Layered Neural Rendering for Retiming People in Video

ERIKA LU, University of Oxford, Google Research
FORRESTER COLE, Google Research
TALI DEKEL, Google Research
WEIDI XIE, University of Oxford
ANDREW ZISSERMAN, University of Oxford
DAVID SALESIN, Google Research
WILLIAM T. FREEMAN, Google Research
MICHAEL RUBINSTEIN, Google Research

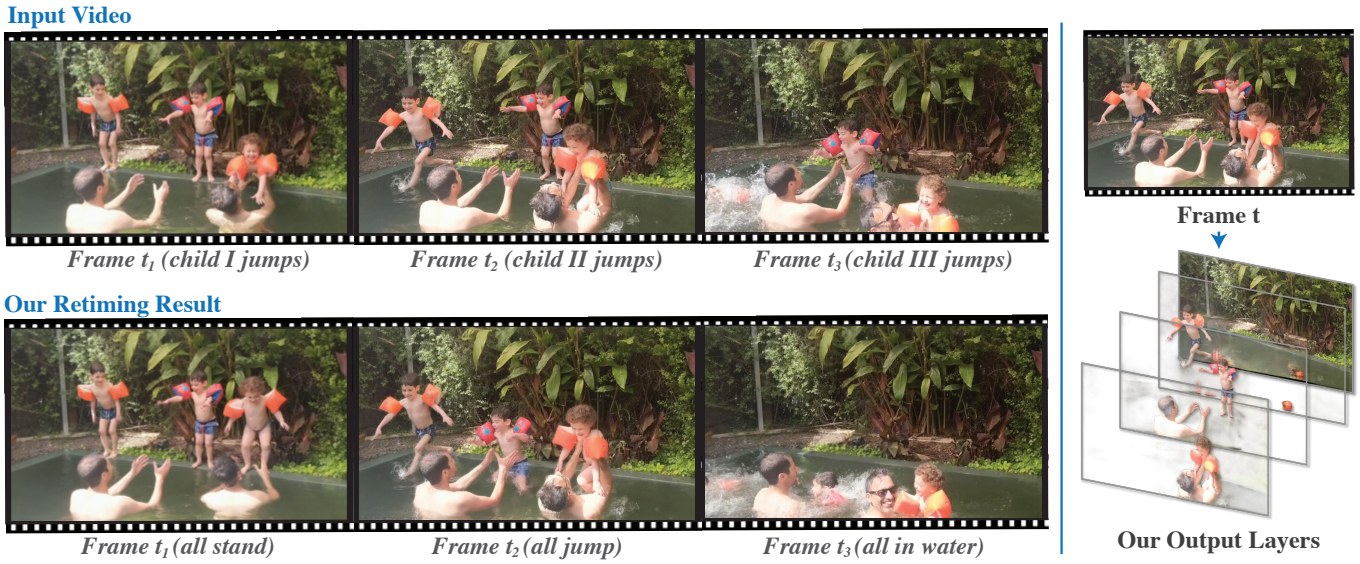


Fig. 1. **Making all children jump into the pool together — in post-processing!** In the original video (left, top) each child is jumping into the pool at a different time. In our computationally retimed video (left, bottom), the jumps of children I and III are time-aligned with that of child II, such that they all jump together into the pool (notice that child II remains unchanged in the input and output frames). In this paper, we present a method to produce this and other people retiming effects in natural, ordinary videos. Our method is based on a novel deep neural network that learns a layered decomposition of the input video (right). Our model not only disentangles the motions of people in different layers, but can also capture the various scene elements that are *correlated* with those people (e.g., water splashes as the children hit the water, shadows, reflections). When people are retimed, those related elements get automatically retimed with them, which allows us to create realistic and faithful re-renderings of the video for a variety of retiming effects. The full input and retimed sequences are available in the supplementary video.

We present a method for retiming people in an ordinary, natural video — manipulating and editing the time in which different motions of individuals in the video occur. We can temporally align different motions, change the speed of certain actions (speeding up/slowing down, or entirely "freezing" people), or "erase" selected people from the video altogether. We achieve these effects computationally via a dedicated learning-based layered video representation, where each frame in the video is decomposed into separate RGBA layers, representing the appearance of different people in the video. A key property of our model is that it not only disentangles the direct motions of each person in the input video, but also correlates each person *automatically* with the scene changes they generate—e.g., shadows, reflections, and motion of loose clothing. The layers can be individually retimed and recombined into a new video, allowing us to achieve realistic, high-quality renderings of retiming effects for real-world videos depicting complex actions and involving multiple individuals, including dancing, trampoline jumping, or group running.

CCS Concepts: • **Computing methodologies** → **Computational photography**; **Neural networks**.

Additional Key Words and Phrases: retiming, neural rendering, video layer decomposition

ACM Reference Format:
Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. 2020. Layered Neural

0730-0301/2020/12-ART256
https://doi.org/10.1145/3414685.3417760

Rendering for Retiming People in Video. *ACM Trans. Graph.* 39, 6, Article 256 (December 2020), 14 pages. https://doi.org/10.1145/3414685.3417760

1 INTRODUCTION

By manipulating the timing of people's movements, we can achieve a variety of effects that can change our perception of an event recorded in a video. In films, altering time by speeding up, slowing down, or synchronizing people's motions is often used for dramatizing or de-emphasizing certain movements or events. For example, by freezing the motions of some people in an action-packed video while allowing others to move, we can focus the viewer's attention on specific people of interest. In this paper, we aim to achieve such effects computationally by retiming people in everyday videos.

The input to our method is an ordinary natural video with multiple people moving, and the output is a realistic re-rendering of the video where the timing of people's movements is modified. Our method supports various retiming effects including aligning motions of different people, changing the speed of certain actions (e.g., speeding up/slowing down, or entirely "freezing" people). In addition, our method can also "erase" selected people from the video. All these effects are achieved via a novel deep neural network-based model that learns a *layered decomposition of the input video*, which is the pillar of our method. Note that in this paper we focus solely on temporal warping. That is, each person's pose in our output exists in some frame in the input; we do not generate new, unseen poses or viewpoints.

Motion retiming has been studied so far mostly in the context of character animation, for editing a character's motion to match a desired duration or target velocity at a given time (e.g., [McCann et al. 2006; Yoo et al. 2015]). In this paper, we take retiming to the realm of *natural real videos*. In the character animation domain, the main challenge is to retime the motion of a set of joints, with the spatiotemporal correlations that exist between them. Analogously, manipulating the timing of people in video not only requires modifying people's own motions, but also the various elements in the scene that they "cause" and are *correlated* with them — shadows, reflections, the flowing dress of a dancer, or splashing water (Fig. 1). When we retime people, we need to make sure that all those correlative events in the scene follow properly and respect the timing changes. Furthermore, unlike character animation, we do not have any ground truth 3D model of the scene over time; hence, rendering photorealistic, high-quality retiming effects in video is much more challenging. More specifically, retiming motions in videos can often result in new *occlusions* and *disocclusions* in the scene. Rendering the scene content in disoccluded regions and maintaining correct depth ordering between subjects are essential for achieving a realistic effect. Finally, as in any video synthesis task, achieving a temporally coherent result is challenging — small errors such as subtle misalignment between frames immediately show up as visual artifacts when the frames are viewed as a video.

