Pose-guided Action Separable Generative Adversarial Net for Novel View Video Synthesis (Supplementary Material)

In this document, we first provide the details of each component of the proposed network. In Table 1, we show the details of the pose transformation module . In Table 2, we show how we extract multi-scale features from image prior. Similarly, our discriminator and perceptual loss are based on the same structure. One thing worth noting is that we modify the last two layers of vgg16 to output a 2-dimensional fully connected layer and a sigmod layer as our discriminator. In Table 3, we show how we estimate the pose from the image prior using a prediction head. In Table 5, we show that how to transform the feature from coarse to fine-grained level. Finally, using the video decoder as shown in Table 4, we generate the output action video. In Figure 1, we compare our methods with others. Second, we show more qualitative results of generated frames in Figure 3 - 4. In addition, we also visualize the generated pose of our recurrent pose transformation module. As shown in Figure 2, our module not only learns the motion from original pose sequence, but also transfers the motion into the target view of the input pose. Moreover, we also include a demo video for novel view action prediction.

Name	Layer	Input	Neurons	Output Dims
	-	-	-	(C $\times$ M)
pose_full1	Linear	p_{s1}	100	2×25
pose_full2	Linear	p_{s2}	100	2×25
pose_full3	Linear	pose_full1+pose_full2	100	2×25
vp_full1	Linear	θ_1	25	1×25
vp_full2	Linear	θ_2	25	1×25
vp_full3	Linear	vp_full1+ vp_full2	50	2×25
T_full1	Linear	vp_full3+ pose_full3	128	2×64
T_full2	Linear	T_full1	256	2×128
T_full3	Linear	T_full2	512	2×256
T_full4	Linear	T_full3	1024	2×512
T_full5	Linear	T_full4	512	2×256
T_full6	Linear	T_full5	256	2×128
T_full7	Linear	T_full6	128	2×64
T_full8	Linear	T_full7	50	2×25
-	ADD	p_a	-	

Table 1: Network details of the $\mathcal{P}_T$, which is used to transform the pose into target view. There are three different modules in this network. The first one is the pose transformation module that takes the subsequent source poses as input and determines the change in pose. Second, the change in viewpoint estimator which takes the source and target viewpoints and learns a viewpoint deviation in latent space. The last module takes the estimated change in pose and transforms it to the target viewpoint with the help of latent encodings for change in viewpoint. Finally, we will take the transformed pose motion to conduct an element-wise addition to our estimated pose and generate the target pose for next time-step.

Limitations We would like to discuss the limitations of our approach. As the results 1 show, our PAS-GAN shows very high-quality generation results both from frame-level and video-level. In fact, the blur produced by the previous method are greatly eliminated. However, our method also produces results with some artifacts that can be seen in the videos and frames. We analyze that this artifact is caused by the fact that our decoder is a video-level 3D decoder (although we use the same decoder as RTNet [4], the artifacts caused by it are obscured by the blur). This is-

