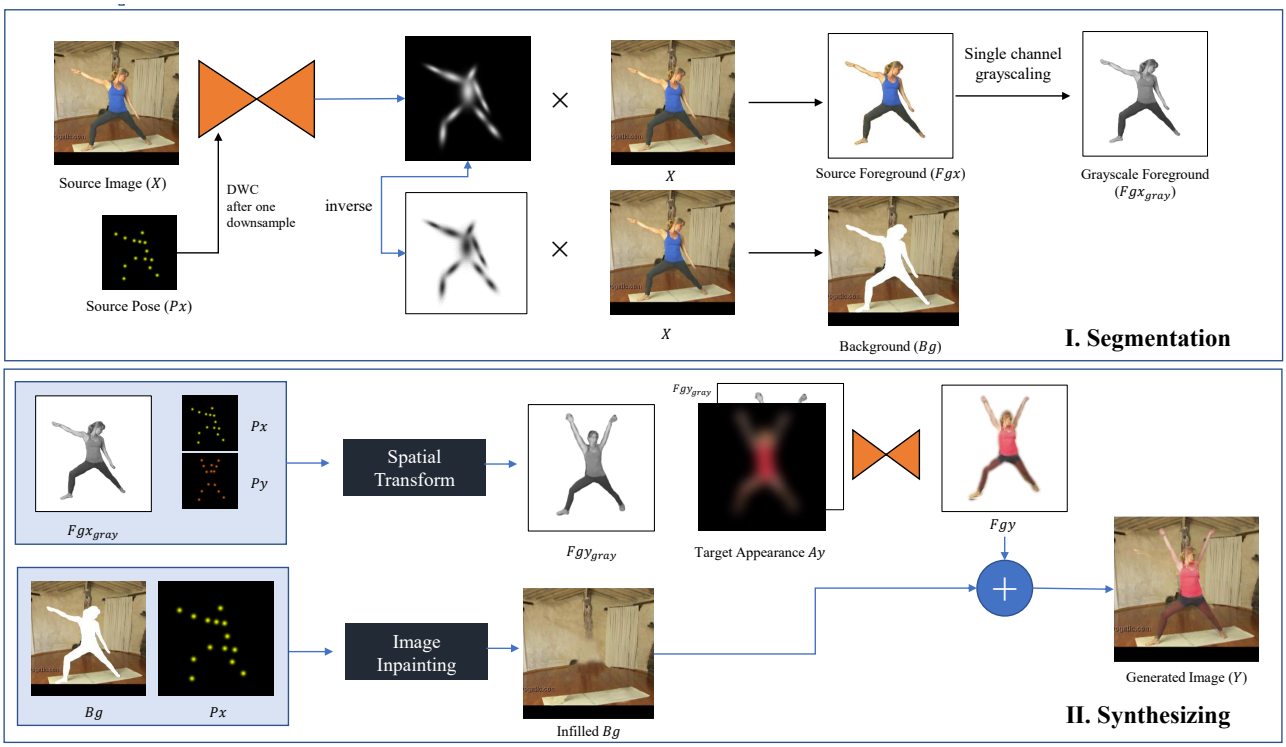

Figure 2. The inputs to the generator network include the source image X, source pose Px, target pose Py, a rough estimate of the position of each body part Mx, and the affine transformations T for transforming Mx into to Py, target appearance Ay. The output is a generated image containing the original person as in X but in the target pose and appearance.

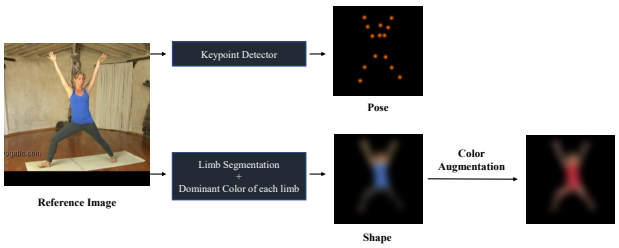

Figure 3. Separation of the pose and appearance components from the images. The right most image shows how this enables us to randomly sample different appearances in the same pose.