The core of our technique, which we dub *layered neural rendering*, is a novel deep neural-network-based model that is optimized per-video to decompose every frame into a set of layers, each consisting of an RGB color image and an opacity matte α (referred to altogether as "RGBA"). We design and train our model such that each RGBA layer over time is associated with specific people in the video (either a single person, or a group of people predefined by the user). Crucially, our method does not require dynamic scene elements such as shadows, water splashes, and trampoline deformations to be manually annotated or explicitly represented; rather, only a rough parameterization of the *people* is required, which can be obtained using existing tools with minor manual cleanup for challenging cases. The model then *automatically* learns to group people with their correlated scene changes. With the estimated layers, the original frames of the video can be reconstructed using standard back-to-front compositing. Importantly, retiming effects can be produced by simple operations on layers (removing, copying, or interpolating specific layers) without additional training or processing.

Our model draws inspiration from recent advances in neural rendering [Thies et al. 2019], and combines classical elements from graphics rendering with deep learning. In particular, we leverage human-specific models and represent each person in the video with a single deep-texture map that is used to render the person in each frame. Having a unified representation of a person over time allows us to produce temporally coherent results.

We demonstrate realistic, high-quality renderings of retiming effects for real-world videos with complex motions, including people dancing, jumping, and running. We also provide insights into why and how the model works through a variety of synthetic experiments.

2 RELATED WORK

Video retiming. Our technique applies time warps (either designed manually, or produced algorithmically) to people in the video, and re-renders the video to match the desired retiming. As such it is related to a large body of work in computer vision and graphics that performs temporal remapping of videos for a variety of tasks. For example, [Bennett and McMillan 2007] sample the frames of an input video non-uniformly to produce computational time-lapse videos with desired objectives, such as minimizing or maximizing the resemblance of consecutive frames. [Zhou et al. 2014] use motion-based saliency to nonlinearly retime a video such that more "important" events in it occupy more time. [Davis and Agrawala 2018] retime a video such that the motions (visual rhythm) in the time-warped video match the beat of a target music.

Other important tasks related to video retiming are video summarization (e.g., [Lan et al. 2018]) and fast-forwarding [Joshi et al. 2015; Poleg et al. 2015; Silva et al. 2018], where frames are sampled from an input video to produce shorter summaries or videos with reduced camera motion or shake; or interactive manipulation of objects in a video (e.g., using tracked 2D object motion for posing different objects from a video into a still frame that never actually occurred [Goldman et al. 2008].)

Most of these papers retime *entire video frames*, by dropping or sampling frames. In contrast, we focus on people, and people's motions, and our effect is applied at the person/layer level. While many methods exist for processing the video in the sub-frame or patch level — both for retiming (e.g., [Goldman et al. 2008; Pritch et al. 2008]) and for various other video manipulation tasks, such as object removal, infinite video looping, etc. (e.g., [Agarwala et al. 2005; Barnes et al. 2010; Wexler et al. 2007]) — none of these works

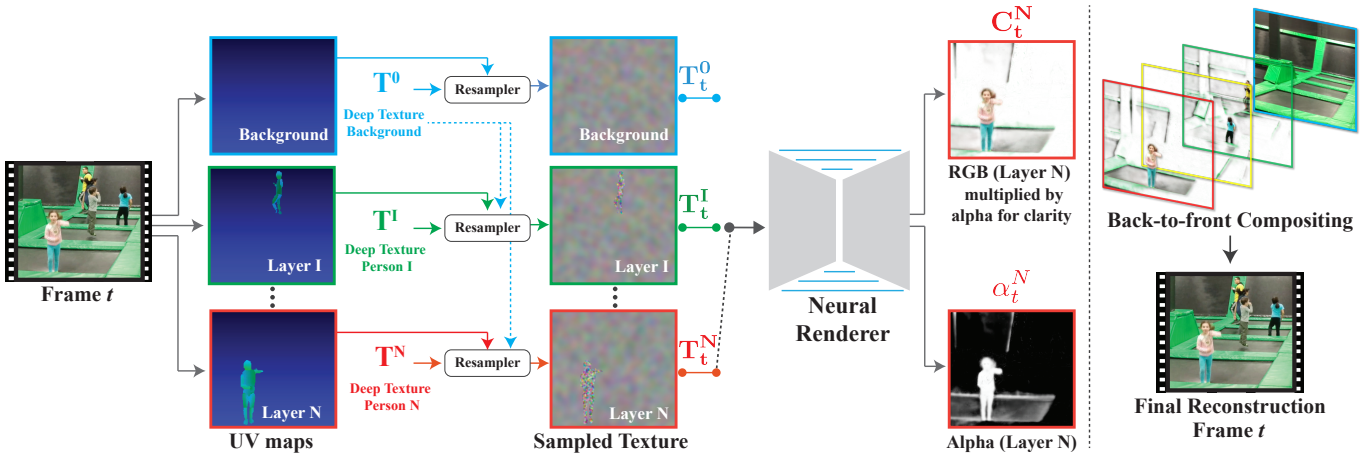


Fig. 2. **Layered Neural Rendering.** Our neural rendering model decomposes each frame of the video into a set of color (C_t^i) and opacity (α_t^i) layers. Each layer is associated with specific people in the video (either a single person, or a group of people predefined by the user). The layers are computed in separate forward passes by feeding to the neural renderer a deep-texture map that corresponds to a single layer. In particular, we represent each person in the video with a single deep-texture map T^i, and the scene background is represented with a deep-texture map T^0. Given pre-computed UV maps, those deep-texture maps are resampled and composited to form the input to the neural renderer. The set of estimated layers can then be composited in a back-to-front fashion to reconstruct the original frames. Retiming effects can be achieved via simple operations on the layers.

can handle automatically correlative motions and changes in the video such as shadows and reflections.

Manipulating human poses in video. In the context of manipulating people's motions, several recent methods have been proposed for transferring motion between two people captured in different videos [Aberman et al. 2019; Chan et al. 2019; Lee et al. 2020; Zhou et al. 2019] or transferring motion from a low-dimensional signal [Gafni et al. 2020]. However, there are two main distinctions between these methods and ours: (i) motion transfer methods are focused on "puppetry" — generating people in new unseen poses — whereas our method is tailored solely for time warping (we do not generate new poses); and (ii) existing motion transfer methods consider only a single person in each video — they do not model occlusions/disocclusions between multiple people — and with the exception of [Gafni et al. 2020], they also do not capture the correlations between people and their environment, elements that are crucial for producing realistic retiming effects.

Image and video matting. Our work is also related to image and video matting as we decompose each input video frame into a set of RGBA layers. Nevertheless, both traditional matting techniques (e.g., [Bai et al. 2009; Chuang et al. 2002; Li et al. 2005; Wang et al. 2005]) as well as more recent learning-based matting methods [Hou and Liu 2019; Xu et al. 2017] do not capture correlated effects such as shadows and reflections. Moreover, matting methods typically rely on accurate trimaps that are often given manually by the user, whereas we automatically generate rougher trimaps from detected keypoints. Even though our trimaps are not as accurate, we are still able to learn complex effects such as loose clothing, even when such regions are not included in the initial, estimated trimaps. Finally, existing matting methods cannot handle well entirely semi-transparent objects like reflections (in matting trimaps, pixels that are labeled "opaque" are fully assigned to the foreground). We show comparisons with image matting in Section 6.

