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

Abstract

We propose a learning based method for generating new animations of a cartoon character given a few example images. Our method is designed to learn from a traditionally animated sequence, where each frame is drawn by an artist, and thus the input images lack any common structure, correspondences, or labels. We express pose changes as a deformation of a layered 2.5D template mesh, and devise a novel architecture that learns to predict mesh deformations matching the template to a target image. This enables us to extract a common low-dimensional structure from a diverse set of character poses. We combine recent advances in differentiable rendering as well as mesh-aware models to successfully align common template even if only a few character images are available during training. In addition to coarse poses, character appearance also varies due to shading, out-of-plane motions, and artistic effects. We capture these subtle changes by applying an image translation network to refine the mesh rendering, providing an end-to-end model to generate new animations of a character with high visual quality. We demonstrate that our generative model can be used to synthesize in-between frames and to create data-driven deformation. Our template fitting procedure outperforms state-of-the-art generic techniques for detecting image correspondences.

1. Introduction

Traditional character animation is a tedious process that requires artists meticulously drawing every frame of a motion sequence. After observing a few such sequences, a human can easily imagine what a character might look in other poses, however, making these inferences is difficult for learning algorithms. The main challenge is that the input images commonly exhibit substantial variations in appearance due to articulations, artistic effects, and viewpoint changes, significantly complicating the extraction of the underlying character structure. In the space of natural images, one can rely on extensive annotations [48] or vast amount of data [52] to extract the common structure. Unfortunately, this approach is not feasible for cartoon characters, since their topology, geometry, and drawing style is far less consistent than that of natural images of human bodies or faces.

To tackle this challenge, we propose a method that learns to generate novel character appearances from a small number of examples by relying on additional user input: a deformable puppet template. We assume that all character poses can be generated by warping the deformable template, and thus develop a deformation network that encodes an image and decodes deformation parameters of the template. These parameters are further used in a differentiable rendering layer that is expected to render an image that matches the input frame. Reconstruction loss can be back-propagated through all stages, enabling us to learn how to register the template with all of the training frames. While the resulting renderings already yield plausible poses, they fall short of artist-generated images since they only warp a single reference, and do not capture slight appearance variations due to shading and artistic effects. To further improve visual quality of our results, we use an image translation network that synthesizes the final appearance.

While our method is not constrained to a particular choice for the deformable puppet model, we chose a layered 2.5D deformable model that is commonly used in academic [15] and industrial [2] applications. This model matches many traditional hand-drawn animation styles, and makes it significantly easier for the user to produce the template relative to 3D modeling that requires extensive expertise. To generate the puppet, the user has to select a single frame and segment the foreground character into constituent body parts, which can be further converted into meshes using standard triangulation tools [50].

We evaluate our method on animations of six characters with 70%-30% train-test split. First, we evaluate how well our model can reconstruct the input frame and demonstrate that it produces more accurate results than state-of-the-art optical flow and auto-encoder techniques. Second, we evaluate the quality of correspondences estimated via the registered templates, and demonstrate improvement over image correspondence methods. Finally, we show that our model can be used for data-driven animation, where synthesized

animation frames are guided by character appearances observed at training time. We build prototype applications for synthesizing in-between frames and animating by user-guided deformation where our model constrains new images to lie on a learned manifold of plausible character deformations. We show that the data-driven approach yields more realistic poses that better match to original artist drawings than traditional energy-based optimization techniques used in computer graphics.

2. Related Work

Deep Generative Models. Several successful paradigms of deep generative models have emerged recently, including the auto-regressive models [21, 42, 59], Variational Auto-encoders (VAEs) [33, 32, 47], and Generative Adversarial Networks (GANs) [20, 44, 49, 26, 6]. Deep generative models have been applied to image-to-image translation [27, 66, 65], image superresolution [36], learning from synthetic data [10, 51], generating adversarial images [43], and synthesizing 3D volumes [58, 53]. These techniques usually make no assumptions about the structure of the training data and synthesize pixels (or voxels) directly. This makes them very versatile and appealing when a large number of examples are available. Since these data might not be available in some domains, such as 3D modeling or character animation, several techniques leverage additional structural priors to train deep models with less training data.

Learning to Generate with Deformable Templates. Deformable templates have been used for decades to address analysis and synthesis problems [5, 9, 3, 4, 40, 67]. Synthesis algorithms typically assume that multiple deformations of the same template (e.g., a mesh of the same character in various poses) is provided during training. Generative models, such as variational auto-encoders directly operate on vertex coordinates to encode and generate plausible deformations from these examples [57, 35, 38]. Alternative representations, such as multi-resolution meshes [45], single-chart UV [7], or multi-chart UV [24] is used for higher resolution meshes. This approaches are limited to cases when all of the template parameters are known for all training examples, and thus cannot be trained or make inferences over raw unlabeled data.

Some recent work suggests that neural networks can jointly learn the template parameterization and optimize for the alignment between the template and a 3D shape [22] or 2D images [29, 25, 60]. While these models can make inferences over unlabeled data, they are trained on a large number of examples with rich labels, such as dense or sparse correspondences defined for all pairs of training examples.

To address this limitation, a recent work on deforming auto-encoders provides an approach for unsupervised group-wise image alignment of related images (e.g. human faces) [52]. They disentangle shape and appearance in latent space, by predicting a warp of a learned template image as well as its texture transformations to match every target image. Since their warp is defined over the regular 2D lattice, their method is not suitable for strong articulations. Thus, to handle complex articulated characters and strong motions, we leverage an user-provided template and rely on regularization terms that leverage the rigging as well as mesh structure. Instead of synthesizing appearance in a common texture space, we do the final image translation pass that enables us to recover from warping artifacts and capture effects beyond texture variations, such as out-of-plane motions.