3.1. Augmentation within Dataset

A simple approach can be obtaining a dataset in which for each individual there are images in varying poses and appearances. Curating such a dataset is a mammoth task in itself and to the best of our knowledge, such a dataset large enough to train a deep neural network doesn't exist. Instead, we use a simple dataset (similar to that used by [1]) containing videos of multiple people performing various activities. Then we extract the pose component using a human pose estimator [2] and we use [5] to obtain the segmentation masks for each body part, assigning the most dominant color in each segment as its corresponding appearance component (Fig: 3). The pose component of image X can be represented as $Px \in \mathbb{R}^{14\times H\times W}$ where H and W is the height and width of the image respectively. The appearance component is essentially the dominant color codes for each segment of body parts and can be represent as $Ay = \{r, g, b\} \in [0, 255]$.

While the obtained pose and appearance components are not sufficient to reconstruct the image they were extracted from, they are adequate for conditioning our network into understanding the target pose and appearance into which the source image has to be transformed. Hence, in a dataset containing n videos, each containing m frames, we can simply mix and match all different appearances Aij with all different poses Pij, where $i \in [0, n]$ and $j \in [0, m]$ of different people in order to get a very large augmented dataset of total $(n\times m)^2$ combinations of varying poses and appearances. While we include the appearances from all frames of each video in order to allow the model to learn the variations in appearances due to different illumination conditions, one may choose to sample the appearance of each video only once instead of for each frame. Even then the total number of combinations will be $n^2 \times m$. Since each layer of the appearance component is simply a solid color, we can also

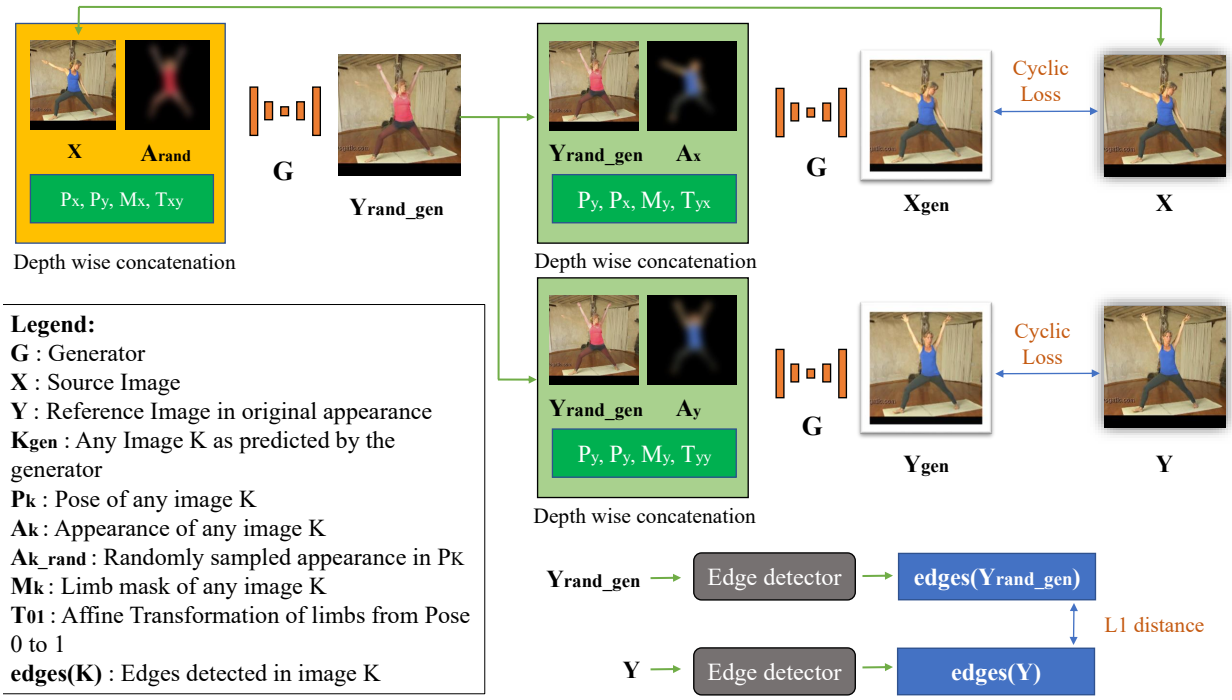

Figure 4. The inputs to the generator network include the source image *X*, source pose Px, target pose Py, a rough estimate of the position of each body part Mx, and the affine transformations T for transforming Mx into to Py, target appearance Ay. The output is a generated image containing the original person as in X but in the target pose and appearance.

randomly assign colors in order to get an infinitely large dataset. However, we prefer the former method of swapping appearances with other videos, since the latter resulted in some very unrealistic appearances and we leave that for future work.

