Supplementary materials for
Robust Disentanglement on Real-World Datasets without Pose-Annotations

We provide additional results obtained by our method in Sec. A and implementation details in Sec. B.

A. Synthesis Results

In Fig. 10 we show some qualitative results of the comparison in Fig. 7.

A.1. Object Image Generation

Since our method does not rely on the existence of pose estimates, it is applicable to a wide range of objects. In Fig. 13 we show additional results of our method obtained on a vehicle surveillance dataset [36].

Fig. 14 shows that our method is not limited to a single object category. Indeed, it can learn shared pose representations across different categories, such as those contained in the norb dataset [34].

A.2. Person Image Generation

In Tab. 3 we provide a qualitative comparison to [13] which highlights the benefits of not requiring keypoint estimates even for domains where keypoint estimators are available. Our approach does not suffer from pose estimation errors and, compared to keypoints, our pose representation is better disentangled from appearance.

We show additional results for Person Image Generation in Fig. 11, Fig. 15 and Fig. 16. Note that our method learns an appearance invariant representation of pose across a wide range of poses and viewpoints (Fig. 11). Fig. 15 shows that it can handle fine grained pose representations such as those required for hands. Fig. 16 demonstrates the applicability of our method for general human actions. Fig. 12 shows interpolation along appearance and pose axes.

A.3. Video Generation

Requiring no pose annotations but only multiple images depicting the same object, our method can be directly applied to video data without additional annotations. Thus, we are also able to perform unsupervised video-to-video translation. We provide examples for the norb dataset (`norb.avi`), the bbc dataset (`bbc.avi`) as well as the ntu dataset (`ntu.avi`).

B. Implementation Details

In this section we provide additional details on the implementation of our method.

B.1. Network Parameters

We use the following notation to describe the network architectures:

- `conv(n)`: a convolutional layer with 3×3-kernel and n filters.
- `down(n)`: a convolutional layer with 3×3-kernel, n filters and stride 2.
- `up(n)`: a convolutional layer with 3×3-kernel, $4n$ filters, followed by a reshuffling of pixels to upsample the feature map by a factor of 2. Also known as subpixel convolution [51].
- `act`: a ReLU activation.
- `res`: a residual block [18]: The input feature map plus the ReLU activated input feature map followed by a convolutional layer with 3×3-kernel with as many filters as the input feature map has channels.

Depending on the dataset, the generated images have resolution 64×64 (sprites dataset), 96×96 (norb dataset) or 128×128 (remaining datasets). Both encoders E_π and E_α have the same architecture:

- Encoders at resolution 64×64: `conv(16), res, down(32), res, down(64), res, down(128), res, down(256), res, res, res, res, res, act, conv(16).`
- Encoders at resolution 96×96: `conv(32), res, down(64), res, down(128), res, down(128), res, down(256), res, down(256), res, res, res, res, res, act, conv(16).`

- Encoders at resolution 128×128: `conv(32), res, down(64), res, down(128), res, down(128), res, down(256), res, down(256), res, down(256), res, res, res, res, res, act, conv(16).`

The decoder D receives both π and α. Each of them is processed seperately by four `res` blocks and the result is concatenated. Depending on the resolution, the remaining decoder architecture is described by:

- Decoder at resolution 64×64: `conv(256), res, up(128), res, up(64), res, up(32), res, up(16), res, conv(3) .`
- Decoder at resolution 96×96: `conv(256), res, up(256), res, up(128), res, up(128), res, up(64), res, up(32), res, conv(3) .`
- Decoder at resolution 128×128: `conv(256), res, up(256), res, up(256), res, up(128), res, up(128), res, up(64), res, up(32), res, conv(3) .`

The classifiers T and T' have the same architecture. Both receive π and α, and process them seperately with a `conv(512)` layer, followed by four `res` blocks. The result is activated and processed through a final `conv(512)` layer before the output is computed as the inner product of the two resulting feature maps.

B.2. Model Parameters

For all experiments, we use $b_\gamma = l_\gamma = 10^{-2}$ and $\mu = 10^{-1}$. $p(\pi|x_2)$ and $r(\pi)$ are both modeled as Gaussian distributions of unit variance. The latter has a mean of zero and E_π estimates the mean of the first. We use the reparameterization trick [29] to obtain low variance estimates of the gradient. Depending on the dataset, we implement the negative log-likelihood $\mathcal{L}_{\text{rec}}$ with a l_2 loss (sprites), a perceptual loss [25] (norb) or a perceptual loss together with a discriminator loss as in [11], weighted by 10^{-3} (remaining datasets).