Name	Layer	Input	Kernel Dims (H $\times$ W)	Strides (H $\times$ W)	Output Dims (H $\times$ W $\times$ C)
Conv1a	2D Conv	P^j	3×3	1×1	$112 \times 112 \times 64$
ReLU1a	ReLU	Conv1a	-	-	$112 \times 112 \times 64$
Conv1b	2D Conv	ReLU1a	3×3	1×1	$112 \times 112 \times 64$
ReLU1b	ReLU	Conv1b	-	-	$112 \times 112 \times 64$
MaxPool1	2D Max Pool	ReLU1b	2×2	2×2	$56 \times 56 \times 64$
Conv2a	2D Conv	MaxPool1	3×3	1×1	$56 \times 56 \times 128$
ReLU2a	ReLU	Conv2a	-	-	$56 \times 56 \times 128$
Conv2b	2D Conv	ReLU2a	3×3	1×1	$56 \times 56 \times 128$
ReLU2b	ReLU	Conv2b	-	-	$56 \times 56 \times 128$
MaxPool2	2D Max Pool	ReLU2b	2×2	2×2	$28 \times 28 \times 128$
Conv3a	2D Conv	MaxPool2	3×3	1×1	$28 \times 28 \times 256$
ReLU3a	ReLU	Conv3a	-	-	$28 \times 28 \times 256$
Conv3b	2D Conv	ReLU3b	3×3	1×1	$28 \times 28 \times 256$
ReLU3b	ReLU	Conv3b	-	-	$28 \times 28 \times 256$
Conv3c	2D Conv	ReLU3b	3×3	1×1	$28 \times 28 \times 256$
ReLU3c	ReLU	Conv3c	-	-	$28 \times 28 \times 256$
MaxPool3	2D Max Pool	ReLU3c	2×2	2×2	$14 \times 14 \times 256$
Conv4a	2D Conv	ReLU2b	3×3	1×1	$14 \times 14 \times 128$
ReLU4a	ReLU	Conv4a	-	-	$14 \times 14 \times 128$
Conv4b	2D Conv	ReLU3c	3×3	1×1	$28 \times 28 \times 128$
ReLU4b	ReLU	Conv4b	-	-	$28 \times 28 \times 128$
Conv4c	2D Conv	MaxPool3	3×3	1×1	$56 \times 56 \times 128$
ReLU4c	ReLU	Conv4c	-	-	$56 \times 56 \times 128$
Conv4d	2D Conv	P^j	3×3	1×1	$112 \times 112 \times 32$
ReLU4d	ReLU	Conv4d	-	-	$112 \times 112 \times 32$

Table 2: Network details of $\mathcal{E}_a$, which was based upon [6]. The above table contains all layers of the encoder and four additional layers to transform the featuremap to maintain the number of channels. The row of Input indicates where the input of this layer comes from. Since the proposed method involves Multi-Scale Learning framework, there are four outputs from this network: ReLU4a, ReLU4b, ReLU4c and ReLU4d.

Name	Layer	Input	Kernel Dims (H $\times$ W)	Strides (H $\times$ W)	Output Dims (H $\times$ W $\times$ C)
Conv1	2D Conv	x_{a1}	3×3	4×4	$14 \times 14 \times 25$
Conv2	2D Conv	x_{a2}	3×3	2×2	$14 \times 14 \times 25$
Conv2	2D Conv	x_{a3}	3×3	1×1	$14 \times 14 \times 25$
Final1	2D Conv	Conv1, Conv2, Conv3	3×3	1×1	$14 \times 14 \times 50$
Final2	2D Conv	Final1	3×3	1×1	$14 \times 14 \times 25$
Final3	SoftmaxMean	Final2	-	-	25×2

Table 3: Network details of $\mathcal{P}_E$. It contains three convolutional layers to transform the input featuremaps to similar spatial size and three additional layers to predict the pose. Notice that, the final3 layer calculates the softmax of the last two dimensions of the input to obtain the probability vector. The output of this network would be number of joints with 2D coordinates.

sue is due to the fact that the interaction between frames resulted from 3D convolution. But compared to frame-by-frame generation network [5, 8], we are more efficient and resource-saving. We believe that designing a more innovative decoder is the key to solve this problem.