3.2. Generator Model

The generator model is divided into four modules. The first module separates the foreground fg from the background bg in the image and also separates the segment for each body part in the foreground. The separated foreground currently consists of 3 color channels for each body part segment and has a shape of $3 \times 10 \times H \times W$. This module then removes the original appearance Px from the foreground by converting each foreground segment into a single channel grayscale layer before sending it to the next module. The final grayscale foreground can be represented as $fg \in \mathbb{R}^{10 \times H \times W}$. In the data processing step, we calculate the transformations for each body part required to go from Px to Py. The second module uses these transformation matrices and transforms each layer of fg and warps and aligns it to match the corresponding body part in the target pose Py. The third module takes the separated background and performs inpainting over the areas previously occluded by the foreground. Finally the the fourth module takes as input the grayscale fg, the colored bg, and the target appearance Ay. It learns to apply the appearance Ay to fg while generating the final image.

3.3. Training process

For each batch, from a single video in the training dataset we choose random pairs of frames for the source image X and the "psuedo" target image Y. The pose and appearance components of X are Px and Ax and of Y are Py and Ay. For the unseen appearance and pose combination, the random appearance A_{rand} is sampled from a frame of another video chosen randomly.

Each training iteration is a three step process. First, the generator is given X as the input image and Py and A_{rand} as the target pose and appearance respectively. The generator outputs an image Y_{rand} that should have the original person (as in X) in the specified target pose and appearance. This would be the general real-life use case of the generator (unseen target pose and appearance). However, due to the replacement of Ay with A_{rand}, there doesn't exist a ground truth to compare Y_{rand} with. The "pseudo" target image Y cannot be used since the appearance is completely different in it. This is where the need to establish cyclic-consistency arises. In the second step, the generator is given the previously generated image Y_{rand} as the input image and Px and Ay as the target pose and appearance respectively. In simpler words, in this step, we are asking the generator to

go back to the original image from the previously generated image. If the generator works well, then the output of the second step X_{gen} should be equal to X. In the third step, the generator is given Y_{rand} again, but this time we give it Py and Ay as the target pose and appearance. We are asking the generator to go from Y_{rand} to Y. Similar to the previous step, if the generator works well, then the output image Y_{gen} from the third step should be equal to Y.

Finally, we also have a discriminator whose job is to tell whether the generated image is real or fake. We send all three generated images (Y_{rand}, X_{gen}, and Y_{gen}) into the discriminator at each training iteration.

3.4. Face Orientation Correction GAN (FOC GAN)

We do not include any data about the facial keypoints in the input to the generator above. A single keypoint, near the forehead is provided as a reference for where the whole face would be. The generator is able to change the global orientation to some extent using that (such as tilting the neck sideways), but is not able to change the facial details in the local orientation (such as rotation of the head, regardless of the relative position) and other details such as expressions. Hence, the generated image has a very similar local facial orientation and facial structure as the source image. To cater to this, we built a separate generator (similar to [3] for correcting the facial orientation using the source and target facial keypoints as the input. This generator is also trained in an adversarial manner and the architecture is borrowed from Isola *et al*. [6]. The input to this network is the cropped region of the face in the generated image, the same cropped region of the face from the source image, the source facial keypoints, and the target facial keypoints. It should be noted that the cropping algorithm adapts to different body sizes in the images so that the face fits well inside the cropped region with minimum space wasted. For training this model, we build a dataset containing the cropped face regions and their corresponding keypoints that we extract using [**?**]. We further employ the main generator on several random pairs from the images that we used in building the dataset for face images and crop the face regions in the generated images as well. Then we load a random generated image of the face, and their corresponding source and target images and keypoints and train the model using the target image as the label and the rest as the input.