B.3. Optimization Parameters

We train our model over batches of size 16 for 100000 steps. We use the Adam optimizer [26] with an initial learning rate of $2 \cdot 10^{-4}$ linearly decayed to zero. We set $\beta_1 = 0.5$ and $\beta_2 = 0.9$. I_T is calculated with an exponential moving average with decay parameter 0.99.

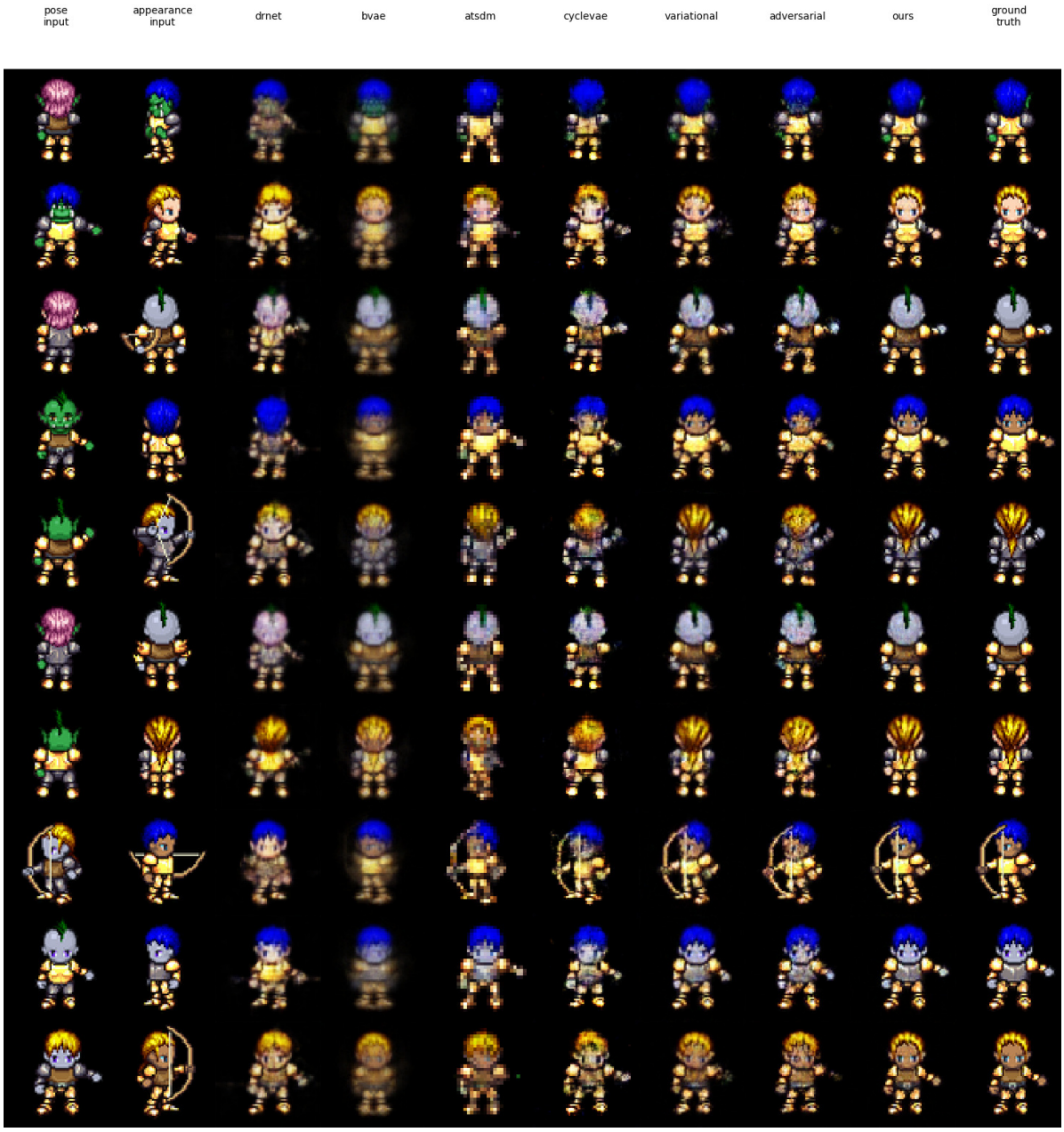

Figure 10. Qualitative results for the comparison in Fig. 7.