Mesh-based models for Character Animation. Many techniques have been developed to simplify the production of traditional hand-drawn animation using computers [12, 16]. Mesh deformation techniques, enable novice users to create animations by manipulating a small set of control points of a single puppet [54, 28]. To avoid overly-synthetic appearance, one can further stylize these deformations by leveraging multiple co-registered examples to guide the deformation [61], and final appearance synthesis [17, 18]. These methods, however, require artist to provide the input in a particular format, and if this input is not available rely on image-based correspondence techniques [11, 56, 14, 19, 55] to register the input. Our deformable puppet model relies on a layered mesh representation [18] and mesh regularization terms [54] used in these optimization-based workflows. Our method jointly learns the puppet deformation model as it registers the input data to the template, and thus yields more accurate correspondences than state-of-the-art flow-based approach [55] and state-of-the-art feature-based method trained on natural images [14].

3. Approach

Our goal is to learn a deformable model for a cartoon character given an unlabeled collection of images. First, the user creates a layered deformable template puppet by segmenting one reference frame. We then train a two-stage neural network that first fits this deformable puppet to every frame of the input sequence by learning how to warp a puppet to reconstruct the appearance of the input, and second, it refines the rendering of deformed puppet to account for texture variations and motions that cannot be expressed with a 2D warp.

3.1. A Layered Deformable Puppet

The puppet geometry is represented with a few triangular meshes ordered into layers and connected at hinge joints. For simplicity, we denote them as one mesh with vertices

(a) (b)

Figure 1: Deformable Puppet. a) For each body part a separate mesh is created, and joints (shown with circles) are specified. b) The meshes are combined into a single mesh. The UV-image of the final mesh consists of translated versions of separate texture maps.

V and faces F, where every layer is a separate connected component of the mesh. Joints are represented as $\{(p_i, q_i)\}$ pairs, connecting vertices between some of these components. The puppet appearance is captured as texture image I^{uv}, which aligns to some rest pose $\hat{V}$. New character poses can be generated by modifying vertex coordinates, and the final appearance can be synthesized by warping the original texture according to mesh deformation.

Unlike 3D modeling, even inexperienced users can create the layered 2D puppet. First, one selects a reference frame and provides the outline for different body parts and prescribes the part ordering. We then use standard triangulation algorithm [50] to generate a mesh for each part, and create a hinge joint at the centroid of overlapping area of two parts. We can further run midpoint mesh subdivision to get a finer mesh that can model more subtle deformations. Figure 1 illustrates a deformable puppet.

3.2. Deformation Network

After obtaining the template, we aim to learn how to deform it to match a target image of the character in a new pose. Figure 2 illustrates our architecture. The inputs to the Deformation Network are the initial mesh and a target image of the character in a new pose. An encoder-decoder network takes the target image, encodes it to a bottleneck via convolutional filters, and then decodes it to vertex position offsets via fully connected layers. Thus, it learns to recognize the pose in the input image and then infer appropriate template deformation to reproduce the pose. We assume the connectivity of vertices and the textures remain the same compared to the template. Hence, we pass the faces and textures of the initial mesh in tandem with the predicted vertex positions to a differentiable renderer R. The rendered image is then compared to the input image using L_2 reconstruction loss:

$$L_{rec} = \|x - R(V_{pred}, F, I^{uv})\|^2 \quad (1)$$

in which x represents the input image. We use the Neural Mesh Renderer [30] as our differentiable renderer, since it can be easily integrated into neural network architectures.

Regularization. The model trained with only the reconstruction loss does not preserve the structure of the initial mesh, and the network may generate large deformations to favor local consistency. In order to prevent this, we use the ARAP regularization energy [54] which penalizes deviations of per-cell transformations from rigidity:

$$L_{reg} = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}_i} w_{ij} \|(\hat{v}_i - \hat{v}_j) - R_i(v_i - v_j)\|^2 \quad (2)$$

in which v_i and $\hat{v}_i$ are coordinates of vertex i before and after deformation, $\mathcal{N}_i$ denotes neighboring vertices of vertex i, w_{ij} are cotangent weights and R_i is the optimal rotation matrix as discussed in [54].

Joints Loss. If we do not constrain vertices of the layered mesh, different limbs can move away from each other, resulting in unrealistic outputs. In order to prevent this, we specify 'joints' $(p_i, q_i), i = 1, \ldots, n$ as pairs of vertices in different layers that must remain close to each other after deformation. We manually specify the joints, and penalize the sum of distances between vertices in each joint:

$$L_{joints} = \sum_{i=1}^{n} \|p_i - q_i\|^2 \quad (3)$$

Our final loss for training the Deformation Network is a linear combination of the aforementioned losses:

$$L_{total} = L_{rec} + \lambda_1 \cdot L_{reg} + \lambda_2 \cdot L_{joints} \quad (4)$$

We use $\lambda_1 = 2500$ and $\lambda_2 = 10^4$ in the experiments.

3.3. Appearance Refinement Network

While articulations can be mostly captured by deformation network, some appearance variations such as artistic stylizations, shading effects, and out-of-plane motions cannot be generated by warping a single reference. To address this limitation, we propose an appearance refinement network that processes the image produced by rendering the deformed puppet. Our architecture and training procedure is similar to conditional Generative Adversarial Network (cGAN) approach that is widely used in various domains [41, 64, 27]. The corresponding architecture is shown in Figure 2. The generator refines the rendered image to look more natural and more similar to the input image. The discriminator tries to distinguish between input frames of character poses and generated images. These two networks are then trained in an adversarial setting [20], where we use pix2pix architecture [27] and Wasserstein GAN with Gradient Penalty for adversarial loss [6, 23]:

$$L_G = -\mathbb{E}\big[D(G(x_{\text{rend}}))\big] + \gamma_1 \|G(x_{\text{rend}}) - x_{\text{input}}\|_1 \quad (5)$$