Layered representations for video. Decomposing videos into layers is a classical problem in computer vision originally proposed in [Wang and Adelson 1994], which has inspired a large body of work (e.g. [Jojic and Frey 2001; Kumar et al. 2008; Nandoriya et al. 2017; Xue et al. 2015; Zitnick et al. 2004]). Recently, several works have leveraged the power of deep neural networks to decompose videos into layers. For example, [Zhou et al. 2018] and [Srinivasan et al. 2019] train a network on input stereo pairs of images to predict layers that correspond to physically accurate depth, with the goal of view synthesis for static scenes. Our layers are not meant to be strictly accurate in terms of depth, as we do not aim to generate scenes from novel viewpoints, but rather composite scene elements from different points of a dynamic video.

Alayrac et al. [2019a; 2019b] train a model to decompose synthetically blended real videos and then apply it to natural videos to remove reflections, shadows, and smoke. Their method is general rather than person-specific like ours, and they use audio as a cue for separating layers, rather than people geometry as we do.

Finally, Gandelsman et al. [2019] extend the idea of "Deep Image Prior" (DIP) networks [Ulyanov et al. 2018] for decomposing a single image or a video into two layers. Their key observation is that the structure of a CNN provides a prior that drives each layer towards a natural image. As above, they have limited control over their output layers and rely entirely on CNN properties to produce meaningful layers. Our method also leverages this property of CNNs (see Section 5), yet provides control over the decomposition, which is required for retiming effects. Furthermore, their method is designed for two-layer decomposition, and generalizing it to N layers requires $N\times$ learnable parameters. In contrast, our method can produce an arbitrary number of layers with the same, fixed number of

network parameters. (We show a comparison with Double-DIP in the supplementary material.)

Neural rendering. Neural networks have recently begun to be used as a final rendering layer for 3D scenes [Kim et al. 2018; Liu et al. 2019; Martin-Brualla et al. 2018; Meshry et al. 2019; Sitzmann et al. 2019; Thies et al. 2019]. A neural network can be trained to bridge the gap between incomplete or inaccurate input geometry and a photorealistic result. Examples of incomplete input are point clouds [Meshry et al. 2019], voxels [Sitzmann et al. 2019], partial 3D scans or 3D models of humans [Kim et al. 2018; Liu et al. 2019; Martin-Brualla et al. 2018], and textured proxy geometry [Thies et al. 2019]. Our work adopts the textured proxy approach — creating geometric proxies for each person in the scene and texturing them with a deep texture map [Thies et al. 2019]. Importantly, we combine this approach in a neural rendering model that outputs a layered decomposition of each frame of the video. This is in contrast to existing neural rendering methods, which output the final reconstruction directly. Our layered decomposition is the key to producing realistic high-quality retiming effects, as demonstrated in Fig. 11 and discussed in detail in Section 6.4.

3 OVERVIEW

A key challenge in generating realistic retiming effects in videos is ensuring that when people's motions are retimed, their related effects in the scene follow with them. For example, if we freeze a person, their shadow should freeze as well.

We achieve this via a learning-based approach, which, at its core, decomposes each frame of the input video into a *human-specific set of layers*. That is, each layer is associated with specific people in the video—either a single person, or a group of people (Section 4.1). A key feature in our approach is that only the people in the video are modeled explicitly, while the rest of the scene elements correlated with each person are *inferred automatically* by our layered neural rendering pipeline.

Our pipeline is illustrated in Fig. 2. Our model is trained per-video, observing only the input video (without any additional examples), and is trained in a self-supervised manner, by learning to reconstruct the original video frames by predicting the layers (Section 4.2). We use off-the-shelf methods (AlphaPose [Fang et al. 2017], DensePose [Güler et al. 2018]) in combination with our own techniques to represent each person at each frame (Section 4.5). This representation is then passed on to the neural renderer and is used for seeding the layer assignment (person per layer). While the input to the neural renderer includes only the people (and a static background), the renderer's task is to generate layers that reconstruct the *full* input video. Thus, it must assign all the remaining, time-varying scene elements (e.g., shadows, reflections) into the appropriate peoples' layers.

The neural renderer succeeds in this task because the network design and training procedure (Section 4.3) encourages scene elements that are correlated with a layer to be captured faster (reconstructed earlier in training) than non-correlated elements. This property is related to the "Deep Image Prior" (DIP) work [Ulyanov et al. 2018], which showed that overfitting a network to a natural image required fewer gradient descent steps than fitting a noise image. We further explore how correlated effects are captured in the correct layers through synthetic experiments in Section 5.

4 METHOD

4.1 A Layered Video Representation

Given an input video V, our goal is to decompose each frame $I_t \in V$ into a set of RGBA (color + opacity) layers:

$$\mathcal{L}_t = \{L_t^i\}_{i=1}^N = \{C_t^i, \alpha_t^i\}_{i=1}^N, \tag{1}$$

where C_t^i is a color image and α_t^i is an opacity map (matte). The i^{th} layer for all frames L_*^i is associated with person i in the video. We add an additional background layer L_t^0, not associated with any person, that learns the background color.

Given this layered representation and a back-to-front ordering for the layers, denoted by o_t, each frame of the video can be rendered using the standard "over" operator [Porter and Duff 1984]. We denote this operation by:

$$\hat{I}_t = Comp(\mathcal{L}_t, o_t) \tag{2}$$

We assume that the compositing order o_t is known, yet time varying, i.e., the depth ordering between people may change throughout the video (see Section 4.5).