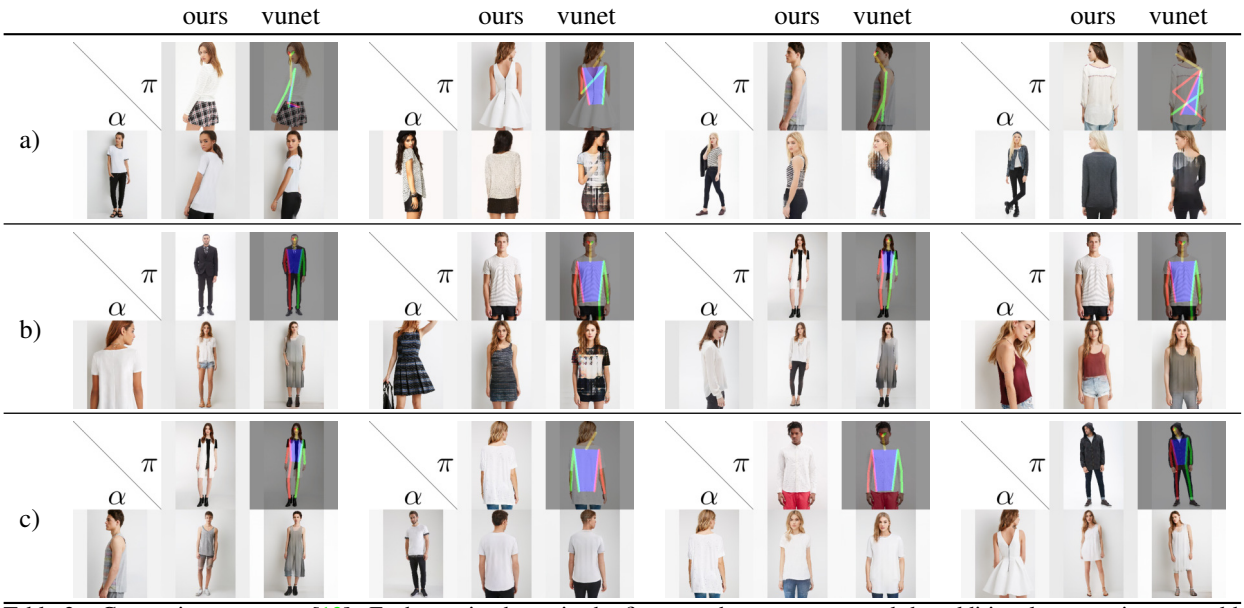

Table 3. Comparison to vunet [13]. Each matrix shows in the first row the pose target and the additional pose estimate used by vunet, and in the second row the appearance target followed by the synthesis of our method and vunet. a) vunet relies on existing pose estimators making it sensitive to estimation errors; our method always uses the direct target image for the pose. b) vunet also relies on pose estimates to obtain localized appearance representations, which can lead to complete failure at capturing the appearance. c) instead of learning a pose representation, vunet assumes that keypoints are good pose representations, but subtle information, e.g. shoulder width, still reveals information about appearances, e.g. gender. Men synthesized in poses estimated from women obtain a feminine look and vice versa. In contrast, our method is designed to learn completely disentangled representations and, in particular, learns gender-neutral pose representations.