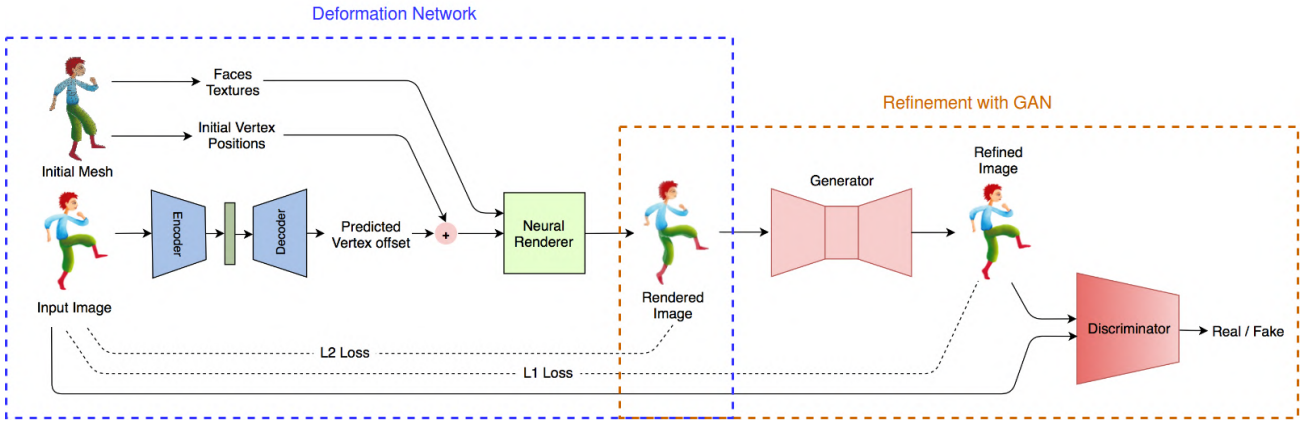

Figure 2: Training Architecture. An encoder-decoder network learns the mesh deformation and a conditional Generative Adversarial Network refines the rendered image to capture texture variations.

And the discriminator's loss is:

$$L_D = \mathbb{E}\big[D(G(x_{\text{rend}}))\big] - \mathbb{E}\big[D(x_{\text{real}})\big] + \gamma_2\, \mathbb{E}\big[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\big] \quad (6)$$

in which $D(\cdot)$ and $G(\cdot)$ are the discriminator and the generator, $\gamma_1, \gamma_2 \in \mathbb{R}$ are weights, x_{input} and x_{rend} are the input and rendered images, x_{real} is an image sampled from the training set, and $\hat{x} = \epsilon\, G(x_{\text{rend}}) + (1-\epsilon)\, x_{\text{real}}$ with ϵ uniformly sampled from the [0, 1] range. The cGAN is trained independently after training the Deformation Network as this results in more stable training.

4. Results and Applications

We evaluate how well our method captures (i.e., encodes and reconstructs) character appearance. We use six character animation sequences from various public sources. Figure 3 shows some qualitative results where for each input image we demonstrate output of the deformation network (*rendered*) and the final synthesized appearance (*generated*). The first three characters are from Dvoroznak et al. [18] with 1280/547, 230/92 and 60/23 train/test images respectively. The last character (robot) is obtained from Adobe Character Animator [2] with 22/9 train/test images. Other characters and their details are given in the supplementary material. The *rendered* result is produced by warping a reference puppet, and thus it has fewer degrees of freedom (e.g., it cannot change texture or capture out-of-plane motions). However, it still provides fairly accurate reconstruction even for very strong motions, suggesting that our layered puppet model makes it easier to account for significant character articulations, and that our image encoding can successfully predict these articulations even though it was trained without strong supervision. Our refinement network does not have any information from the original image other than the re-rendered pose, but, evidently, adversarial loss provides sufficient guidance to improve the final appearance and match the artist-drawn input. To confirm that these observations hold over all characters and frames, we report L_2 distance between the target and generated images in Table 1. See supplemental material for more examples.

We compare our method to alternative techniques for re-synthesizing novel frames (also in Figure 3). One can use optical flow method, such as PWC-Net [55] to predict a warping of a reference frame (we use the frame that was used to create the puppet) to the target image. This method was trained on real motions in videos of natural scenes and tends to introduce strong distortion artifacts when matching large articulations in our cartoon data. Various autoencoder approaches can also be used to encode and reconstruct appearance. We compare to a state-of-the-art approach that uses deforming auto-encoders [52] to disentangle deformation and appearance variations by separately predicting a warp field and a texture. This approach does not decode character-specific structure (the warp field is defined over a regular grid), and thus also tends to fail at larger articulations. Another limitation is that it controls appearance by altering a pre-warped texture, and thus cannot correct for any distortion artifacts introduced by the warp. Both methods have larger reconstruction loss in comparison to our rendered as well as final results (see Table 1).

One of the key advantages of our method is that it predicts deformation parameters, and thus can retain resolution of the artist-drawn image. To illustrate this, we render the output of our method as 1024×1024 images using vanilla OpenGL renderer in Figure 4. We show full-size images in the supplementary material. The final conditional generation step can also be trained on high-resolution images.

4.1. Inbetweening

In traditional animation, a highly-skilled artist creates a few sparse keyframes and then in-betweening artists draw the other frames to create the entire sequence. Various com-

Figure 3: Input images, our rendered and final results, followed by results obtained with PWC-Net [55] and DAE [52] (input images for the first three characters are drawn by ©Zuzana Studená. The fourth character ©Adobe Character Animator).

putational tools have been proposed to automate the second step [34, 56, 8, 62], but these methods typically use hand-crafted energies to ensure that intermediate frames look plausible, and rely on the input data to be provided in a particular format (e.g., deformations of the same puppet). Our method can be directly used to interpolate between two raw images, and our interpolation falls on the manifold of deformations learned from training data, thus generating in-

	Char1	Char2	Char3	Char4	Avg
Rendered	819.8	732.7	764.1	738.9	776.1
Generated	**710.0**	**670.5**	**691.7**	**659.2**	**695.3**
PWC-Net	1030.4	1016.1	918.3	734.6	937.1
DAE	1038.3	1007.2	974.8	795.1	981.6

Table 1: Average L_2 distance to the target images from the test set. Rendered and generated images from our method are compared with PWC-Net [55] and Deforming Autoencoders [52]. The last column shows the average distance across six different characters.

