TemporalUV: Capturing Loose Clothing with Temporally Coherent UV Coordinates (Supplementary)

You Xie[†], Huiqi Mao[‡], Angela Yao[‡], Nils Thuerey[†]
[†]Department of Informatics, Technical University of Munich
[‡]Department of Computer Science, National University of Singapore
{you.xie, nils.thuerey}@tum.de, huiqi.mao@u.nus.edu, ayao@comp.nus.edu.sg

In the following, we will first illustrate more details about UV preprocessing, such as UV extension (section A), UV optimization (section B), and UV temporal relocation (section C). Then, additional details about our training will be given in section D. In section E and section F, we will further discuss our evaluation and results.

A. UV extension

In this section, we provide further details about UV extension (section 4.1 of the main paper). Given an image, we first remove the background to obtain I_t. Note that the DensePose model is actually an IUV model, i.e. the UV coordinates P_t^r has an additional I channel that encodes the 24 individual parts of the body (see Figure 11b). Subsequently, we unwrap the surface in a part-by-part manner based on the individual parts before applying UV extrapolation to P_t^r. We assign the labels of the extended parts manually according to their location, e.g. we label the hair on the left side with the same I values as the left head part. An example of our labelling is shown in Figure 11.

After labelling the I values, we linearly extrapolate values for empty positions with neighbouring points with the same I value inside a small area with size 3×3. Then we map the new extrapolated point from I_t to T_t and apply a *virtual mass-spring* system in T_t to reduce the surface distortion. We assume that all neighbouring points $O_1, O_2, ..., O_n$ inside a region of size 40×40 are connected with this new extrapolated point O_0 via *virtual* springs in the texture. The pushing/pulling forces $\mathbf{f_1}, \mathbf{f_2}, ..., \mathbf{f_n}$ from neighbour points will drive O_0 to the direction of $\sum_{i=1}^{n} \mathbf{f_i}$ for every step until O_0 arrives at an equilibrium state with a new position, where $\sum_{i=1}^{n} \mathbf{f_n} = \mathbf{0}$.

From our experience, we found that applying pure pushing forces can generate valid results, since the parts that need to be extended are always located at the outline of the body. After applying pushing forces, we switch the forces to pulling in order to make the texture more compact. To summarize, we extend UV coordinates in the UV space and then continue with a *virtual mass-spring* system in the texture map to reduce the unwrapped surface distortion. For P_t^r generated from the SMPL model, which only contains UV channels, the UV extension step is performed without labelling the I values, i.e. we directly extend P_t^r to the full silhouette with linear extrapolation before applying the *virtual mass-spring* system.

Figure 11. Example of a) I_t and b) I channel from DensePose P_t^r; c) after the labelling, the I channel is extended to the full silhouette.

B. UV optimization

In section 4.2 of the main paper, we discussed the steps to obtaining $P_t^o = \arg\min \left\| I_t - I_t^{'} \right\|_F^2$. In this section, we will illustrate how we calculate $\frac{\partial \mathcal{L}_{\text{app}}}{\partial P_t}$ to optimize P_t with the objective function