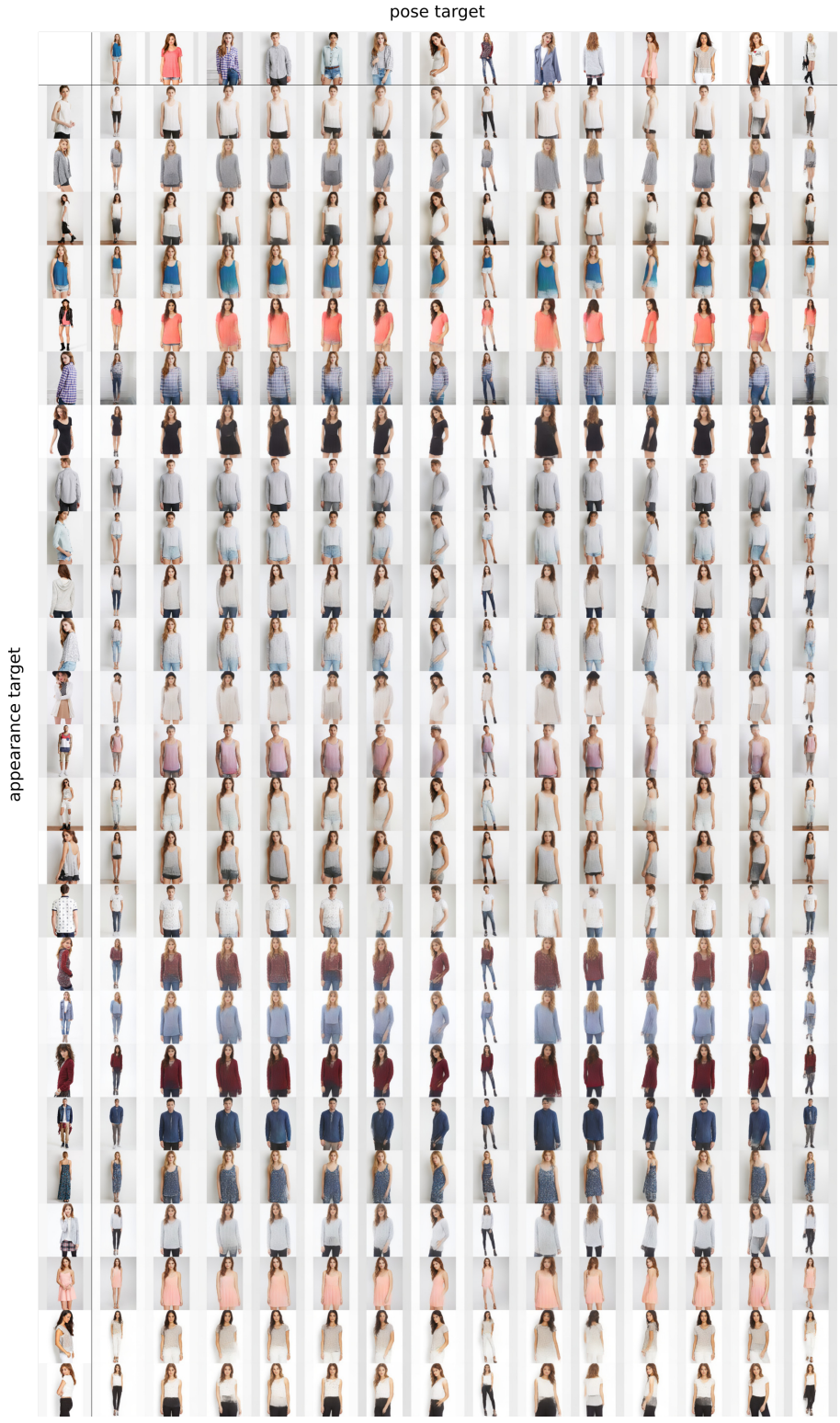

Figure 11. Retargeting on the DeepFashion dataset [37, 38]. Note how our method is able to retarget appearances to a wide range of poses, including a change from half-body to full-body views. Similiarly, large appearance changes (e.g. changes in gender) are possible while retaining the pose. Training requires only pairs of images containing the same appearance.