Figure 4: Output of our method rendered into 1024×1024 images (zoom in for details).

betweens that look similar to the input sequence.

Given two images x_1 and x_2 we use the encoder in deformation network to obtain their features, $z_i = E(x_i)$. We then linearly interpolate between z_1 and z_2 with uniform step size, and for each in-between feature z we apply the rest of our network to synthesize the final appearance. The resulting interpolations are shown in Figure 5. The output images smoothly interpolate the motion, while mimicking poses observed in training data. This suggests that the learned manifold is smooth and can be used directly for example-driven animation. We further confirm that our method generalizes beyond training data by showing nearest training neighbor to the generated image (using Euclidean distance as the metric).

4.2. User-constrained Deformation

Animations created with software assistance commonly rely on deforming a puppet template to target poses. These deformations are typically defined by local optima with respect to user-prescribed constraints (i.e., target motions) and some hand-crafted energies such as rigidity or elasticity [54, 37, 13]. This is equivalent to deciding on what kind of physical material the character is made of (e.g., rubber, paper), and then trying to mimic various deformations of that material without accounting for artistic stylizations and bio-mechanical priors used by professional animators. While some approaches allow transferring these effects from stylized animations [18], they require artist to provide consistently-segmented and densely annotated frames aligned to some reference skeleton motion. Our model does not rely on any annotations and we can directly use our learned manifold to find appropriate deformations that satisfy user constraints.

Given the input image of a character x, the user clicks on any number of control points $\{p_i\}$ and prescribes their desired target positions $\{p_i^{\text{trg}}\}$. Our system then produces the image x' that satisfies the user constraints, while adhering to the learned manifold of plausible deformations. First, we use the deformation network to estimate vertex parameters to match our puppet to the input image: $\mathbf{v} = D(E(x))$ (where $E(\cdot)$ is the encoder and $D(\cdot)$ is the decoder in Figure 2). We observe that each user-selected control point p_i can now be found on the surface of the puppet mesh. One can express its position as a linear combination of mesh vertices, $p_i(\mathbf{v})$, where we use barycentric coordinates of the triangle that encloses p_i. The user constrained can be expressed as an energy, penalizing distance to the target:

$$L_{user} = \sum_i \left\| p_i(\mathbf{v}) - p_i^{\text{trg}} \right\|^2 \tag{7}$$

For the deformation to look plausible, we also include the regularization terms:

$$L_{deform} = L_{user} + \alpha_1 \cdot L_{reg} + \alpha_2 \cdot L_{joints} \tag{8}$$

We use $\alpha_1 = 25$, $\alpha_2 = 50$ in the experiments.

Since our entire deformation network is differentiable, we can propagate the gradient of this loss function to the embedding of the original image $z_0 = E(x)$ and use gradient descent to find z that optimizes L_{deform}:

$$z \longleftarrow z - \eta \nabla_z L_{deform} \tag{9}$$

where $\eta = 3 \times 10^{-4}$ is the step size. In practice, a few (2 to 5) iterations of gradient descent suffice to obtain satisfactory results, enabling fast, interactive manipulation of the mesh by the user (in the order of seconds).

The resulting latent vector is then passed to the decoder, the renderer and the refinement network. Figure 6 (left) illustrates the results of this user-constrained deformation. As we observe, deformations look plausible and satisfy the user constraints. They also show global consistency; for instance, as we move one of the legs to satisfy the user constraint, the torso and the other leg also move in a consistent manner. This is due to the fact that the latent space encodes high-level information about the character's poses, and it learns that specific poses of torso are likely to co-occur with specific poses of legs, as defined by the animator.

We compare our method with optimizing directly in the vertex space using the regularization terms only (Figure 6, right). This approach does not use the latent representation,

Figure 5: Inbetweening results. Two given images (first row) are encoded. The resulting latent vectors are linearly interpolated yielding the rendered and generated (final) images. For each generated image, the corresponding Nearest Neighbor image from the training set is retrieved.