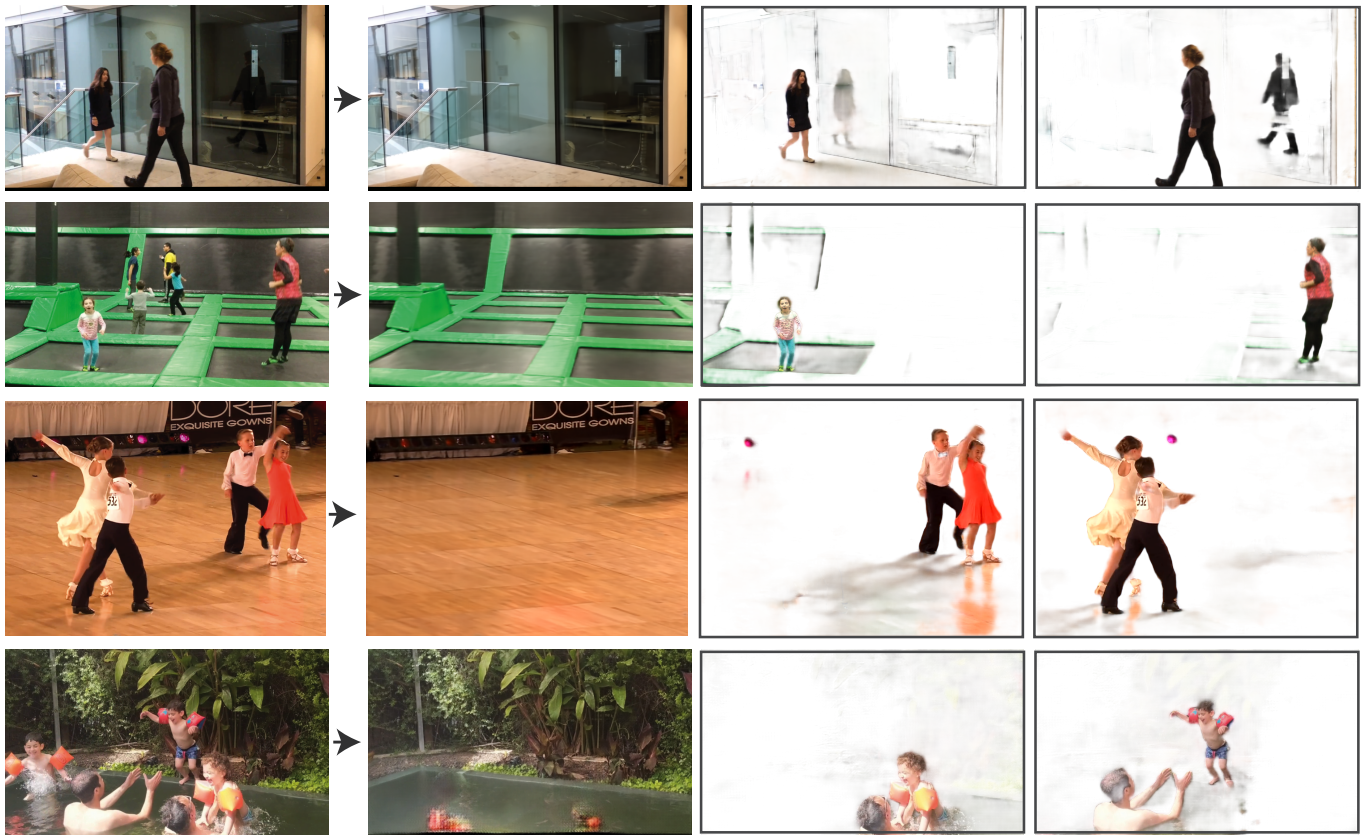
A key property of this representation is that retiming effects can be achieved by simple operations on individual layers. For example, removing person i from frame t can be done simply by removing the i^{th} layer from the composition, i.e., by substituting in $\mathcal{L}_t \setminus L_t^i$ into Eq. 2. Similarly, generating a video where person i is frozen at a time t_0 is achieved by copying $L_{t_0}^i$ over L_t^i for all frames. We expand on other operations in Section 6.

4.2 Layered Neural Rendering

Decomposing a real-world video into a desired set of layers is a difficult, under-constrained problem – there are numerous possible decompositions that can provide an accurate reconstruction of the original frame I_t. For example, a single visible layer that contains the entire frame can perfectly reconstruct the video. We therefore constrain our layered neural renderer to steer the solution towards the desired person-specific decomposition. We do so in several ways including incorporating human specific representations, tailoring the input to the network, and applying dedicated losses and training regimes.

As mentioned in Section. 3, the layers are predicted in *separate feed-forward passes* through the neural renderer, where the input for each pass represents only one person (or a pre-defined group of people). In doing so, we control the association of people with the output layers. More specifically, we construct the input to the renderer as follows:

Person representation. We parameterize each person in the video with a single human texture atlas T^i and a per-frame UV-coordinate map UV_t^i, which maps each pixel in the human region in frame I_t to the texture atlas. We use the parameterization of the SMPL model [Loper et al. 2015] that can be estimated from an input image using existing methods (e.g., DensePose [Güler et al. 2018]). This provides a unified parameterization of the person over time and a convenient model for appearance and geometry. We follow a



(a) Original Frame (b) Background Layer (c) Representative People Layers

Fig. 3. **Example layer decompositions by our model**. For each example, we show (a) an original frame from the video, (b) the predicted background layer, and (c) two representative (RGBA) layers associated with people (in *Ballroom* and *Splash*, third and forth rows, respectively, the grouping of people into layers was specified manually; in the other examples each person is assigned to a separate layer). Our model successfully disentangles the people into different layers along with the visual changes they cause on the environment, such as trampoline deformations, shadows, reflections, and water splashes. (Artifacts at the bottom of the background layer in the pool scene (bottom row) are due to the fact that those regions are never visible in the input video.)

similar approach to Thies *et. al.* [2019] and replace the classic RGB texture map with a learnable, high-dimensional texture map, which can encode more powerful and richer appearance information. The deep texture maps are then decoded into RGB values using a neural rendering network. To represent person i at time t, we sample its deep texture map T^i using UV_t^i, obtaining T_t^i.

Background representation. The background is represented with a single texture map T^0 for the entire video. Sampling from the background is performed according to a UV map UV_t^0. For a static camera, UV_t^0 is an identical xy coordinate grid for all frames. If homography transformations estimated from camera tracking are available (see Section 4.6), UV_t^0 is the result of transforming an xy coordinate grid by the homography for frame t.

Input-Output. To construct the input, we place the background's UV map behind each person's UV map to provide background context for the renderer (see Fig. 2). That is, the input for layer i at time t is the sampled deep texture map T_t^i, which consists of person i's sampled texture placed over the sampled background texture. Each of the re-sampled texture layers $\{T_t^{i\,N}_{i=1}\}$ as well as the background re-sampled texture is fed to the renderer separately. The output is $L_t^i = \{C_t^i, \alpha_t^i\}$, the time-varying color image and opacity map for that layer, respectively.

The neural renderer is trained to reconstruct the original frames from the predicted layers, using Eq. 2. The depth ordering of the people layers, which may vary over time, is provided by the user. To accurately reconstruct the *entire* original video, the neural renderer must reconstruct any space-time scene element that is not represented by the input UV maps, such as shadows, reflections, loose clothing or hair. Crucially, disentangling people into separate inputs is key for guiding the neural renderer to assign those scene elements into the layers with which they are most strongly correlated in space and/or time (e.g. attached to a person, moves like the person, etc.). We expand on this in greater detail in Section 5.

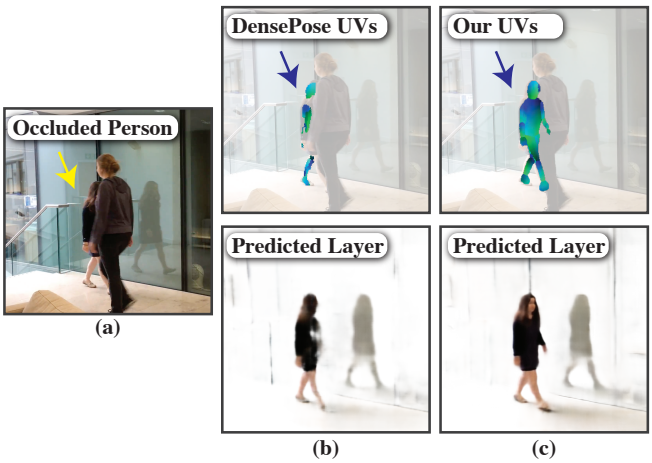


Fig. 4. **Full-body UVs.** Existing tools such as DensePose [Güler et al. 2018] provide UV coordinates only for fully visible regions. In contrast, our method is able to produce full-body UVs for occluded people by detecting and interpolating keypoints, then passing the occluded person's full skeleton to a trained network that converts it to a UV map. This allows us to create editing effects that require disoccluding people.

4.3 Training

We now turn to the task of learning the optimal parameters θ of the neural renderer, and the set of latent textures $\{T^i\}_{i=0}^N$ by optimizing the learned decomposition for each frame.