$$\mathcal{L}_{\text{app}} = \left\| I_t - I_t^{'} \right\|_F^2, \tag{14}$$

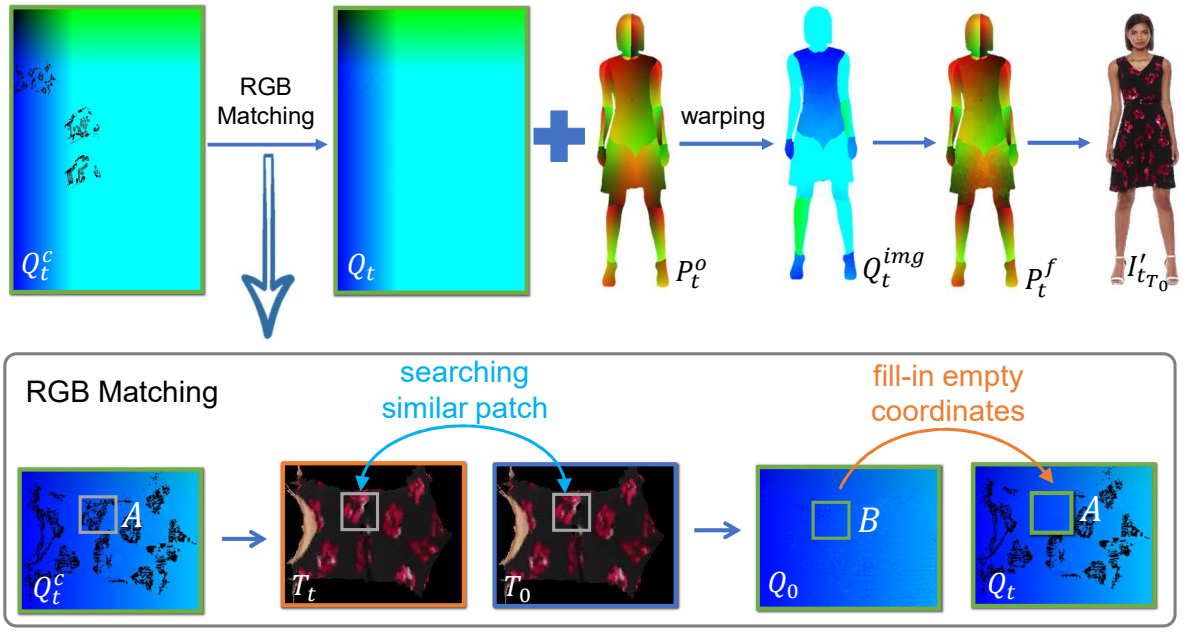

Figure 12. RGB matching step of UV temporal relocation. Empty positions in Q_t^c are filled via patch matching between T_0 and T_t.

from which initially we will have

$$\frac{\partial \mathcal{L}_{\text{app}}}{\partial P_t} = \frac{\partial \mathcal{L}_{\text{app}}}{\partial I'_t} \times \frac{\partial I'_t}{\partial P_t}. \tag{15}$$

We compute the gradient $\frac{\partial I'_t}{\partial P_t}$ via the intermediate warping grids $\omega_T(P_t^r)$ and $\omega_I(P_t^r)$ from UV mappings

$$T_t = \mathcal{W}(I_t,\, \omega_T(P_t)) \quad \text{and} \quad I'_t = \mathcal{W}(T_t,\, \omega_I(P_t)). \tag{16}$$

And relationship between $\omega_I(P_t)$ and P_t can be written as

$$\omega_I(P_t(\mathbf{x})) = \mathbf{x} - P_t(\mathbf{x}). \tag{17}$$

This gives:

$$\begin{aligned}
\frac{\partial I'_t}{\partial P_t} &= \frac{\partial I'_t}{\partial T_t} \times \frac{\partial T_t}{\partial P_t} + \frac{\partial I'_t}{\partial \omega_I(P_t)} \times \frac{\partial \omega_I(P_t)}{\partial P_t}; \\
\frac{\partial T_t}{\partial P_t} &= \frac{\partial T}{\partial \omega_T(P_t)} \times \frac{\partial \omega_T(d(I_t))}{\partial P_t}; \\
\frac{\partial \omega_T(P_t)}{\partial P_t} &= \frac{\omega_T(P_t)}{\partial \omega_I(P_t)} \times \frac{\partial \omega_I(P_t)}{\partial P_t}.
\end{aligned} \tag{18}$$

In an implementation, we can conveniently obtain $\frac{\partial \omega_T(P_t)}{\partial P_t}$ via

$$\frac{\partial \omega_T(P_t)}{\partial P_t} = \mathcal{W}(\frac{\partial \omega_I(P_t)}{\partial P_t}, \omega_T(P_t)), \tag{19}$$

and $\frac{\partial \omega_I(P_t)}{\partial P_t}$ can be computed via equation 17. This provides $\frac{\partial I'_t}{\partial P_t}$ for optimization and learning steps.

We apply a gradient descent optimizer with $\alpha_1 = 100$ and $\alpha_2 = 10$ for regularizer L_r. Due to the large distance between P_t^e and P_t^o, we use a large learning rate, such as 10.0, to accelerate the optimization procedure. We found that promising results can be obtained after ca. 16500 steps. UV optimization takes about 75s/frame, measured for resolution 1200×800 with a NVIDIA RTX 2080 Ti GPU. All frames can be optimized in parallel.

C. UV temporal relocation

In this section, we will introduce how we use RGB matching in section 4.3 of the main paper to fill in the empty areas in Q_t^c. Specifically, we assume that similar, nearby texture patches in T_0 and T_t will have the same correspondences in Q_0 and Q_t. For a missing area A in Q_t^c , as shown in Figure 12, we locate the region with the same position as A in T_t. We record the values of Q_t^c and T_t inside region A with $[Q_t^c]_A$ and $[T_t]_A$, respectively. Then we can find a region B in T_0 via

$$min\ ||[T_t]_A - [T_0]_B||_F^2, \tag{20}$$

and $[Q_0]_B$ are used to fill in $[Q_t^c]_A$ to obtain Q_t.