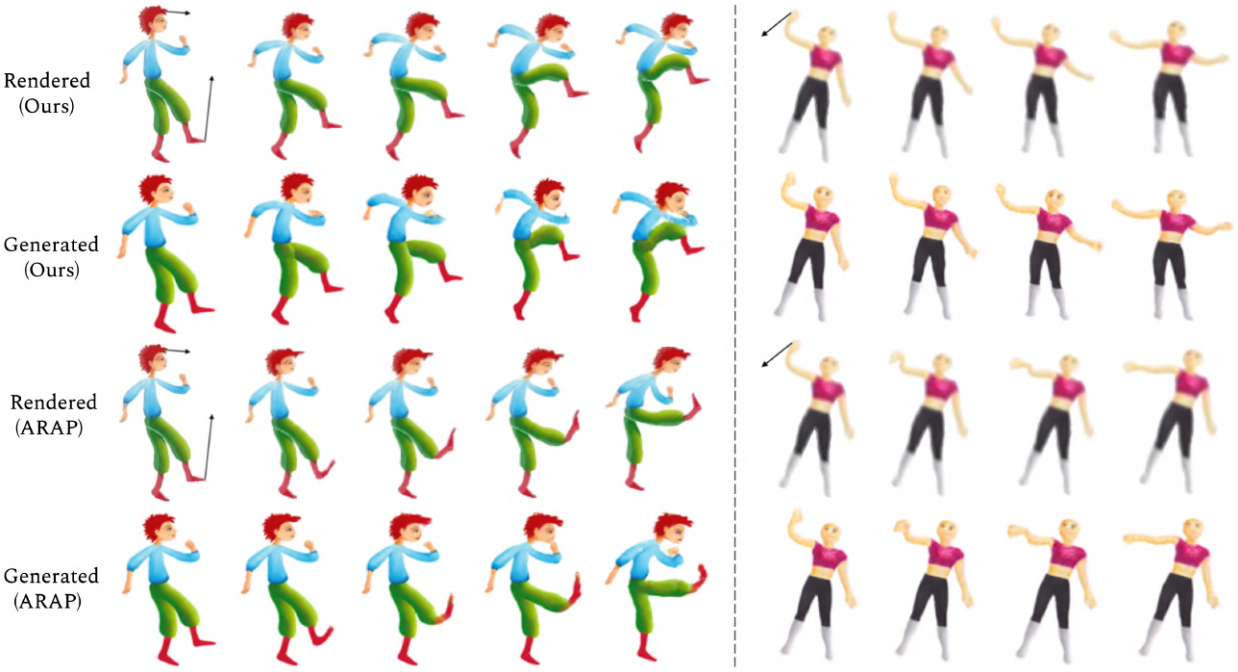

Figure 6: User-constrained deformation. Given the starting vertex and the desired location (shown with the arrow), the model learns a plausible deformation to satisfy the user constraint. Our approach of searching for an optimal latent vector achieves global shape consistency, while optimizing directly on vertex positions only preserves local rigidity.

and thus does not leverage the training data. It is similar to traditional energy-based approaches, where better energy models might yield smoother deformation, but would not enforce long-range relation between leg and torso motions.

4.3. Correspondence Estimation

Many video editing applications require inter-frame correspondence. While many algorithms have been proposed to address this goal for natural videos [14, 39, 31, 63, 46],

Figure 7: Characters in the wild. The model learns both the outline and the pose of the character (input frames and the character ©Soyuzmultfilm).

	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.025$
Ours	**67.18**	**46.39**	**24.17**
UCN	67.07	43.84	21.50
PWC-Net	62.92	40.74	18.47

Table 2: Correspondence estimation results using PCK as the metric. Results are averaged across six characters.

they are typically not suitable for cartoon data, as it usually lacks texture and exhibits strong expressive articulations. Since our Deformation Network fits the same template to every frame, we can estimate correspondences between any pair of frames via the template. Table 2 compares correspondences from our method with those obtained from a recent flow-based approach (PWC-Net [55]), and with a supervised correspondence method, Universal Correspondence Networks (UCN) [14]. We use the Percentage of Correct Keypoints (PCK) as the evaluation metric. Given a threshold $\alpha \in (0, 1)$, the correspondence is classified as correct if the predicted point lies within Euclidean distance $\alpha \cdot L$ of the ground-truth, in which $L = \max(\text{width}, \text{height})$ is the image size. Results are averaged across pairs of images in the test set for six different characters. Our method outperforms UCN and PWC-Net in all cases, since it has a better model for underlying character structure. Note that our method requires a single segmented frame, while UCN is trained with ground truth key-point correspondences, and PWC-Net is supervised with ground truth flow.

4.4. Characters in the Wild

We also extend our approach to TV cartoon characters "in the wild". The main challenge posed by these data is that the character might only be a small element of a complex scene. Given a raw video[1], we first use the standard tracking tool in Adobe After Effects [1] to extract per-frame bounding boxes that encloses the character. Now we can use our architecture to only analyze the character appearance. However, since it still appears over a complex background, we modify our reconstruction loss to be computed only over the rendered character:

$$L_{\text{rec}}^{\text{masked}} = \frac{\left\| x \odot R(V_{pred}, F, I^{\mathbb{1}}) - R(V_{pred}, F, I^{uv}) \right\|^2}{\sum R(V_{pred}, F, I^{\mathbb{1}})} \tag{10}$$

where x is the input image, $\odot$ is the Hadamard product, and $R(V_{pred}, F, I^{\mathbb{1}})$ is a mask, produced by rendering a mesh with 1-valued (white) texture over a 0-valued (black) background. By applying this mask, we compare only the relevant regions of input and rendered images, i.e. the foreground character. The term in the denominator normalizes the loss by the total character area. To further avoid shrinking or expansion of the character (which could be driven by a partial match), we add an area loss penalty:

$$L_{area} = \left| \sum R(V_{pred}, F, I^{\mathbb{1}}) - \sum R(V_{init}, F, I^{\mathbb{1}}) \right|^2 \tag{11}$$

Our final loss is defined similarly to Equation 4 but uses the masked reconstruction loss $L_{\text{rec}}^{\text{masked}}$ and adds L_{area} loss with weight 2×10^{-3} (L_{reg} and L_{joints} are included with original weights). We present our results in Figure 7, demonstrating that our framework can be used to capture character appearance in the wild.

5. Conclusion and Future Work

We present novel neural network architectures for learning to register deformable mesh models of cartoon characters. Using a template-fitting approach, we learn how to adjust an initial mesh to images of a character in various poses. We demonstrate that our model successfully learns to deform the meshes based on the input images. Layering is introduced to handle occlusion and moving limbs. Varying motion and textures are captured with a Deformation Network and an Appearance Refinement Network respectively. We show applications of our model in inbetweening, user-constrained deformation and correspondence estimation. In the future, we consider using our model for applications such as motion re-targetting.