References

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

Name	Layer	Input	Kernel Dims (T × H × W)	Strides (T × H × W)	Output Dims (T × H × W × C)
Conv1a	3D Conv	$\mathcal{E}_a$final(1) + $\mathcal{P}_T$	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 14 \times 14 \times 128$
ReLU1a	ReLU	Conv1a	-	-	$16 \times 14 \times 14 \times 128$
Conv1b	3D Conv	ReLU1a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 14 \times 14 \times 128$
ReLU1b	ReLU	Conv1b	-	-	$16 \times 14 \times 14 \times 128$
Inter1	Interpolate	ReLU1b	-	-	$16 \times 28 \times 28 \times 128$
Conv2a	3D Conv	$\mathcal{E}_a$final(2) + $\mathcal{P}_T$ + Inter1	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 28 \times 28 \times 128$
ReLU2a	ReLU	Conv2a	-	-	$16 \times 28 \times 28 \times 128$
Conv2b	3D Conv	ReLU2a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 28 \times 28 \times 64$
ReLU2b	ReLU	Conv2b	-	-	$16 \times 28 \times 28 \times 64$
Inter2	Interpolate	ReLU2b	-	-	$16 \times 56 \times 56 \times 64$
Conv3a	3D Conv	$\mathcal{E}_a$final(3) + $\mathcal{P}_T$ + Inter2	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 56 \times 56 \times 128$
ReLU3a	ReLU	Conv3a	-	-	$16 \times 56 \times 56 \times 128$
Conv3b	3D Conv	ReLU3a	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 56 \times 56 \times 32$
ReLU3b	ReLU	Conv3b	-	-	$16 \times 56 \times 56 \times 32$
Inter3	Interpolate	ReLU3b	-	-	$16 \times 112 \times 112 \times 32$
Conv4a	3D Conv	Inter3 + $\mathcal{E}_a$final(4) + $\mathcal{P}_T$	$3 \times 3 \times 3$	$1 \times 1 \times 1$	$16 \times 112 \times 112 \times 8$
ReLU4a	ReLU	Conv4a	-	-	$16 \times 112 \times 112 \times 8$
Conv4b	3D Conv	ReLU4a	$1 \times 1 \times 1$	$1 \times 1 \times 1$	$16 \times 112 \times 112 \times 3$
Sig	Sigmoid	Conv4b	-	-	$16 \times 112 \times 112 \times 3$

Table 4: Network details for the Video Decoder, $\mathcal{D}_{\mathcal{V}}$, which generates the final output video v_t based upon the three sets of transformed appearance features and the Multi-scale attention $\mathcal{M}_{\mathcal{A}}$. Note that hierarchical generation is used, so the larger appearance features are concatenated as input where appropriate. The final output has the same dimensions as the input video V^i.

[2] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[3] M. Lakhal, O. Lanz, and A. Cavallaro. View-lstm: Novel-view video synthesis through view decomposition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7576–7586, 2019.

[4] Kara Marie Schatz, Erik Quintanilla, Shruti Vyas, and Y. Rawat. A recurrent transformer network for novel view action synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[5] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7137–7147. Curran Associates, Inc., 2019.

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

[7] Shruti Vyas, Yogesh Singh Rawat, and Mubarak Shah. Time-aware and view-aware video rendering for unsupervised representation learning. *CoRR*, abs/1811.10699, 2018.

[8] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Name	Layer	Input	Kernel Dims (H × W)	Strides (H × W)	Output Dims (H × W × C)
STN-Conv1	1D Conv	p_t1	3	1	25×2
STN-Conv2	1D Conv	p_t2	3	1	25×2
STN-Conv3	1D Conv	STN-Conv1& STN-Conv1	3	1	25×2
STN-Linear1	Linear	STN-Maxpool2	1	-	32
STN-Linear2	Linear	STN-Linear1	1	-	6
Pose-crop	-	$\mathcal{E}_a$-Conv3(1)	-	-	scale × scale
Affine_Trans	Grid_sampler	STN-Linear2 +Pose-crop	-	-	$14 \times 14 \times 128$
GTN-Conv1	2D Conv	Affine_Trans	Grid_sampler		
		+ p_t-Gaussian	7×7	1×1	$14 \times 14 \times 256$
GTN-Split	Split	GTN-Conv1	-	-	$14 \times 14 \times 128$
					$14 \times 14 \times 128$
GTN-Sig1	Sigmoid	GTN-Split(1)	-	-	$14 \times 14 \times 128$
GTN-Sig2	Sigmoid	GTN-Split(2)	-	-	$14 \times 14 \times 128$
GTN-Conv2	2D Conv	$\mathcal{E}_a$-Conv3(1)			
		+ p_t-Gaussian	7×7	1×1	$14 \times 14 \times 128$
GTN-Tanh	Tanh	GTN-Conv2	-	-	$14 \times 14 \times 128$
GTN-Final	Concat	(1 - GTN-Sig2) * p_t-Gaussian			
		+ GTN-Sig2 * GTN-Tanh	-	-	$14 \times 14 \times 128$