4. Training Losses

For calculating the loss between two images, we adopt the method used in [1]. In this we take the first 16 layers of the VGG19 model pretrained for image classification [] and calculate the $L1$ distance in a feature space Φ that is built by concatenating the channels of these layers. This enforces the generator to recognize and reconstruct certain patterns, edges, textures, etc. that works better and looks more realistic than simply calculating the $L1$ or $L2$ losses as demonstrated by [1]. The VGG loss between image I and I_0 is

$$\mathcal{L}_V(I, I_0) = \mathbb{E}\left[\|\phi(I) - \phi\left(I_o\right)\|_1\right] \tag{1}$$

Our loss function consists of four terms, and they are

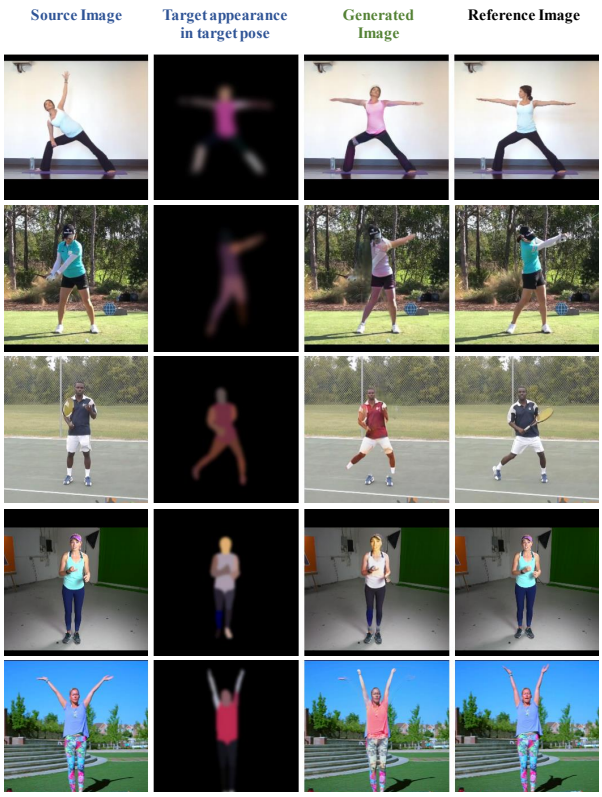

Figure 5. Example results on data augmentation using our framework. Each row is a different observation. The first two columns (source image and the target pose and appearance) are the input to the model and the third column is the generated output. The fourth column shows the reference image from which the pose was sampled.

4.1. Pose Loss

First, we use a sobel edge detector to find the edges of Y_{rand} and Y. While the appearances in Y_{rand} and Y are different, an image of the detected edges does not hold any information about the appearance. Hence the loss $\mathcal{L}_V(e(Y_{rand}), e(Y))$, where $e(y)$ is the detected edges, helps in keeping a check on the pose of the generated image. Also, if the identity of the person is maintained in Y_{rand}, then the calculated edges should mostly be similar to that in Y since edges are detected on those features that we want to retain (such as facial features and patterns on cloths).