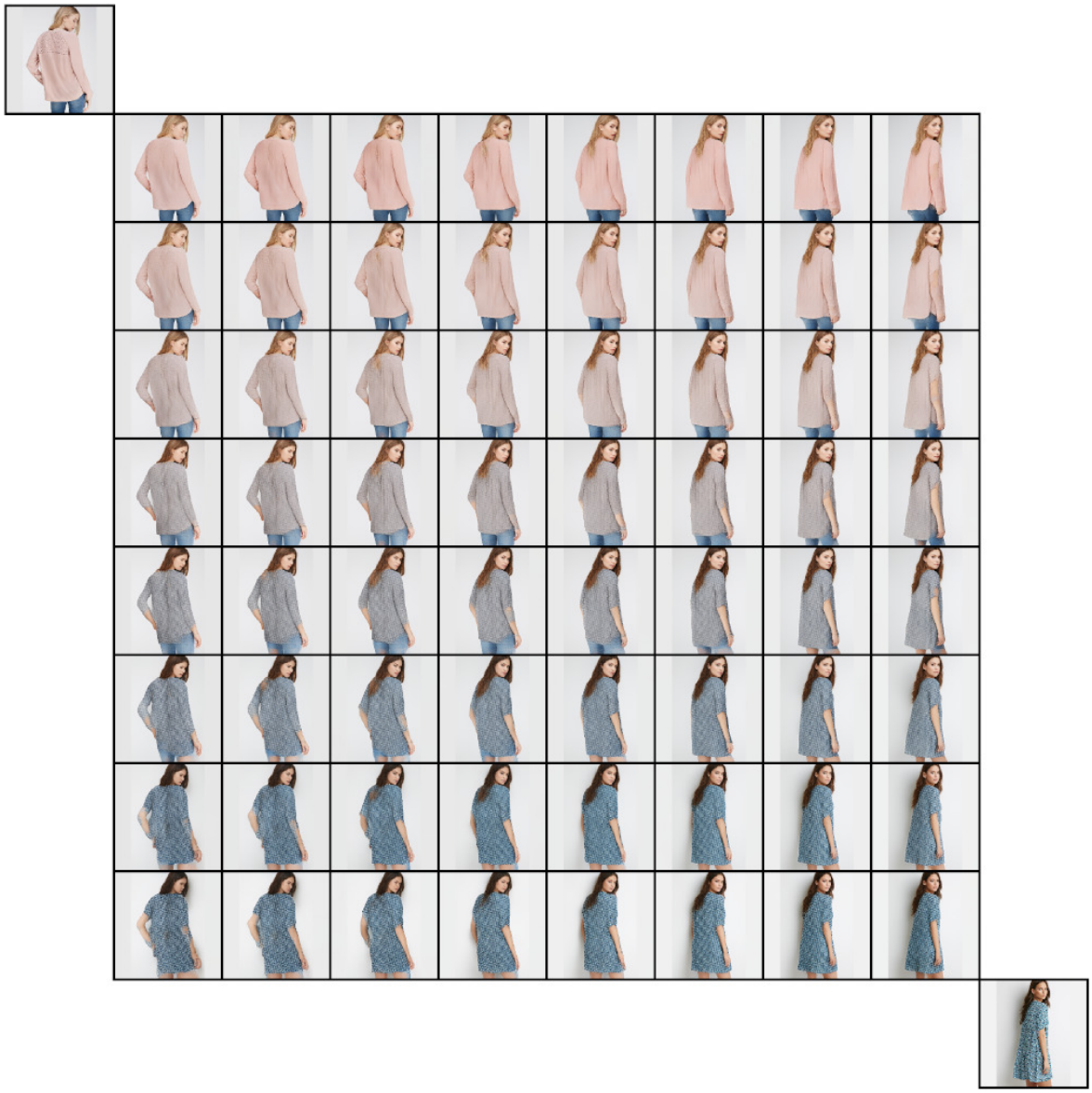

Figure 12. Interpolating between appearance (vertical direction) and pose (horizontal direction).

