Supplementary material for "Neural Puppet: Generative Layered Cartoon Characters"

Omid Poursaeed[1,2] Vladimir G. Kim[3] Eli Shechtman[3] Jun Saito[3] Serge Belongie[1,2]

[1]Cornell University [2]Cornell Tech [3]Adobe Research

We provide results for new characters and additional samples of existing characters in Figure A. The first character is from Adobe Character Animator [1], and the second one is from Sykora *et al.* [4]. We use 33/10 and 22/8 train/test samples respectively. As we observe, our method clearly outperforms the baselines in all cases. We also show generated images for the character in the wild (corresponding to Section 4.4. and Figure 7 in the main paper). In this case, we feed masked input images to the discriminator of the conditional GAN. Figure B illustrates additional in-betweening results, and corresponds to Section 4.1. and Figure 5 in the main paper. Figure C depicts character deformations based on various constraints (corresponding to Section 4.2. and Figure 6 in the main paper). We also provide videos showing smooth interpolations and deformations of the characters. *Interpolation.mp4* shows examples of in-betweening for various poses. *Deformation.mp4* illustrates user-constrained deformations corresponding to Figure 6 of the paper, and compares our model with ARAP. In our method, we first generate the z-values using a uniform step size as in equation (9) in the paper. In order to make the deformation look smoother, we then re-sample the z values uniformly. Let z_{init} and z_{final} denote the initial and final z values. The i-th re-sampled z is then:

$$z_i = z_{\text{init}} + (z_{\text{final}} - z_{\text{init}}) * \frac{i}{N} \tag{1}$$

in which $N = 1000$ represents the number of steps. z_i is then passed to the decoder and the renderer to obtain the images. As illustrated in Figure C and the video, our approach achieves global shape consistency, while optimizing directly on vertex positions (as in ARAP) only preserves local rigidity. Note that since we predict the vertex positions, we can render images with arbitrary resolutions. We show 256×256 images in the videos. Using a better renderer can further improve the results. In Figure D we show full-resolution output of our method as 1024×1024 images using vanilla OpenGL renderer.

References

[1] Adobe. Adobe character animator, 2019. 1

[2] Z. Shu, M. Sahasrabudhe, A. Guler, D. Samaras, N. Paragios, and I. Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. *arXiv preprint arXiv:1806.06503*, 2018. 2, 3

[3] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 2, 3

[4] D. Sỳkora, M. Ben-Chen, M. Čadík, B. Whited, and M. Simmons. Textoons: practical texture mapping for hand-drawn cartoon animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, pages 75–84. ACM, 2011. 1

Figure A: Input images, our rendered and final results, followed by results obtained with PWC-Net [3] and DAE [2].