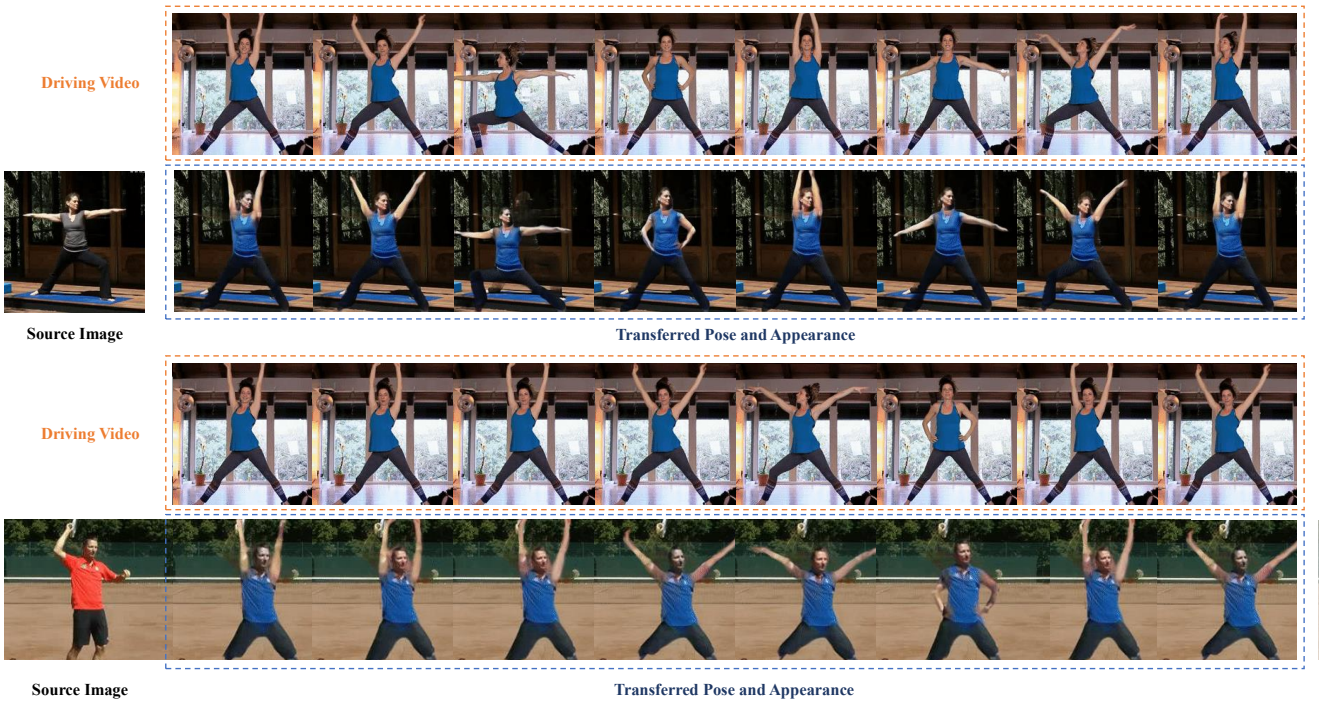

Figure 6. Example results of video animation conditioned on a reference video.

4.2. Appearance Loss

We compare the output image Y_{gen} with Y. It should be noted that during the generation of y_{gen}, the pose is kept constant. Hence this loss $\mathcal{L}_V(Y_{gen}, Y)$ helps in keeping a check on the appearance transfer capabilities of the generator.

4.3. Pose and appearance loss

We compare the output image X_{gen} with X. Unlike the previous loss, in this the generator has to perform both pose and appearance transfer at the same time. This loss $\mathcal{L}_V(X_{gen}, X)$ helps in keeping a check on both pose and appearance transfer capabilities of the generator.

4.4. Adversarial Loss

The aim for this loss is to enforce the images to look more realistic. In a traditional GAN setup, the adversarial loss looks like the following,

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(D, y, y_{gen}) = \mathbb{E}\left[\log D(y)\right] \\ + \mathbb{E}\left[\log\left(1 - D(y_{gen})\right)\right]\end{aligned} \tag{2}$$

.

We employ this adversarial loss thrice in each training iteration, once for each step (Y_{rand},X_{gen}, Y_{gen}). The final adversarial loss is

$$\begin{aligned}\mathcal{L}_{\text{adv}} = \mathcal{L}_{\text{GAN}}(D, y, y_{gen}) \\ + \mathcal{L}_{\text{GAN}}(D, y, y_{rand}) \\ + \mathcal{L}_{\text{GAN}}(D, x, x_{gen}).\end{aligned} \tag{3}$$

,

The final objective for the network is

$$\begin{aligned}\mathcal{L}(\phi) = \mathcal{L}_V(e(Y), e(Y_{rand})) + \mathcal{L}_V(Y, Y_{gen}) \\ + \mathcal{L}_V(X, X_{gen}) + \mathcal{L}_{\text{adv}}.\end{aligned} \tag{4}$$

where $\phi = (G, D, x, y, x_{gen}, y_{gen}, y_{rand})$

5. Experiments and Results

First we split our dataset into training and testing data with a split ratio of 7:3 randomly. Our approach enables many applications such as realistic data augmentation, synthesising realistic video animations and transfer, and virtual try-on of different shades of cloths. We demonstrate the efficacy of our approach on all three applications. We compare our work with the state-of-the-arts in each domain. In all three experiments, we use the exact same model.

5.1. Implementation Details

We randomly sample a pair of image from a randomly chosen video and use one image as the source and the other as the reference. For the randomly sampled appearance, we

Figure 7. Example results of pose transfer using the main generator without and with the face orientation correcting GAN. Each row is a different observation. The first column shows the source image, the second column shows the reference images (the image from which the target pose and appearance was sampled), the third column shows the result of the main generator without the use of the face correcting GAN, and the fourth column shows the results when the face correcting GAN was used along with the main generator.

choose an image from a different video and replace the color values in the source appearance with the sampled appearance.