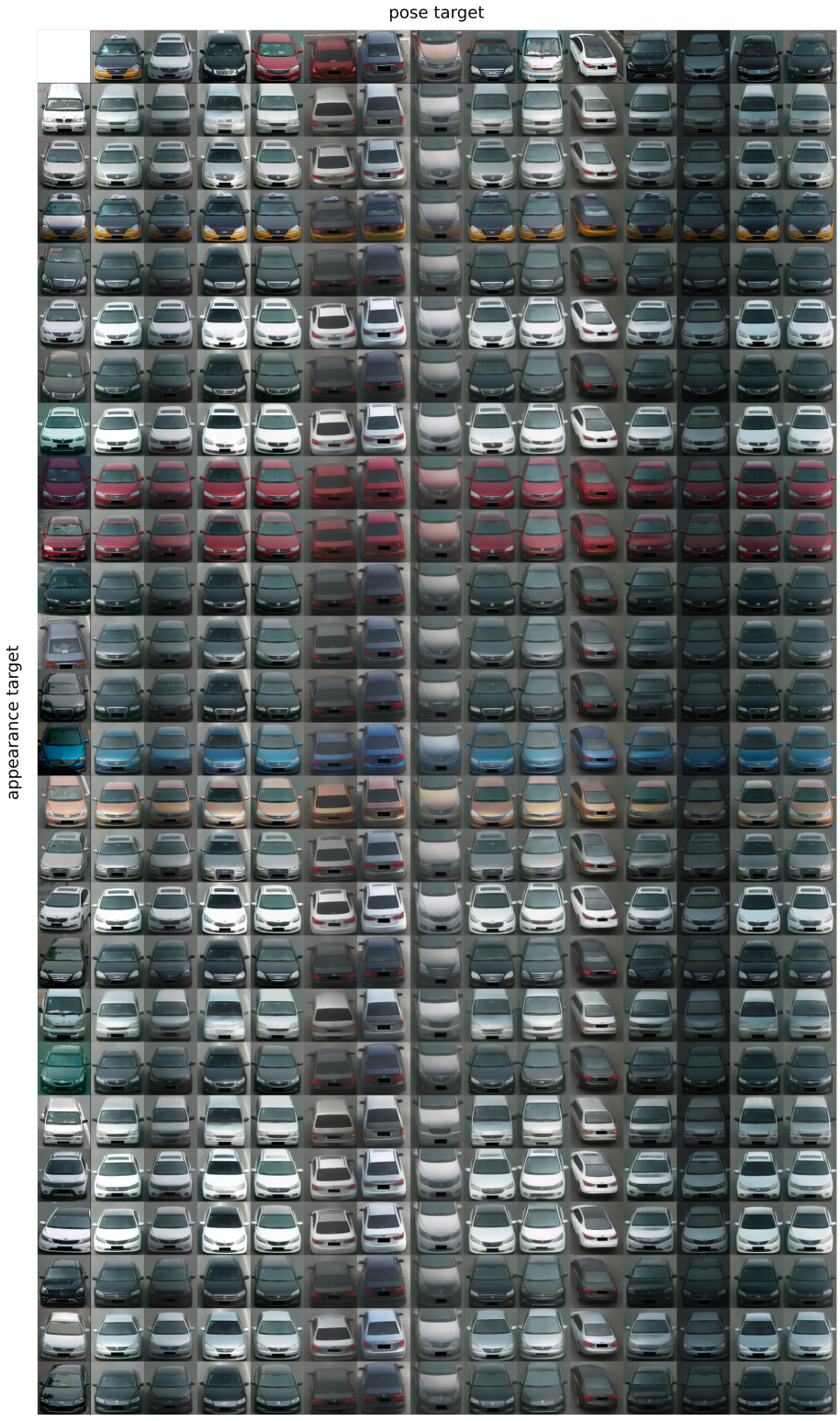

Figure 13. Retargeting on the PKU Vehicle ID [36] dataset. Without changes in the architecture, our method handles both rigid objects, as seen here, as well as deformable and articulated objects such as humans.