Afterwards, Q_t can be mapped to Q_t^{img} in the image space via P_t^o. $Q_t^{img}(\mathbf{u})$ is directly our P_t^f if P_t^r is represented with the same coordinates system as $\mathbf{u}$, such as the P_t^r unwrapped from the SMPL model. But for the P_t^r from the DensePose model, which uses a different coordinate system from $\mathbf{u}$, we additionally transform $Q_t^{img}(\mathbf{u})$

	PSNR↑	LPIPS↓ $\times 10^{-2}$	tOF↓ $\times 10^{4}$	tLP↓ $\times 10^{-2}$	T-diff↓ $\times 10^{5}$
P_t^r	22.1	8.1	1.69	**1.0**	5.42
V_1	23.8	7.0	1.95	1.7	**4.33**
V_2	**23.9**	**6.7**	1.76	1.3	4.67
V_3	23.6	6.8	**1.68**	1.2	4.55

Table 2. Quantitative comparisons between P_t^r and our different versions without cropping to fit $=P_t^r$. Our method show significant improvements for both spatial (PSNR and LPIPS) and temporal (tOF, T-diff) evaluation metrics. Evaluations with full shape lead to further improved PSNR and LPIPS evaluations for our results.

into P_t^f.

D. Training details

In this section, more details about temporal UV model training (section 4.4 of the main paper) will be illustrated. The DensePose model outputs UV coordinates with an extra I channel to classify different body parts. Below, we use I subscripts to denote the I channel of a UV coordinate, e.g., $G_I(P_t^r)$ refers to the I channel of $G(P_t^r)$. Then, for P_t^r from the DensePose model, we use an extra cross-entropy loss

$$L_I = -\boldsymbol{e}_{[P_t^f]_I} log G_I(P_t^r) \tag{21}$$

for I channel constraint, where $-\boldsymbol{e}_{[P_t^f]_I}$ is a one-hot vector indicating the $[P_t^f]_I^{\text{th}}$ I channel with $-\boldsymbol{e}_{[P_t^f]_{Ij}} = 1$ if $j = [P_t^f]_I$. We train G for DensePose P_t^r with the architecture shown in Figure 13, and all of the discriminators D_s, D_t, and D_{img} follow the same encoder structure, as shown in Figure 14. For UV data without an I channel, e.g., P_t^r generated from SMPL models, our pipeline is still applicable by training without L_i and removing the I channel part in G.

We apply gradient clipping for the gradients from L_s^{img} and L_t^{img} to stabilize the training of G. Parameters λ_2 and $\lambda_{uv,s}$ start from 200 and 10, respectively. They are decreased with rate 0.99 for every 1000 steps. On the other hand, $\lambda_{img,s}$, λ_{smo}, $\lambda_{uv,t}$, and $\lambda_{img,t}$ start from 0.001, 0.1, 1, and 1, respectively, but they are gradually increased with rate 1.01 for every 1000 steps.

E. Evaluation of results

In section 5 of the main paper, we follow [1] to evaluate temporal coherence of the results and estimate the differences of warped frames, i.e., T-diff $= \|I_{g_t}, \mathcal{W}(I_{g_t}, v_t)\|_1$. In our setting, we use the UV coordinates to calculate v_t, so that T-diff will purely be influenced by P_t. We first warp all the point coordinates $\mathbf{x}$ in I_{g_t} to the texture space, then we can calculate the displacement of all the points from I_{g_t} to $I_{g_{t+1}}$:

$$v_t^{texture} = \mathcal{W}(c_{img}, \omega_T(P_{t+1}^g)) - \mathcal{W}(c_{img}, \omega_T(P_t^g))), \tag{22}$$

where $c_{img}(\mathbf{x}) = \mathbf{x}$. Then v_t can be obtained with:

$$v_t = \mathcal{W}(v_t^{texture}, \omega_I(P_{t+1}^g)). \tag{23}$$

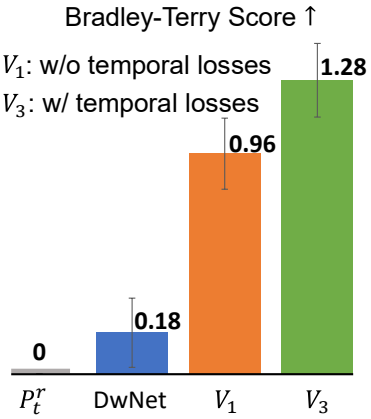

Figure 18. User study for the red-black dress case. Our full version V_3 significantly improves over V_1 and DwNet.