One necessary property of the learned decomposition is that it will allow us to accurately reconstruct the original video. Formally,

$$\mathbf{E}_{\text{recon}} = \frac{1}{K}\sum_t \|I_t - Comp(\mathcal{L}_t, o_t)\|_1, \tag{3}$$

where $\mathcal{L}_t$ are the output layers for frame t, o_t is the compositing order, and K is the total number of frames.

The reconstruction loss alone is not enough to make the optimization converge from a random initialization, so we bootstrap training by encouraging the learned alpha maps α_t^i to match the people segments that are associated with layer i. To do so, we apply the following loss:

$$\mathbf{E}_{\text{mask}} = \frac{1}{K}\frac{1}{N}\sum_t\sum_i D(m_t^i, \alpha_t^i), \tag{4}$$

where m_t^i is a trimap derived from the UV maps UV_t^i (see Fig. 10), and $D(\ )$ is a distance measure. Masks are trimaps with values in $[0, 0.5, 1]$, where the uncertain area is produced by morphological dilation of the binary UV mask. For a trimap m, let b_0 be the binary mask of the pixels where $m = 0$, with b_1 defined likewise. Losses in the positive and negative regions are balanced while the uncertain area is ignored. The distance measure is:

$$D(m, \alpha) = \frac{\|b_1 \odot (1.0 - \alpha)\|_1}{2\,\|b_1\|_1} + \frac{\|b_0 \odot \alpha\|_1}{2\,\|b_0\|_1} \tag{5}$$

where $\odot$ is the element-wise (Hadamard) product.

Since the UV mask does not include information from correlated effects such as shadows and reflections, $\mathbf{E}_{\text{mask}}$ is only used to bootstrap the model and is turned off as optimization progresses.

We further apply a regularization loss to the opacities α_t^i to encourage them to be spatially sparse. This loss is defined as a mix of L_1 and an approximate-L_0:

$$\mathbf{E}_{\text{reg}} = \frac{1}{K}\frac{1}{N}\sum_t\sum_i \gamma\,\|\alpha_t^i\|_1 + \Phi_0(\alpha_t^i) \tag{6}$$

where $\Phi_0(x) = 2 \cdot \texttt{Sigmoid}(5x) - 1$ smoothly penalizes non-zero values of the alpha map, and γ controls the relative weight between the terms.

Our total loss is then given by:

$$\mathbf{E}_{\text{total}} = \mathbf{E}_{\text{recon}} + \gamma_m \mathbf{E}_{\text{mask}} + \beta \mathbf{E}_{\text{reg}}, \tag{7}$$

where γ_m and β control the relative weights of the terms.

As usual with non-linear optimization, the results are sensitive to the relative weights of the error terms. To produce the results in this paper, we vary the loss weights based on the current epoch e and the "bootstrap" epoch e_b, which is the first epoch where $\mathbf{E}_{\text{mask}} < 0.02$:

$$\gamma_m = \begin{cases} 50 & e \le e_b \\ 5 & e_b < e \le 2e_b \\ 0 & \text{otherwise} \end{cases} \quad \gamma = \begin{cases} 2 & e \le 200 \\ 0 & \text{otherwise} \end{cases} \quad \beta = 0.005 \tag{8}$$

This schedule puts a heavy initial loss on the masking term to force the optimization towards a plausible solution, then relaxes it to allow the optimization to introduce effects not present in the masks.

4.4 High-Resolution Refinement and Detail Transfer

We take a multi-scale approach and first train our model using Eq. 7 on a downsampled version of the original video. We then upsample the result to the original resolution using a separate lightweight refinement network, which consists of several residual blocks (see Appendix for details) operating on each RGBA layer separately.

We can avoid the additional expense of training with perceptual and adversarial losses by directly transferring high-resolution details from the original video in a post-processing step. The residual between the neural renderer output and the video defines the detail to transfer, and the amount of the residual to transfer to each layer is determined by the transmittance map τ_t^i:

$$\tau_t^i = 1.0 - Comp_\alpha(\mathcal{L}_t \setminus \{L_t^j \mid j < i\}, o_t \setminus \{j \mid j < i\}) \tag{9}$$

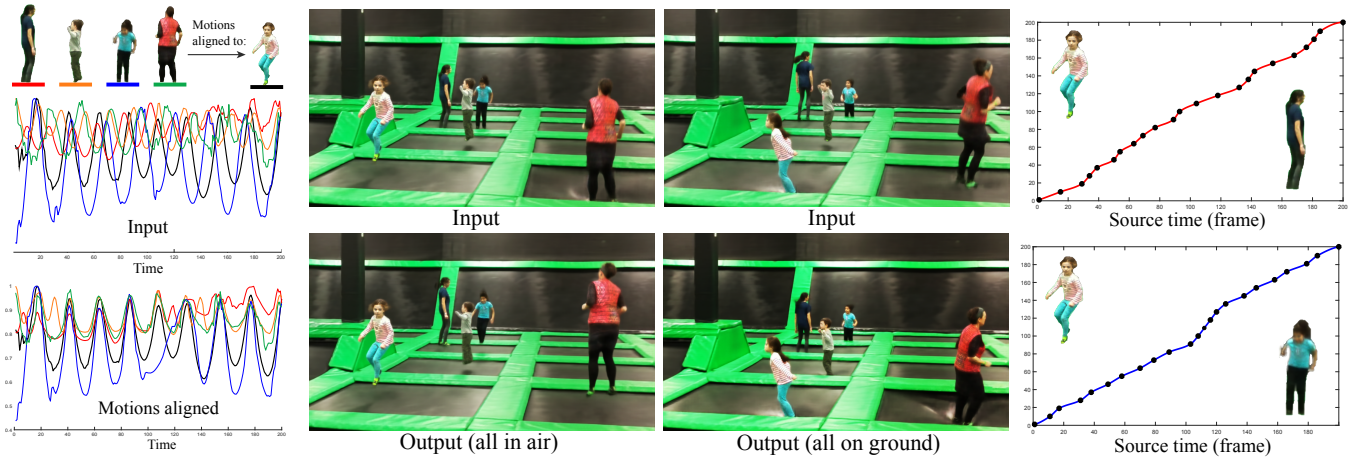
where $Comp_\alpha$ denotes the alpha channel of the composite produced by the neural renderer. The final layer colors are:

$$C_t^i = Cnr_t^i + \tau_t^i(I_t - Comp(\mathcal{L}_t, o_t)) \tag{10}$$

where Cnr is the color produced by the neural renderer. See supplementary material (SM) for visualizations. Given this transfer, the upsampling network needs only to refine the predicted alpha mattes and produce reasonable colors in occluded regions, where ground-truth high-frequency details are not available. For pixels with no occlusion and a predicted alpha value of 1, this high-frequency transfer amounts to using the original RGB pixel.

4.5 Keypoints-to-UVs

Estimating people UV maps from images can be done via off-the-shelf methods (e.g., DensePose [Güler et al. 2018]). However, using such methods directly has two main drawbacks:



(a) Vertical motion **(b) Representative input/output frames** **(c) Representative time warp curves**

Fig. 5. **Automatic motion alignment**. People jumping on trampolines are motion-aligned to the little girl in the front left, such that they all jump in sync. (a) Top: motion signals for the different people in the video, taken as the normalized y-coordinate of the center of mass of each person over time, as computed from each person's tracked keypoints. Bottom: time-warped motion signals, after aligning each one to the motions of the little girl (black line in the plots) using Correlation Optimized Warping [Tomasi et al. 2004] (a variation of Dynamic Time Warping). (b) Each column shows corresponding frames from the input video (top), and from the output, retimed video (bottom). (c) Two estimated time warps, aligning two of the jumpers with the jumps of the front little girl.