[1] e.g. https://www.youtube.com/watch?v=hYCbxtzOfL8

References

[1] Adobe. Adobe after effects, 2019. 8

[2] Adobe. Adobe character animator, 2019. 1, 4

[3] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM transactions on graphics (TOG)*, volume 22, pages 587–594. ACM, 2003. 2

[4] B. Allen, B. Curless, Z. Popović, and A. Hertzmann. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–156. Eurographics Association, 2006. 2

[5] Y. Amit, U. Grenander, and M. Piccioni. Structural image restoration through deformable templates. *Journal of the American Statistical Association*, 86(414):376–387, 1991. 2

[6] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2, 3

[7] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh. Modeling facial geometry using compositional vaes. *In practice*, 1:1, 2018. 2

[8] W. Baxter, P. Barla, and K. Anjyo. N-way morphing for 2d animation. *Computer Animation and Virtual Worlds*, 20(2-3):79–87, 2009. 5

[9] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003. 2

[10] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017. 2

[11] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: Motion capturing cartoons. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 399–407, 2002. 2

[12] E. Catmull. The problems of computer-assisted animation. In *ACM SIGGRAPH Computer Graphics*, volume 12, pages 348–353. ACM, 1978. 2

[13] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A simple geometric model for elastic deformations. In *ACM transactions on graphics (TOG)*, volume 29, page 38. ACM, 2010. 6

[14] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016. 2, 7, 8

[15] W. T. Corrêa, R. J. Jensen, C. E. Thayer, and A. Finkelstein. Texture mapping for cel animation. pages 435–446, July 1998. 1

[16] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5 d modelling techniques. In *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No. 01TH8596)*, pages 192–200. IEEE, 2001. 2

[17] M. Dvorožňák, P. Bénard, P. Barla, O. Wang, and D. Sỳkora. Example-based expressive animation of 2d rigid bodies. *ACM Transactions on Graphics (TOG)*, 36(4):127, 2017. 2

[18] M. Dvorožnák, W. Li, V. G. Kim, and D. Sỳkora. Toonsynth: example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics (TOG)*, 37(4):167, 2018. 2, 4, 6

[19] X. Fan, A. Bermano, V. G. Kim, J. Popovic, and S. Rusinkiewicz. Tooncap: A layered deformable model for capturing poses from cartoon characters. *Expressive*, 2018. 2

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 3

[21] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013. 2

[22] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Shape correspondences from learnt template-based parametrization. *arXiv preprint arXiv:1806.05228*, 2018. 2

[23] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 3

[24] H. B. Hamu, H. Maron, I. Kezurer, G. Avineri, and Y. Lipman. Multi-chart generative surface modeling. *SIGGRAPH Asia*, 2018. 2

[25] P. Henderson and V. Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. *arXiv preprint arXiv:1807.09259*, 2018. 2

[26] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie. Stacked generative adversarial networks. In *CVPR*, volume 2, page 3, 2017. 2

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017. 2, 3

[28] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 30(4):78:1–78:8, 2011. 2

[29] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*, 2018. 2

[30] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 3

[31] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2314, 2013. 7

[32] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 2

[33] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[34] A. Kort. Computer aided inbetweening. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 125–132. ACM, 2002. 5

[35] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and B. Joan. Surface networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018. 2

[36] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017. 2

[37] Z. Levi and C. Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics*, 21(2):264–277, 2015. 6

[38] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia. Deformable shape completion with graph convolutional autoencoders. *arXiv preprint arXiv:1712.00268*, 2017. 2

[39] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011. 7

[40] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015. 2

[41] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 3

[42] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 2

[43] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018. 2

[44] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[45] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3d faces using convolutional mesh autoencoders. *arXiv preprint arXiv:1807.10267*, 2018. 2

[46] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016. 7

[47] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 2

[48] I. K. R{i1
za Alp Güler, Natalia Neverova. Densepose: Dense human pose estimation in the wild. *arXiv*, 2018.

[49] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. 2

[50] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996. 1, 3

[51] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017. 2

[52] 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. 1, 2, 4, 5, 6

[53] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *The IEEE conference on computer vision and pattern recognition (CVPR)*, volume 3, page 4, 2017. 2

[54] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. 2, 3, 6

[55] 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, 4, 5, 6, 8

[56] D. Sỳkora, J. Dingliana, and S. Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 25–33. ACM, 2009. 2, 5

[57] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3d mesh models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[58] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, volume 2, page 8, 2017. 2

[59] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[60] A. Venkat, S. S. Jinka, and A. Sharma. Deep textured 3d reconstruction of human bodies. 2

[61] K. Wampler. Fast and reliable example-based mesh ik for stylized deformations. *ACM Transactions on Graphics (TOG)*, 35(6):235, 2016. 2

[62] B. Whited, G. Noris, M. Simmons, R. W. Sumner, M. Gross, and J. Rossignac. Betweenit: An interactive tool for tight inbetweening. In *Computer Graphics Forum*, volume 29, pages 605–614. Wiley Online Library, 2010. 5

[63] H. Yang, W. Lin, and J. Lu. Daisy filter flow: A generalized approach to discrete dense correspondences. CVPR, 2014. 7

[64] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016. 3

[65] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 2

[66] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017. 2

[67] S. Zuffi and M. J. Black. The stitched puppet: A graphical model of 3d human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3546, 2015. 2

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

We provide results for new characters and additional samples of existing characters in Figure A. The top-left character is from Textoons [4], and the top-right one is from ToonSynth [1]. 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). As we observe, 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. 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] M. Dvorožnák, W. Li, V. G. Kim, and D. Sỳkora. Toonsynth: example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics (TOG)*, 37(4):167, 2018. 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