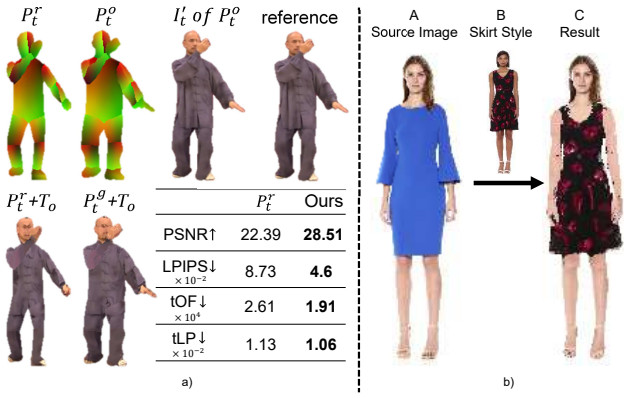

Figure 19. a) Results of the Tai-Chi dataset. b) Pose-guided generation application. Our model is generalized to different poses from different videos.

In Table 1 of the main paper, we show quantitative comparisons between P_t^r and our different versions, which are made fair by cropping to fit $=P_t^r$. These results show improvements for both spatial and temporal evaluations. Here, we also show comparisons of those versions without cropping in Table 2. We can see that our versions outperform P_t^r even further in terms of spatial quality. P_t^r performs the best with tLP, as P_t^r cannot generate the extended skirt and hair parts, which significantly decreases the area for evaluation. Here, we also can see that V_3 shows similar spatial quality as V_2 but an improved temporal coherence. Please refer to the supplementary video to see the improved temporal coherence of the synthesized sequence.

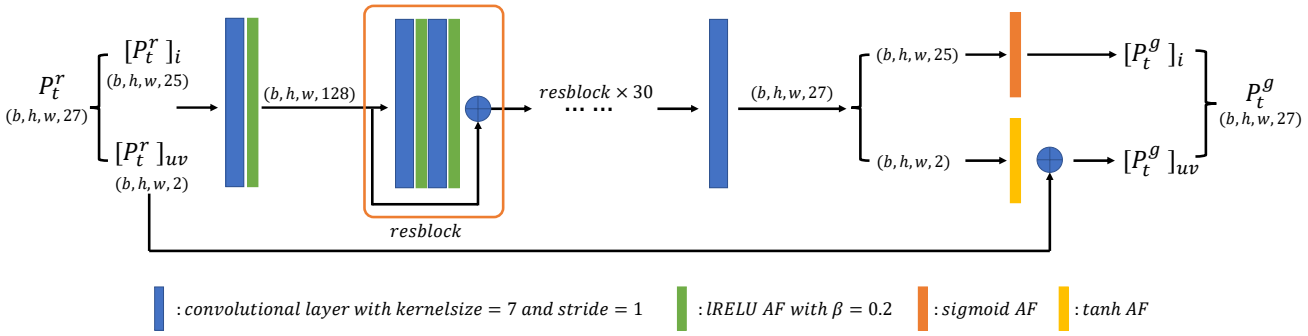

Figure 13. Generator structure for training with P_t^r from the DensePose model. The corresponding part of I channel will be removed when training with P_t^r generated from the SMPL model.

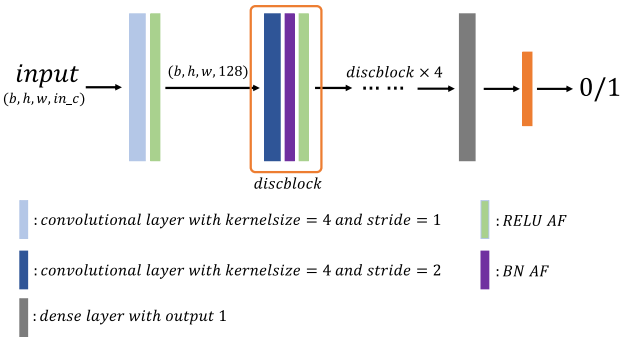

Figure 14. Architecture of the discriminator networks, such as D_s, D_t, and D_{img}. Input channels in_c for D_s, D_t, and D_{img} are 2, 6, and 9, respectively.