(1) *Disoccluded people*: Video retiming effects often result in disocclusions of people who were partially or even fully occluded in the original frames. Because we model the appearance of a person with a single texture map that is learned jointly for the entire video, we can render disoccluded content as long as we can correctly sample from it. Thus, we want to ensure that all UV maps represent the full body of each individual, even in the presence of occlusion. Existing methods, however, provide an estimate only for the visible human regions (Fig. 4(a)).
(2) *Robustness*: Existing methods tend to suffer from erratic errors such as missing body parts, especially in the presence of motion blur, noise or occlusion (see SM for examples). Due to their dense nature, temporally filtering or interpolating UV maps in order to remove such errors is challenging.

To overcome these limitation, we train a neural network that predicts clean full-body UV maps from human keypoints. Keypoint estimators such as AlphaPose [Fang et al. 2017] tend to be more robust, and unlike UV maps, can easily be interpolated and manually corrected. To train our keypoint-to-UV model, we use the *Let's Dance Dataset* [Castro et al. 2018], curated to contain only single-person video frames, and our own filmed video of approximately 10 minutes of a single person doing a variety of poses. We then generate approximately 20K keypoint-UV training examples in which people are fully visible, by running AlphaPose and DensePose on the original frames. We follow a similar training regimen as in [Güler et al. 2018]; see Appendix for further details.

At test time, we can estimate full-body UV maps even in the presence of significant occlusions: we first estimate keypoints using AlphaPose, track the keypoints using PoseFlow [Xiu et al. 2018], and linearly interpolate occluded keypoints. The trained keypoint-to-UV network then processes these keypoints to generate complete UV maps (Fig. 4(b)). In challenging cases, current state-of-the-art keypoint detectors and trackers can still fail due to motion blur, occlusions, or noise. For the results in this paper, we manually fixed these errors using a rotoscoping tool for between zero and 10% of input frames, taking a maximum of 30 minutes for particularly troublesome videos. See SM videos for examples of keypoint edits.

4.6 Camera Tracking

When the input video contains a moving camera (as is the case with all our video examples), we first estimate the camera motion using a feature-based tracking algorithm similar to the one described in [Grundmann et al. 2011]. We model the camera motion at each frame using a homography transformation, which we estimate robustly from matched ORB features [Rublee et al. 2011] between frames. When stabilizing small camera motions or natural hand shake, we compute the homographies between each frame and a single reference frame (which works better than tracking the camera over time), then use them to stabilize the input video. When the video contains large camera motion or substantial panning, we estimate homographies over time between consecutive frames, use them to register all the frames with respect to a common coordinate system, then apply this coordinate system transformation to the background UV map to preserve the original camera motion. Retiming a layer from frame t to $\bar{t}$ is achieved by transforming the layer to the common coordinate system using the transformation at t, then applying the inverse transformation at $\bar{t}$.

5 WHY IT WORKS

As discussed in Section 4.3, our training ensures that specific people are assigned into specific layers (by bootstrapping the predicted alpha mattes with rough people trimaps), whereas effects outside the human trimap region, such as shadows and reflections, are learned incrementally as the network trains. But how does the

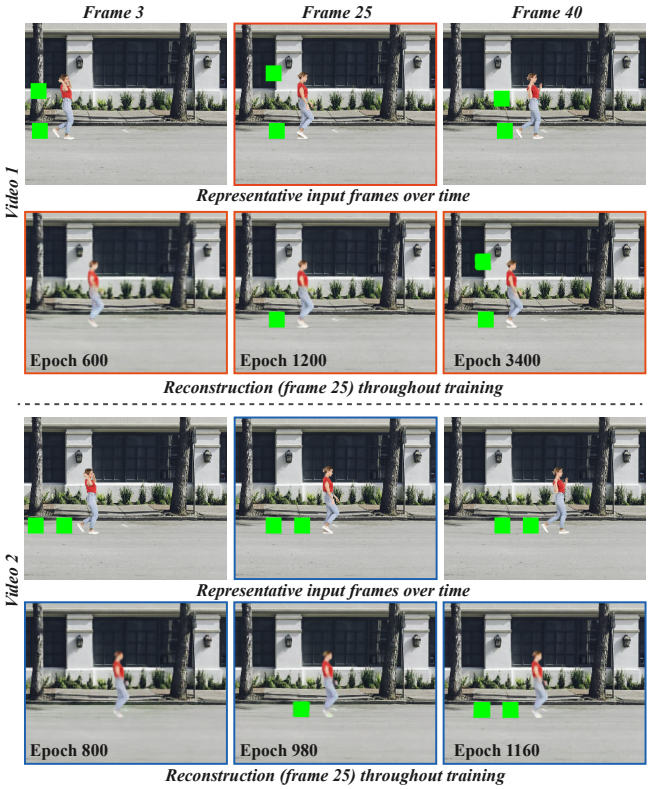


Fig. 6. **Synthetic examples for testing effects of motion correlation and spatial proximity.** In video 1, one square moves smoothly with the person while the other is randomly placed. The smoothly moving square is reconstructed earlier in training than the random square. In video 2, two squares have the same motion but one is closer to the person. The closer square is reconstructed earlier in training.

model manage to associate such scene elements with the correct layers?

Single person and their correlated effects. The reason for this successful assignment stems from properties of CNNs. Specifically, space-time scene elements that are most strongly correlated with the input are reconstructed earlier in training (i.e., require smaller changes to the network weights) than non-correlated elements. To illustrate this, consider the synthetic sequences shown in Fig. 6 (the full videos are available in the SM). A person is superimposed on a fixed background along with two green squares. The first video examines correlations in motion and the second examines correlations in space (proximity). In the first video, one square moves consistently left-to-right behind the person, while the other is randomized around the person. The network first learns to reconstruct the static background and person, then the correlated square, and finally the random square. In the second video, two squares have the same motion, but one is closer to the person. The closer square is reconstructed earlier in training than the farther square.