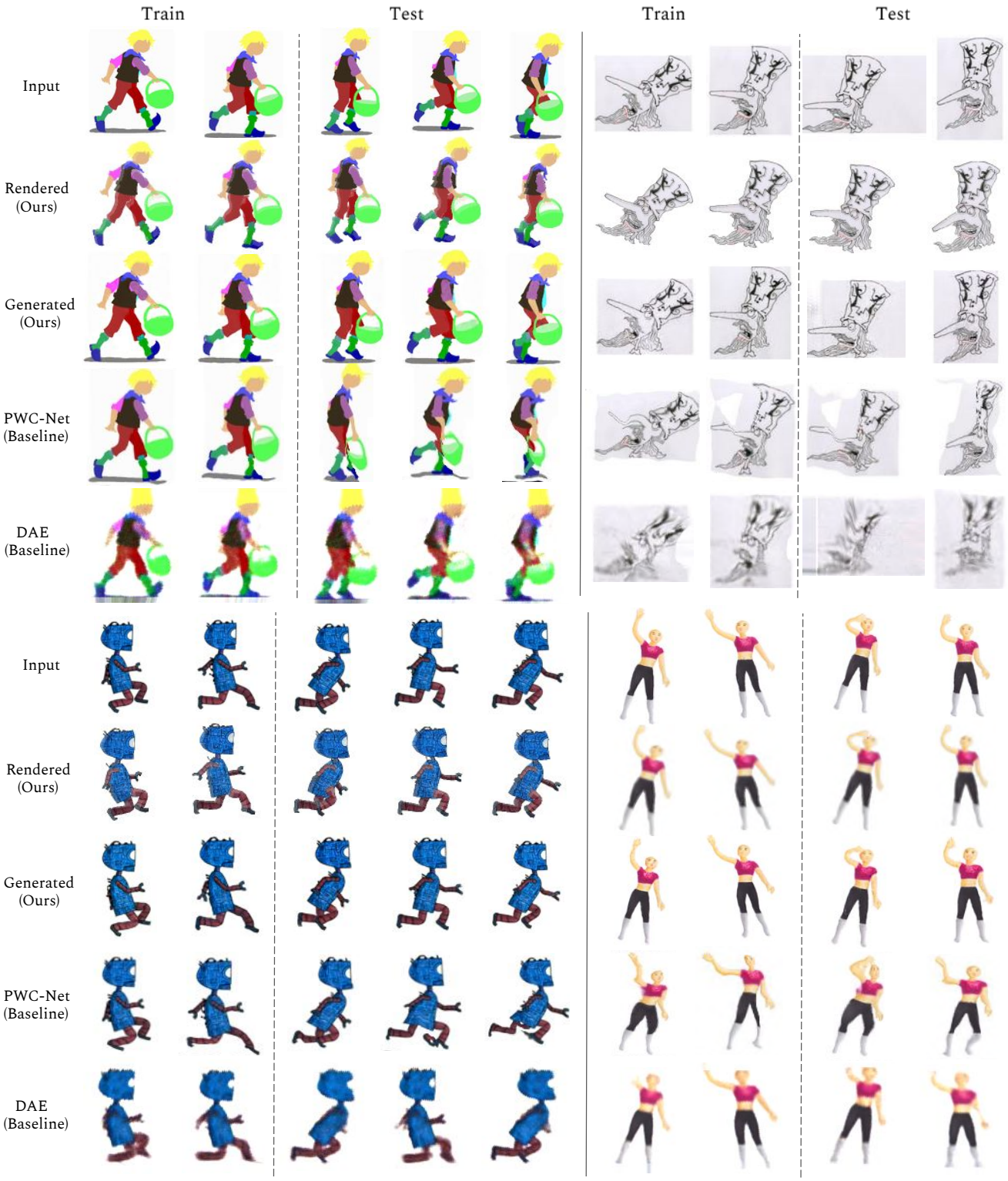

Figure A: Input images, our rendered and final results, followed by results obtained with PWC-Net [3] and DAE [2] (characters in the first row ©Anifilm Studio, bottom-left character ©Adobe Character Animator, bottom-right character ©Zuzana Studená).