Figure 15. For parts that do not interact with the clothing, such as head and feet (in the yellow rectangles), we reuse texture of T_t to reconstruct those parts in I'_t. For the rest of the parts (blue rectangle), we use the constant texture T_o.

We conducted a user study to evaluate coherence (see Figure 18. Raw DensePose P_t^r is the baseline, while models V_1 and V_3 are trained with P_t^f, without and with temporal losses, respectively. V_3 gives significantly improved evaluations from the participants. We also outperform DwNet with high confidence, confirming the effectiveness of the temporal losses and the tOF and tLP evaluations.

F. More results

Similar to Figure 6 in the main paper, we show more examples of P_t^o in Figure 16. It becomes visible that our extrapolation and optimization pipeline can significantly improve the spatial quality of UV coordinates and recover the full silhouette. We also show more virtual try-on applications in Figure 17, from which we can see that P_t^g generated from our model can be efficiently applied to change clothing texture. For the virtual try-on application, we replace the clothing texture from the original constant texture T_o with a new texture for an updated constant texture T_o. Then, the new sequence is synthesized with the updated T_o. It is worth pointing out that since we focus on the clothing, we reuse the texture of other parts from the source video so that the evaluation can focus on these regions. As shown in Figure 15, the head and feet do not interact with the clothing, so we reuse the texture of those parts from T_t to synthesize I'_t. Our pipeline can also be applied to datasets containing more diverse motions and complex backgrounds, such as the Tai-Chi dataset (see results in Figure 19a). Results are in line with the conclusions of our main paper: our optimization result P_t^o successfully recovers the missing UV coordinates and generates full images I'_t. After training, our synthesized result $(P_t^g + T_o)$ is closer to the reference than DensePose $(P_t^r + T_o)$ for temporal and spatial evaluations. Lastly, our models are specific to garment silhouettes, not individual videos. Retraining is only necessary if the silhouette changes. E.g. in Figure 19b, the model is trained with the sequence B (with sleeveless dress) and can be conditioned on poses in A (with long sleeves) to generate C. We aim for this direction since the silhouettes of common

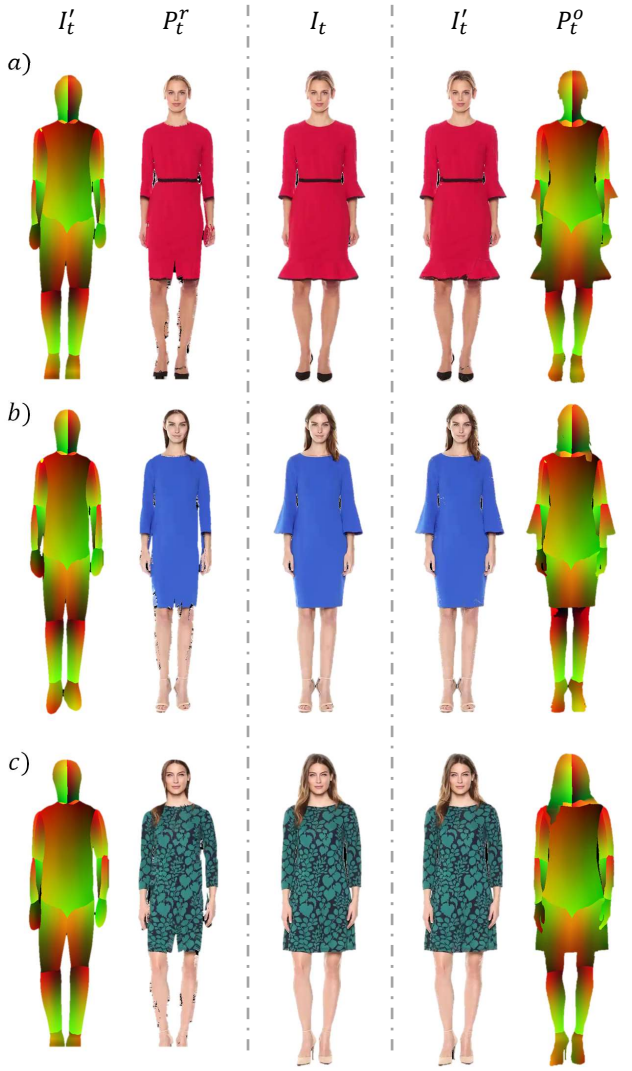

Figure 16. Additional results comparing raw UV coordinates P^r_t (the fist column) with UV coordinates P^o_t (the fifth column) after our optimization step. Here we also show I'_t for P^r_t (the second column) and P^o_t (the fourth column). We can see that the results I'_t generated with P^o_t are closer to the reference I_t.

clothes are limited.