Table 5: Network details of the $\mathcal{LGTN}$. It is based on the Spatial transformation network [2], whose output would be the predicted affine matrix. We adopt the key-region separator as discussed in main paper to crop the appearance featuremap. Then, we use grid sampler to transform the feature. Then we use a GRU[1] global transformation on the output of local-transformed feature. At last, we add this transformed foreground feature back to the background generated by the $\mathcal{P}_{crop}$.

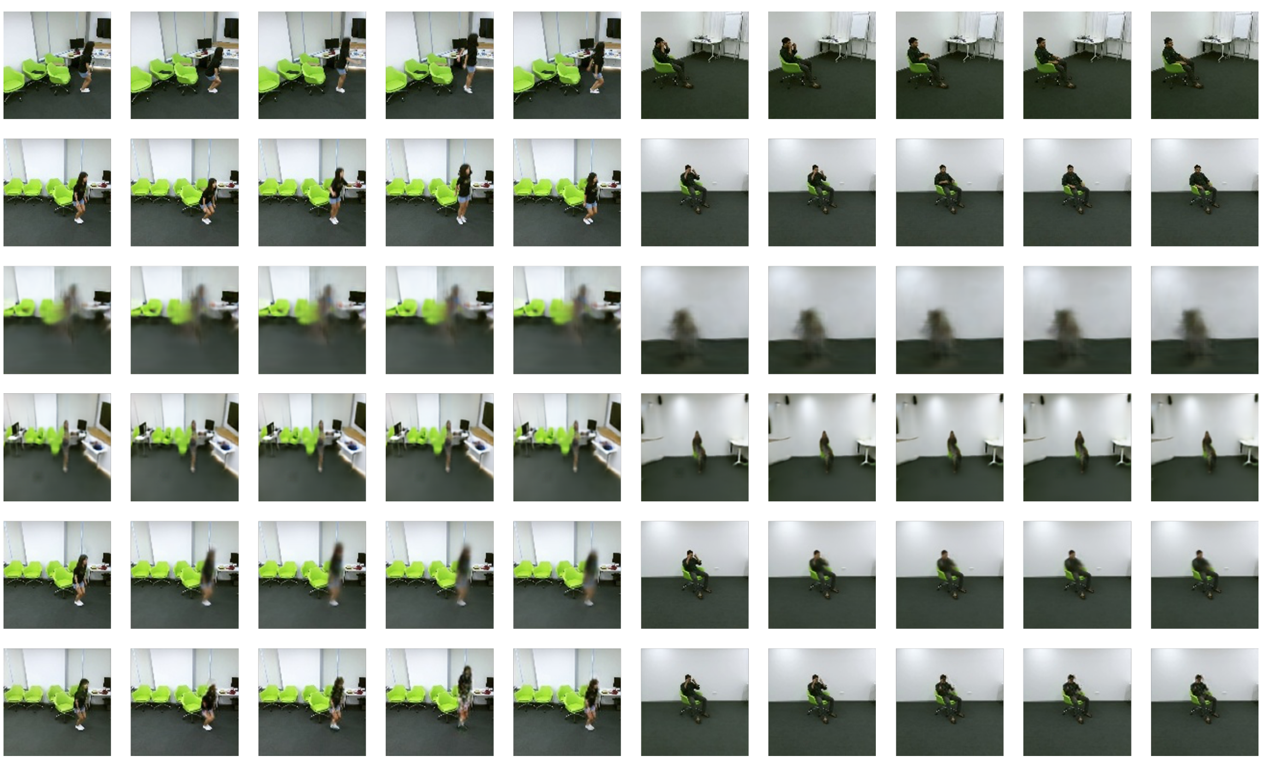