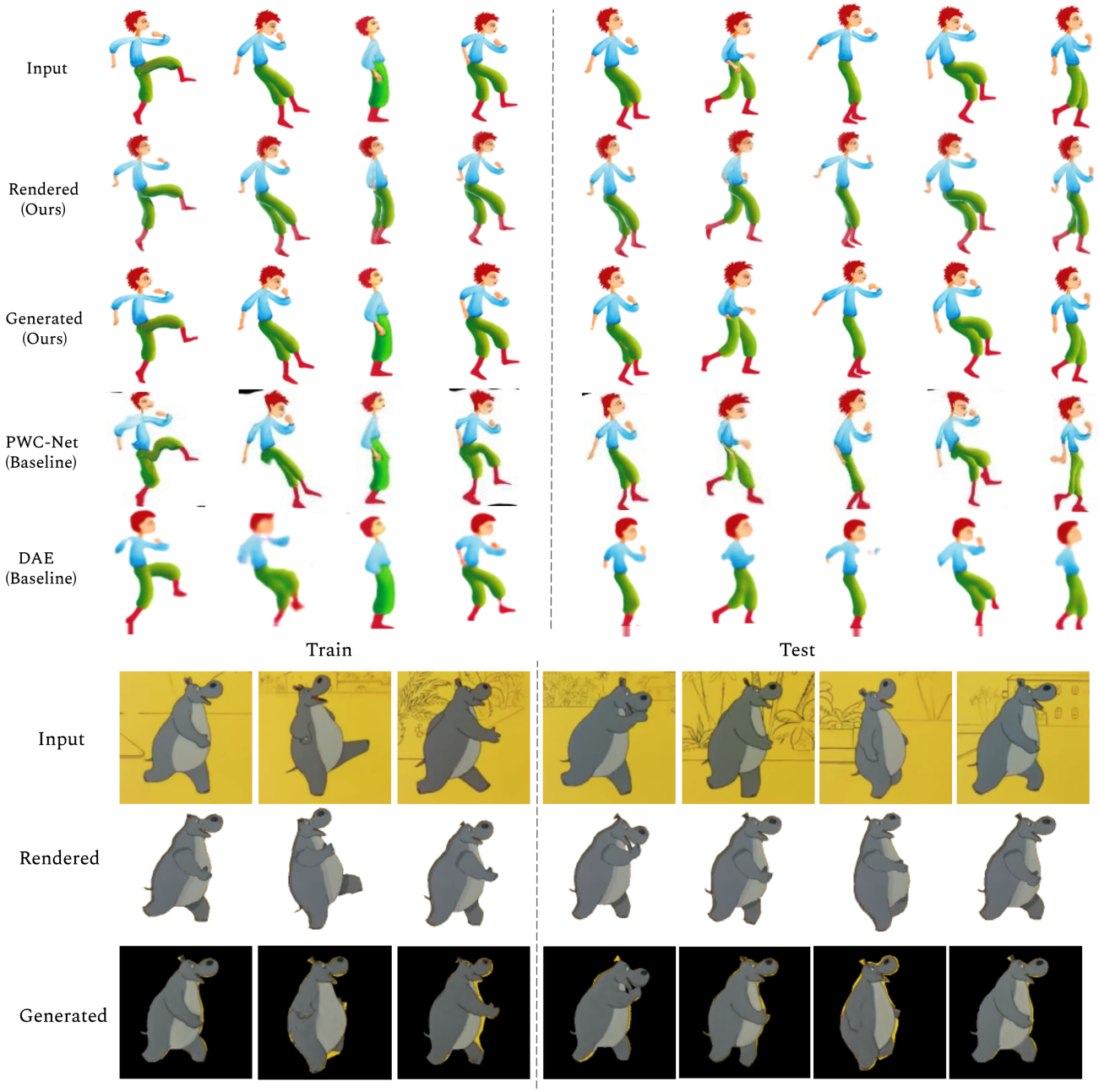

Figure A: (cont.) Input images, our rendered and final results, followed by results obtained with PWC-Net [3] and DAE [2].

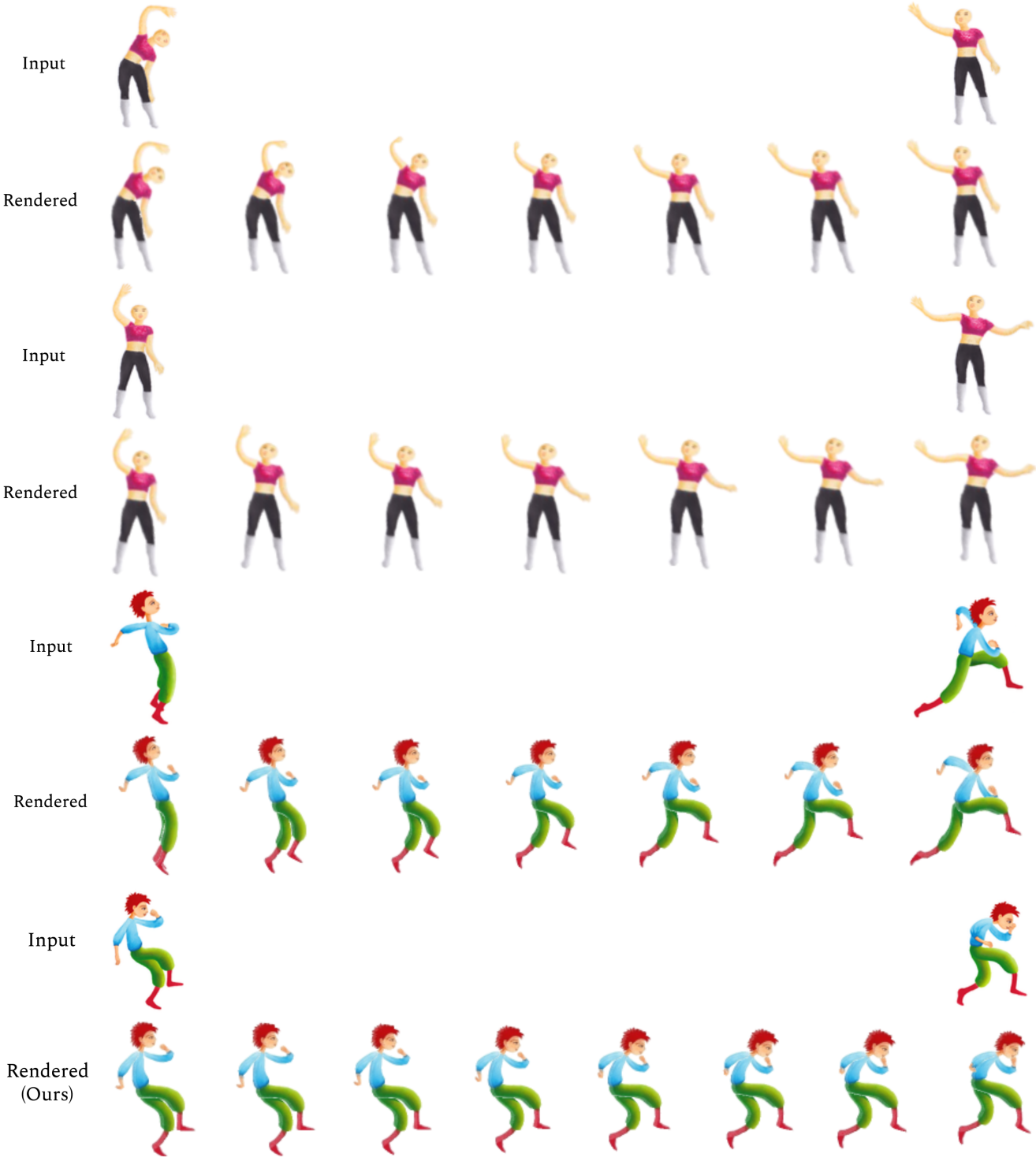

Figure B: Inbetweening results. Two given test images (first row) are encoded. The resulting latent vectors are linearly interpolated yielding the resulting rendered images.

Figure C: User-constrained deformation. Given the starting vertex and the desired location (shown with the arrow), the model learns a plausible deformation to satisfy the user constraint.

Figure D: Output of our method rendered into 1024×1024 images.