References

[1] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1105–1114, 2017. 3

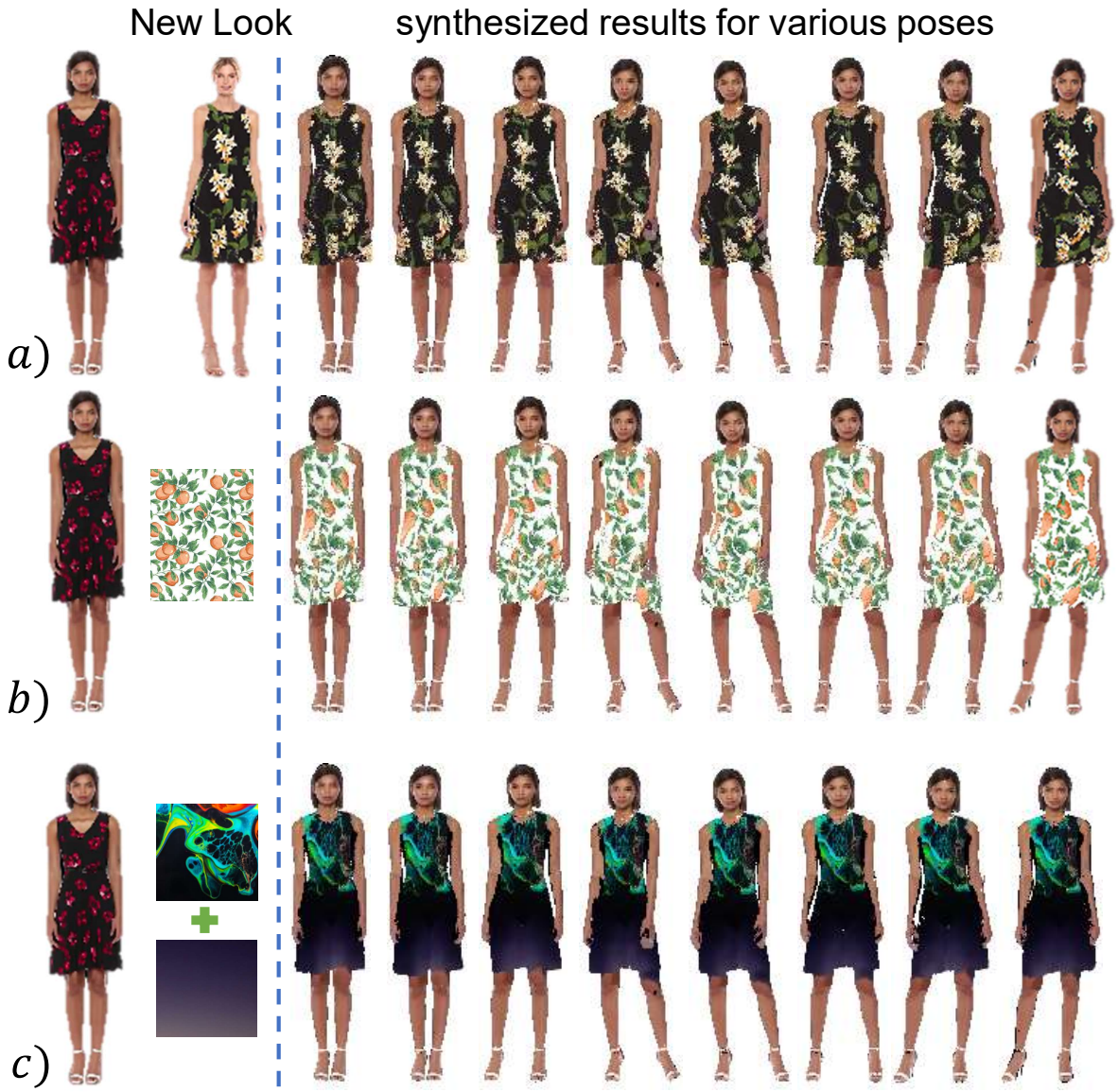

Figure 17. Additional virtual try-on results. Different textures, regardless of complexity, can be applied as a new look to our source video very efficiently.