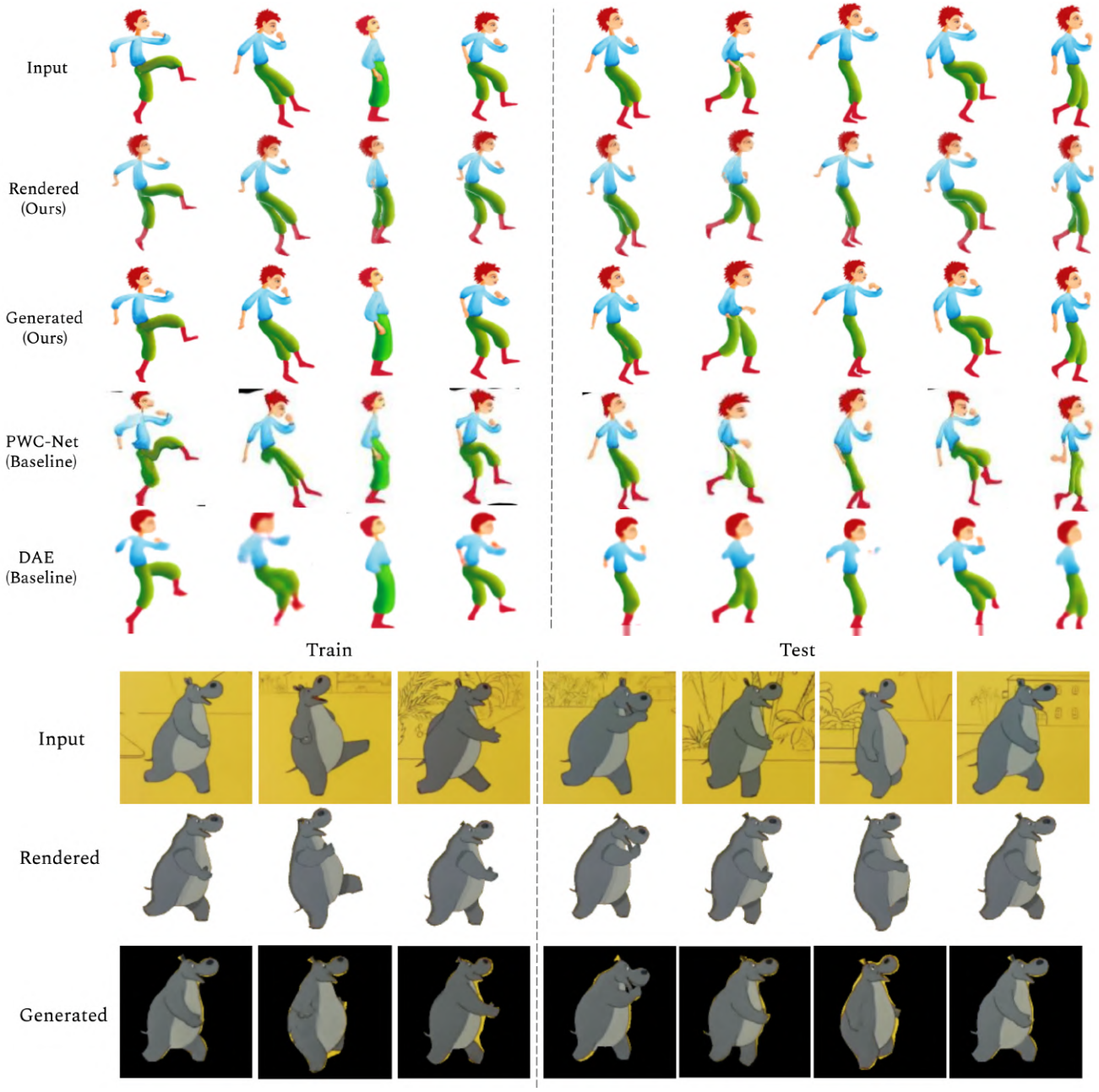

Figure A: (cont.) Input images, our rendered and final results, followed by results obtained with PWC-Net [3] and DAE [2] (Top character ©Zuzana Studená, bottom character © Soyuzmultfilm).

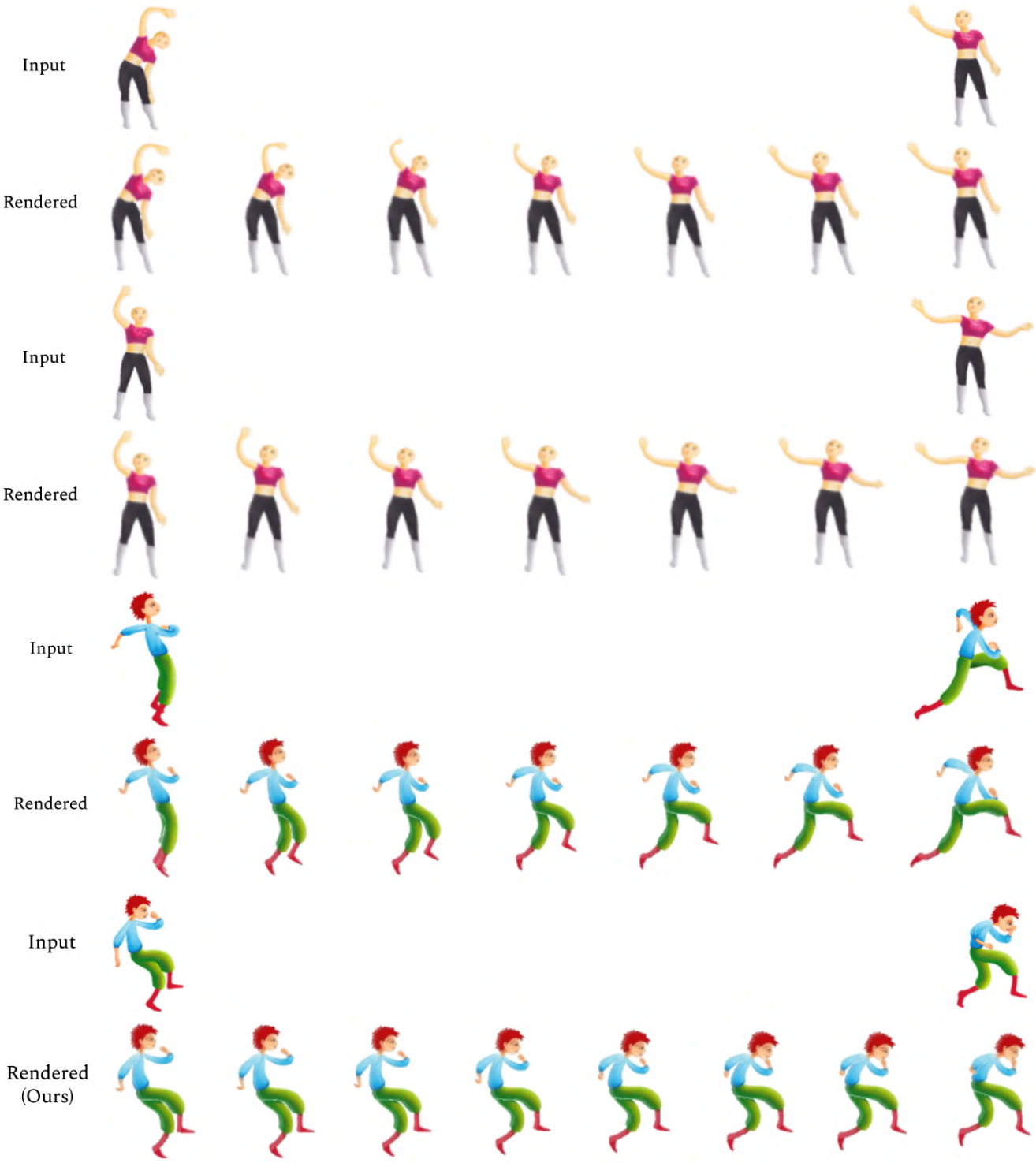

Figure B: Inbetweening results. Two given test images (first row) are encoded. The resulting latent vectors are linearly interpolated yielding the resulting rendered images (input drawings ©Zuzana Studená).

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 (input drawings ©Zuzana Studená).

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