5.2. Data augmentation

In this, from a single image of a person, we generate a large amount of data containing images of that person in varying poses and appearances. For this, we can randomly select an image and sample various combinations of poses and appearance from our dataset and employ our generator them. To evaluate this, we sample the reference poses from different frames of the same video the source image was selected from. This gives us a reference image to compare the pose in the generated image with. To evaluate the appearance generation, we find the dominant color of each body part in the generated image and compare that to the values in the sampled appearances. Finally to evaluate how well the identity has been retained, we show the edges of the

	SSIM	LPIPS
PG2[10]	**0.854**	0.135
SHUP[1]	0.832	**0.099**
DSC[14]	0.829	0.129
Ours	0.831	0.108

Table 1. A qualitative comparison of the results of pose transfer with previous works.

	RMSE	SSIM
Without FOC GAN		
With FOC GAN		

Table 2. A qualitative comparison of the results with and without the FOC GAN. When the FOC GAN is employed, the outputs are much more similar to the ground truth images.

generated and the reference image. Sample results can be seen in Fig 5. Our framework successfully generate images of the original person in arbitrary poses and appearances. The comparison of our approach with previous works can be seen in Table 1.

5.3. Video animation using attribute transfer

Next we demonstrate the capabilities of our approach in generating video sequences from a single image of a person conditioned on a reference video. In the generated video, the pose and appearance from the reference video is applied frame by frame to the source image and the results frames are combined to form the final video. The results in Fig 6 show that the generated images from our model are temporally consistent. It should be noted that the model was not explicitly trained for this.

5.4. Facial Orientation

In order to evaluate the need and efficacy of the generator for correcting the face orientation, we use our main model on the test dataset first. Then we employ the face orientation correction GAN (FOC GAN) on the generated images and compare the outputs. We show a few example results in Fig 7. It should be noted, that we do not transfer the appearance while training this model. Hence, during this experiment, we only transferred the pose to another random pose in the same video. To quantitatively evaluate this generator, we compare the cropped regions of the face in both of the generated images (without and with the FOC GAN) with the cropped region of the face in the target image. We specifically calculate the root mean square error (RMSE) and structural similarity index measure (SSIM) values of each. We show the comparison in table 2

6. Conclusions

We presented a framework for manipulating both pose and appearance simultaneously in images in a controlled manner. Due to the lack of a dedicated dataset with varying poses and appearances for each individual, we built a method for separating the pose and appearance in the images and augmenting unseen combinations of them over our dataset. To cater to the lack of labelled data for the augmented data that we use for training, we establish a virtual-supervised environment by maintaining a two-way cyclic consistency that keeps a check on both pose and appearance. Furthermore, in order to retain the identity of the face while manipulating its orientation, we also use use a face orientation correcting GAN. Our demonstrations on applications such as realistic data augmentation, video animation synthesis, and pose transfer show that our work can be used in a variety of fields reliably.

References

[1] G. Balakrishnan, Amy Zhao, Adrian V. Dalca, F. Durand, and J. Guttag. Synthesizing images of humans in unseen poses. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8340–8348, 2018.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[3] C. Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A. Efros. Everybody dance now. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5932–5941, 2019.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, M. Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[5] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[7] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[8] Y. Li, C. Huang, and Chen Change Loy. Dense intrinsic appearance flow for human pose transfer. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3688–3697, 2019.

[9] Wen Liu, Wenhan Luo Lin Ma Zhixin Piao, Min Jie, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[10] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems*, pages 405–415, 2017.

[11] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[12] S. Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, B. Schiele, and H. Lee. Learning what and where to draw. *NIPS*, 2016.

[13] S. Reed, A. Oord, Nal Kalchbrenner, V. Bapst, M. Botvinick, and N. D. Freitas. Generating interpretable images with controllable structure. 2017.

[14] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuilière, and Nicu Sebe. Deformable gans for pose-based human image generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[15] M. Zanfir, A. Popa, A. Zanfir, and C. Sminchisescu. Human appearance transfer. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2018.

[16] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[17] 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 Conference on Computer Vision and Pattern Recognition*, pages 2347–2356, 2019.