Figure 1: Comparison of the generated frames between our proposed and existing methods. Row 1: source, row 2: target, row 3: VDNet [3], row 4: VRNet [7], row 5: BasicNet, row 6: RTNet [4], row 7: proposed method. We can observe that that RTNet [4] generates good quality frames, but it lacks action dynamics.

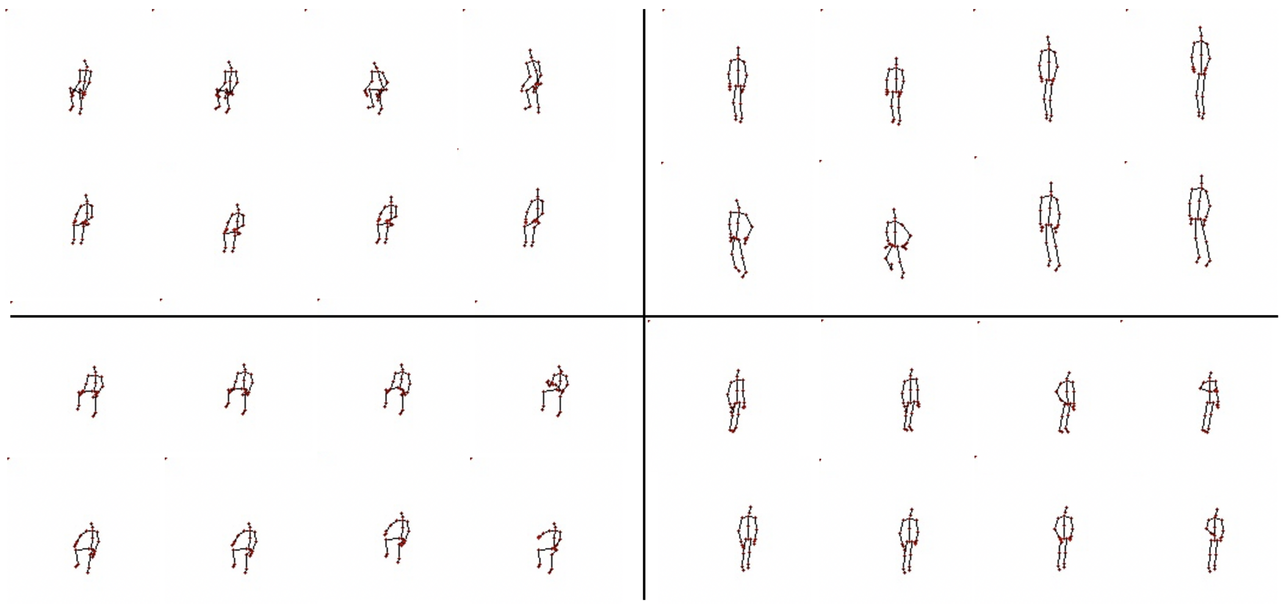

Figure 2: Generated pose results. Each corner represents one sample. In every corner, the first row is the target and the second row is the generated results. We sample 4 frames (1, 3, 5 and 7) from original eight generated frames.