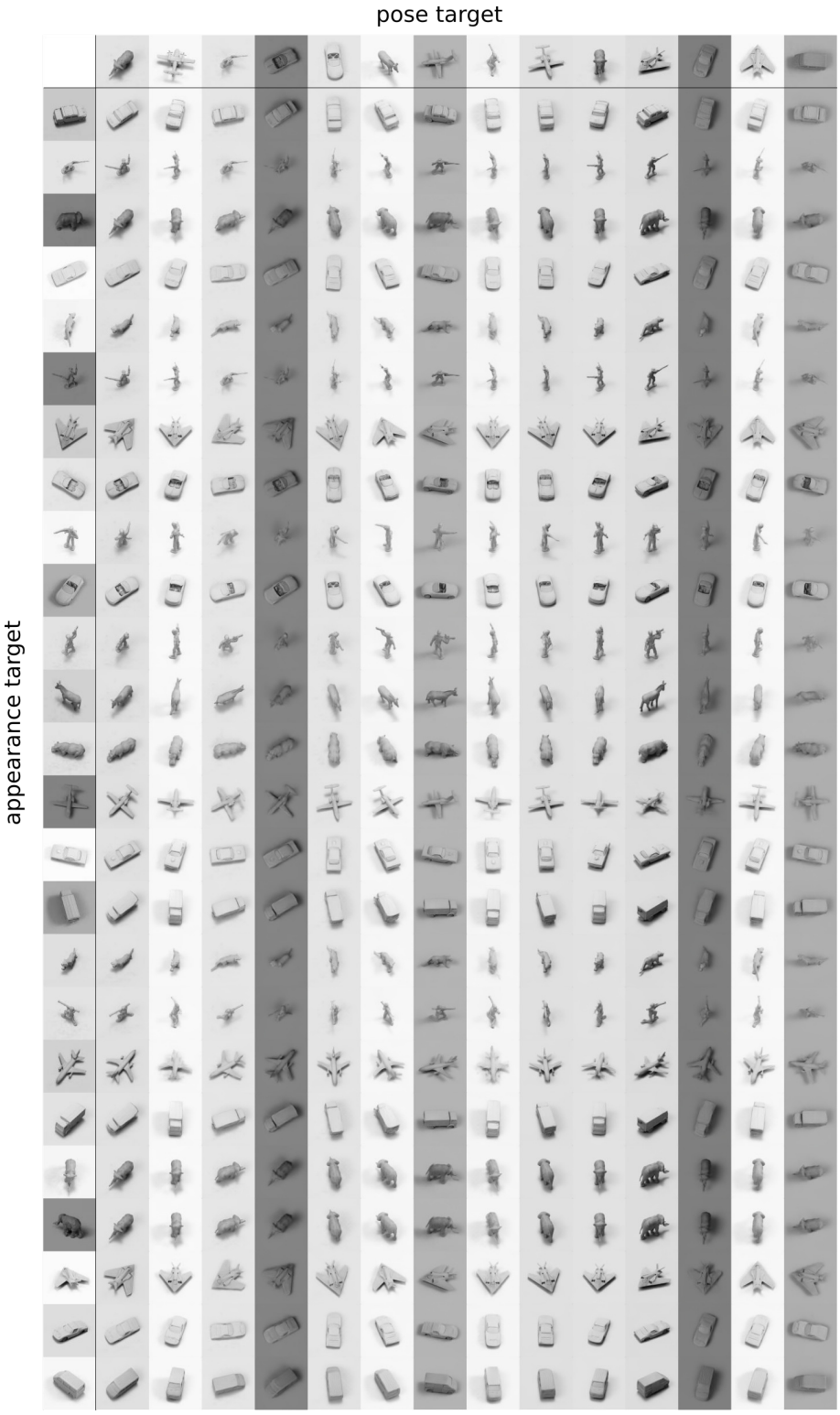

Figure 14. Retargeting on the Norb dataset [34]. Our method successfully finds a shared representation for pose, which can be used to retarget poses across different object categories. An animated version can be found in `norb.avi`, where the target pose is rotated and after each full turn, the elevation is increased.

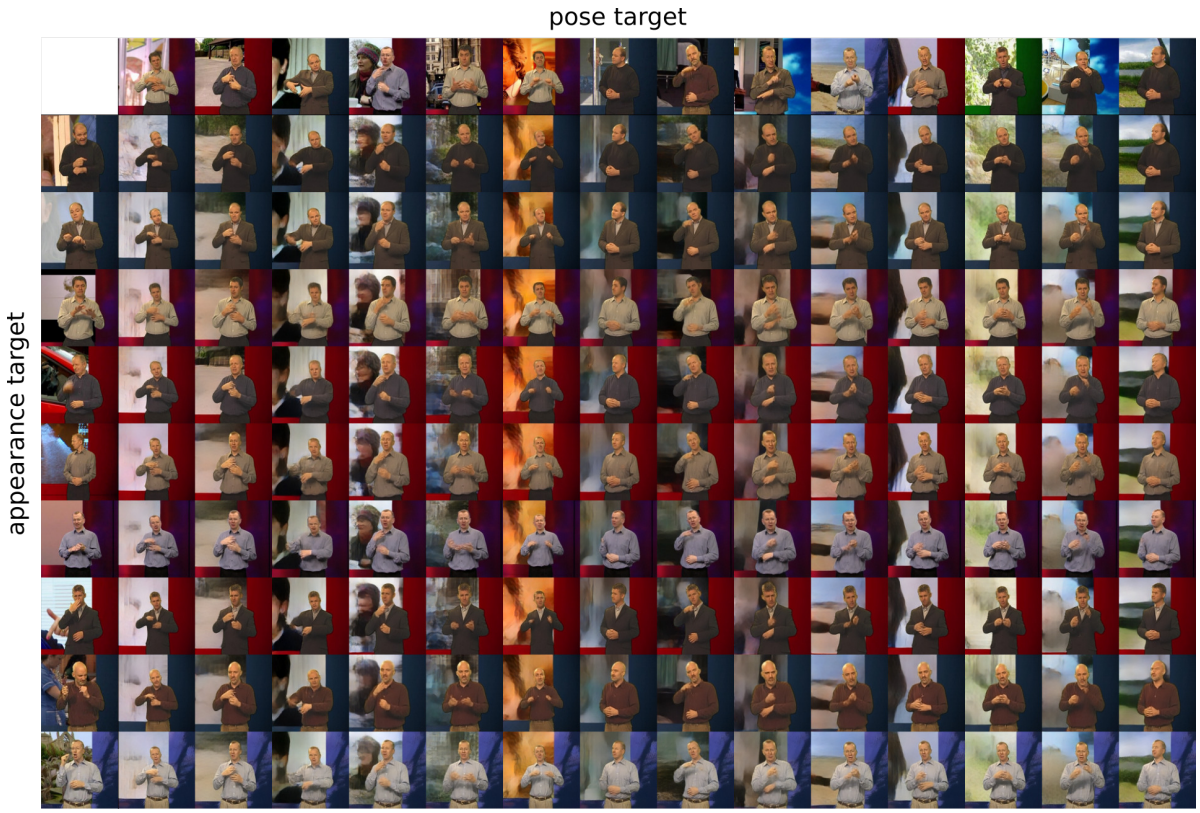

Figure 15. Retargeting on the BBC Pose dataset [6]. No annotations are required. Our method can utilize different frames from a video to learn the transfer task. An animated version can be found in `bbc.avi`.