This behavior is tightly related to the "Deep Image Prior" (DIP) principle [Ulyanov et al. 2018]. An image is represented by the parameters of a CNN by "overfitting" it with gradient descent until the network output matches the image. Importantly, it was shown that fitting a natural image (which contains repetitive patches) takes

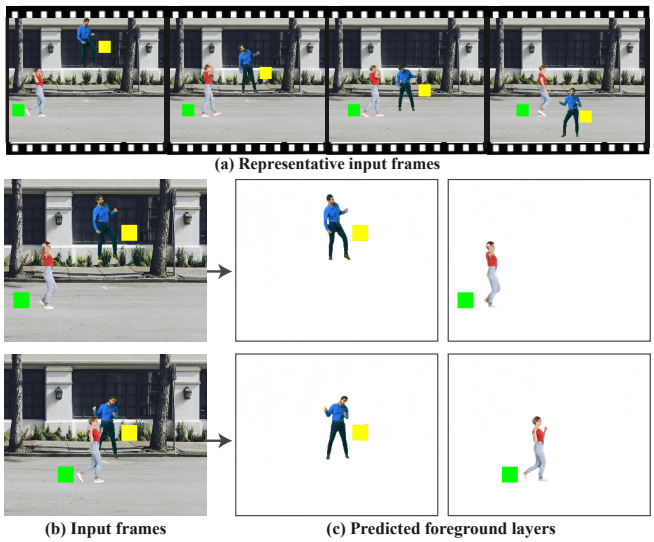


Fig. 7. **A synthetic example demonstrating assignment into layers.** (a-b) A woman (in red) is translating from left to right together with a green square next to her. Similarly, a man (in blue) is oscillating up and down with a yellow square next to him. (c) With gradient descent iterations, each person layer "grabs" more quickly the square that is closest and moves most similarly to it (see Fig. 6). The squares are consistently assigned into each person's layers throughout the video, even in the presence of occlusions between the people and squares.

fewer iterations than fitting a random noise image (or a natural image corrupted by noise). In our case, effects that are closely correlated with a person in space, time, or both, are learned more quickly than other effects. Our synthetic examples and results on natural videos support this observation.

Assigning the correct effects to multiple people. When there are multiple people present in the video, each person layer needs to "grab" the scene elements that are correlated with it. This assignment is achieved via our training regime and the construction of the input to the network. The renderer sees one layer's UV coordinates at a time, so must explain each missing scene element using only the information from a single person (or preselected group of people). Because correlated effects are learned faster than uncorrelated effects, each person layer will "grab" the effects that are most correlated with it before the other layers do. Once an effect is reconstructed in a layer, the reconstruction loss for that effect is minimized and no gradient encourages the effect to be learned in other layers. In practice, this means that loose clothing, shadows, reflections, or other effects that are most strongly correlated with the person creating them, are likely to be decomposed into the correct layer.

Fig. 7 shows such a synthetic, two-person example. The second green square from the synthetic sequence in Fig. 6 is replaced by a second person (a man in blue), who has a yellow correlated square. The motions of the two people are uncorrelated: a man (in blue) oscillates vertically while a woman (in red) translates horizontally. We bootstrap each layer with its respective person's trimap and train the network to reconstruct the video. The network learns to reconstruct the green square in the woman's layer, and the yellow

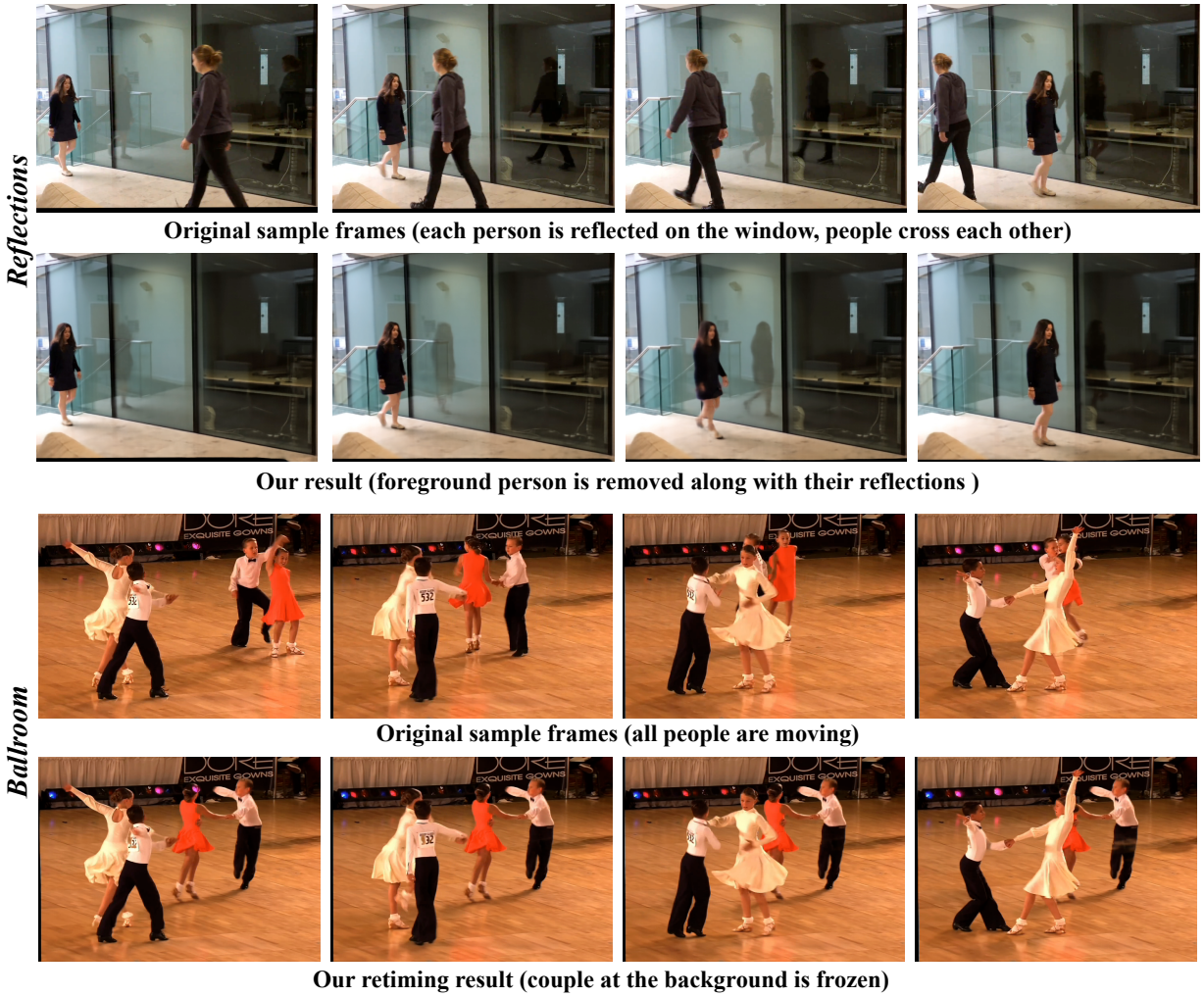


Fig. 8. **Retiming and editing results** for person removal (top) and motion freeze (bottom). For each example (pairs of rows) we show sample frames from the original video (top row) and their corresponding retimed/edited frames (bottom row).

square in the man's layer. Further, the yellow square (and the man) are fully reconstructed, even for frames where they were occluded by the woman or the green square.

Note that if instead of the layered input, the network sees the entire scene at one time (similar to [Thies et al. 2019]), the network learns to explain the missing scene elements using *combinations* of people, not individuals. Artifacts then appear when individuals are retimed to form a new combinations not seen during training (Fig. 11).

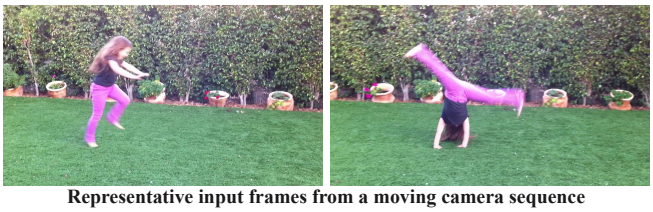
6 RESULTS

We tested our method on a number of real-world videos, most of which are captured by hand-held cellphone cameras (either captured by us or taken from the Web). All the videos depict multiple people moving simultaneously and span a wide range of human actions (e.g., dancing, jumping, running) in complex natural environments. Representative frames from these videos are shown in Figs. 1, 3 and 8, and the full input and output sequences are available in the supplementary material.