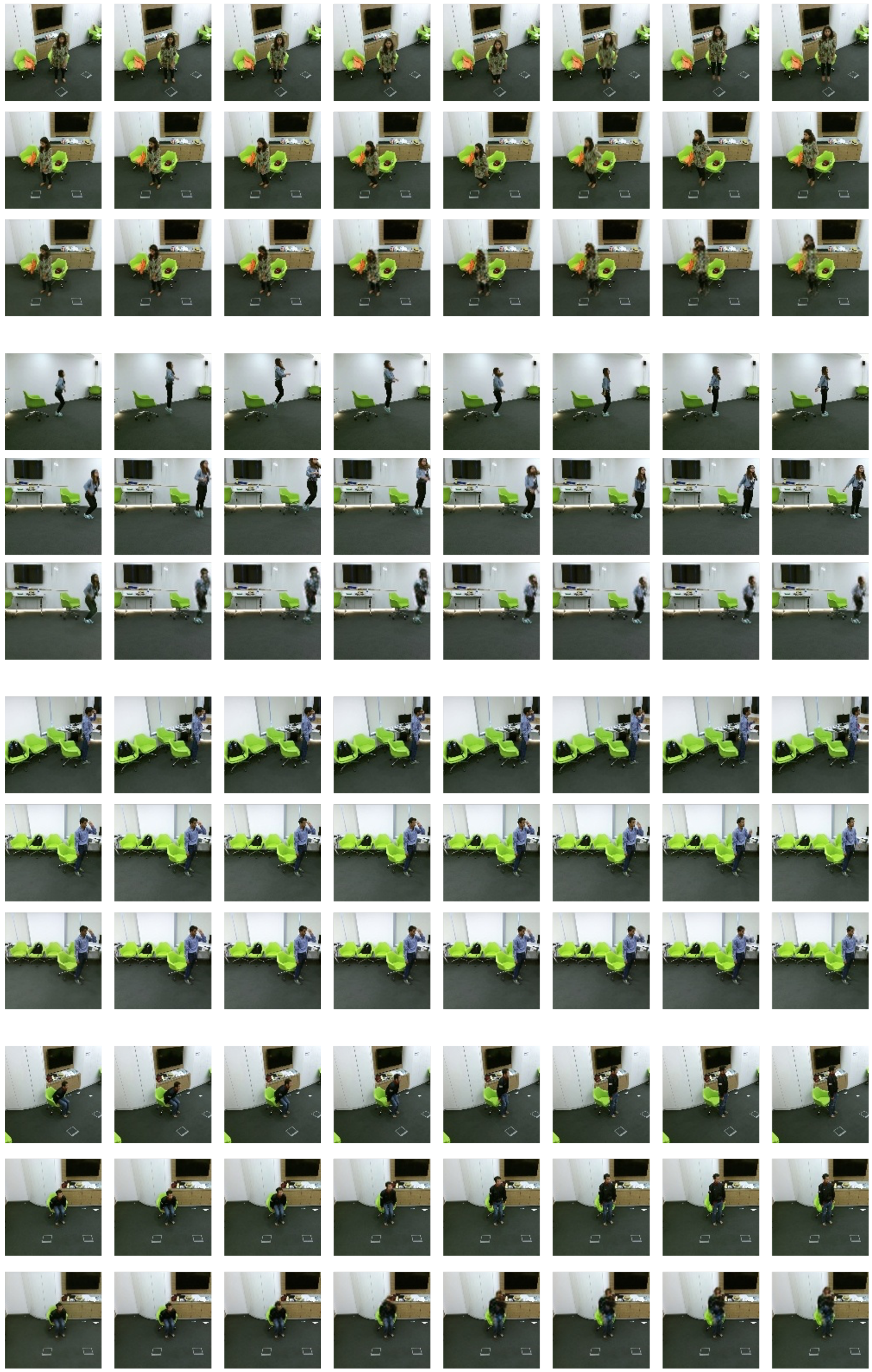

Figure 3: More qualitative results. Every three rows represent a video sample. In each sample, first row: the source video; second row: the target video; third row: the generated video.

Figure 4: More qualitative results. Every three rows represent a video sample. We sample 8 frames. Each column represents one frame. In each sample, first row: source video. The second row: target video. The third row: generated video.