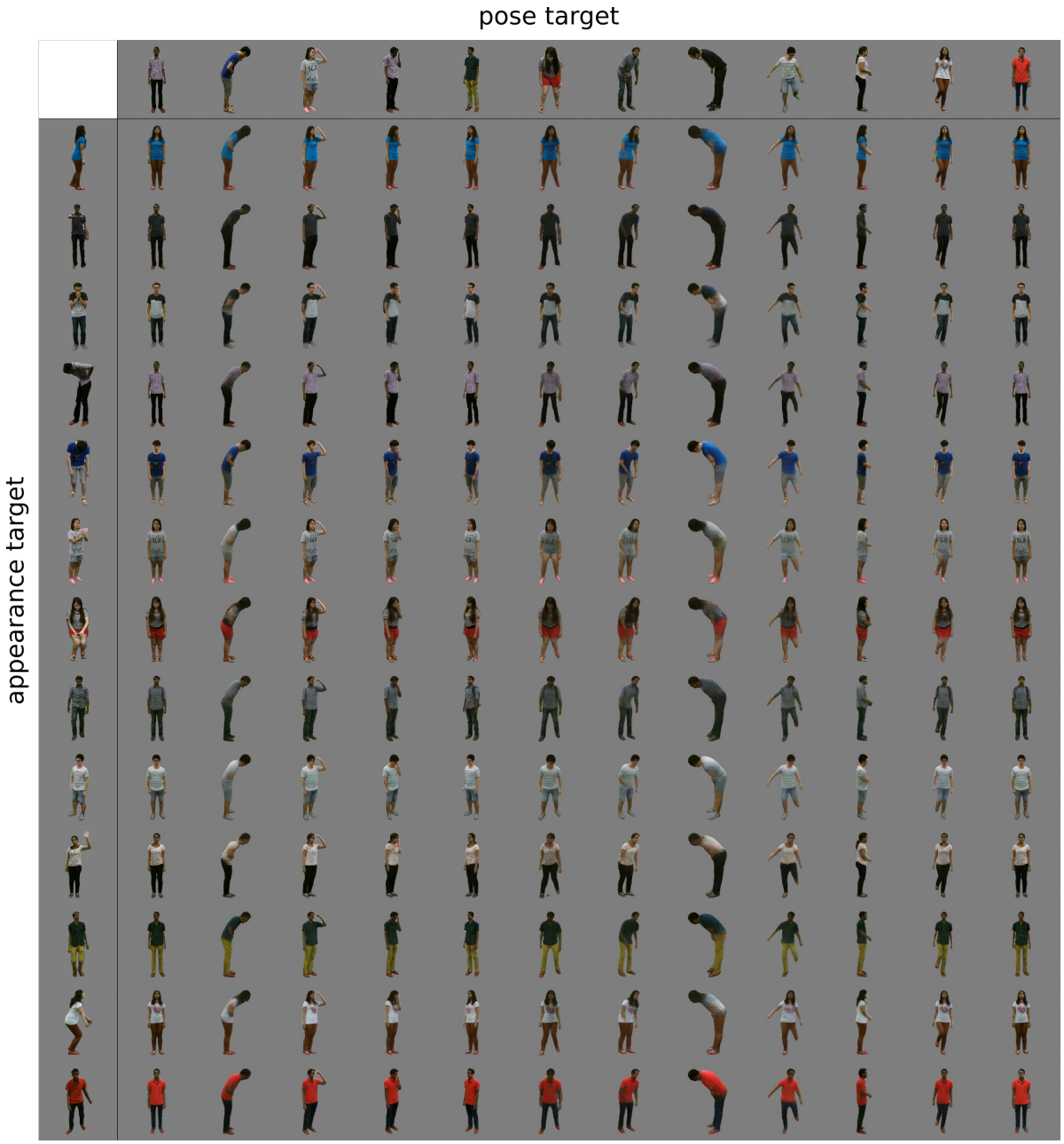

Figure 16. Retargeting on the NTU dataset [50]. Again, our model directly learns to disentangle pose and appearance using only video data without requiring additional annotations. An animated version can be found in `ntu.avi`.