We implemented our networks in PyTorch and used the Adam optimizer [Kingma and Ba 2014] with a learning rate of $1e-3$. Depending on video length and number of predicted layers, total training time took between 5 and 12 hours on 2 NVIDIA Tesla P100 GPUs. See Appendix A.1 for further implementation details.

6.1 Layer Decomposition

Several of our layer decompositions are visualized in Fig. 3. For *Ballroom* and *Splash*, we group certain people into one layer; for all other videos, each person is assigned with their own layer. For all the videos, our model successfully disentangles the people into the layers. The layers capture fine details such as loose hair and clothing (e.g., foreground dancer's white dress in *Ballroom*), or objects attached to the people (e.g., children's floaties in *Splash*). This is in spite of initializing our model with a rough people UV map without explicitly representing these elements. This ability of our



Representative input frames from a moving camera sequence

Our result (duplicating the girl in purple, freezing the girl in blue)

Fig. 9. **Cartwheel.** In the original video, the camera is panning. In our edited result, we preserve the original camera motion, while duplicating the girl who performs the cartwheel and freezing the girl in blue so that she remains in the frame rather than moving offscreen.

model to accurately segment the people regions is also illustrated more closely in Fig. 10.

Furthermore, the predicted layers successfully correlated the people with other nontrivial visual time-varying scene elements that are related to them—for example, shadows cast on the floor by the different dancers (*Ballroom*), complicated reflections of two people crossing each other (*Reflections*), surface deformation (*Trampoline*), or water splashes (*Splash*) caused by people's motion.

6.2 Retiming and Editing Results

With the predicted layers in hand, we produced a variety of retiming and editing effects via simple operations on the layers. We show several such retiming results in Fig. 1, Fig. 5, and Fig. 8 (full videos are available in the SM).

In Fig. 1 and Fig. 5, we have multiple people performing a similar action, but their motions are not in sync. In Fig. 1, the children jump into the pool one after the other, whereas in Fig. 5 the periodic motions of the people jumping are independent. In both examples, we retime the people to align their motions. For *Splash*, several alignment points defined manually were sufficient to align the children's jumps. In *Trampoline*, because of the periodic nature of the motion, the alignment is performed automatically using Correlation Optimized Warping [Tomasi et al. 2004] (a variation of Dynamic Time Warping). Notice how all the time-varying scene elements caused by the people—water splashes as they hit the water, trampoline deformations as they bounce on it—follow automatically with them in our retimed results.

We can also use our method to "freeze" people at a certain point in time, while letting other people move as in the original video. This allows viewers to focus their attention on the moving people while ignoring the rest of the motions in the original video. In *Ballroom* (Fig. 8), we freeze the dancing couple in the back throughout the video, while the couple in front keeps moving. Here too, the shadows and reflections on the floor move realistically with the moving couple, while the shadows of the background couple remain static. Furthermore, disoccluded regions of the back couple are realistically rendered.

In *Kids Running* (supplementary material), we show how our model can scale to multiple layers (8) to produce complex retiming effects involving many people. We retime the original video, where the kids are crossing the faint finish line on the ground at different times, to produce a 'photo-finish' video, where all the kids cross the finish line together. We achieve this result by slowing down the layers of the children that run offscreen. Even though this sequence involves many individuals, our model is able to obtain clean mattes for each child. In addition, occluded people and large occluded regions in the background are realistically inpainted, all while handling significant motion blur that exists in the input video.

As mentioned, in addition to retiming effects, our method can also support easy removal of people in video—a byproduct of the layered representation we use. In *Reflections*, we demonstrate person removal in a video containing two people crossing paths in front of a window. Here our model manages to perform several nontrivial tasks: it completely disoccludes the person walking in the back; it associates each person properly with their reflection and shadow; and it disentangles the two reflections when they overlap (despite the fact that none of these elements are represented explicitly by the model). Consider generating such a result with a traditional video editing pipeline: the reflections would have to be tracked along with the people to perform proper removal; the person in the back would have to be manually inpainted at the point where they are occluded by the person in front. The advantage of our method is that by merely inputting UVs for each person in a separate layer, and turning those layers 'on and off', we can achieve the same result with significantly less manual work.

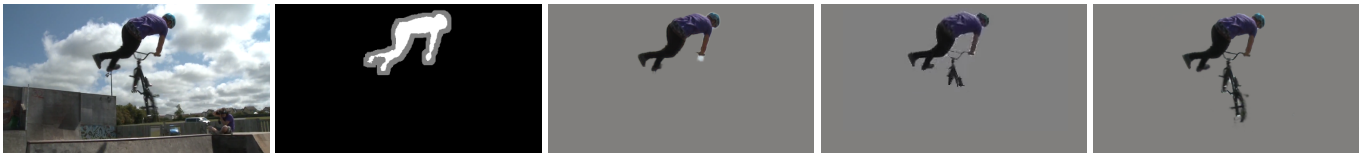
In Fig. 9, we show an example where the original camera motion is preserved in our retimed result (see Section 4.6) – the girl on the left is duplicated with a short time offset between each of her copies, and the girl on the right is frozen, all while the camera is panning as in the original video.

6.3 Comparisons

We compare our method to state-of-the-art learning-based image matting [Hou and Liu 2019] and Double-DIP [Gafni et al. 2020] in Fig. 10. Matting methods are unsuitable for retiming because they fail to capture desired effects outside of the trimap region. To capture such effects using matting would require annotating the trimap to represent these regions. Our method, however, is not strictly bound by the trimap, as we use it to bootstrap our training only in the initial epochs, after which the optimization is allowed to grow the matte to encompass regions beyond the trimap border. Thus, we can capture complex effects *without explicitly representing them in the trimap*, merely by representing the human region coarsely using automatically generated trimaps. While Double-DIP is similarly unbound by the trimap region, it fails to segment the entire bicycle.

6.4 Ablations

We ablate components of our method, specifically the keypoint-to-UV network and layer decomposition.



(a) Original Frame (b) Trimap for (c) and (e) (c) Image Matting [Hou and Liu 2019] (d) Double DIP [Gandelsman et al. 2019] (e) Ours

Fig. 10. **Comparison with matting and Double-DIP.** We obtain trimaps from a mask of the person's UV; the gray pixels represent a dilated region around this mask for which the matching loss is ignored. Matting is insufficient for our retiming task because it fails to capture correlated effects that are not included in the input trimap (e.g. the bicycle). Unlike matting, our final model prediction is able to move beyond the bounds provided by the coarse trimap to produce a more accurate segmentation mask that includes correlated regions. While Double-DIP is not limited by trimap bounds, it fails to segment the entire bicycle.

(a) Non-layered Model

(b) Our Layered Model

Fig. 11. **Layered vs. non-layered neural rendering.** Removing the layer decomposition component of our model results in a lack of generalization ability. While both models are able to reconstruct the original video, the non-layered model struggles to disentangle people and their correlations (reflection in right column is not fully removed when the person is removed).

6.4.1 Keypoint-to-UV network vs. DensePose. We compare layer decomposition results from using our keypoint-to-UV network to predict UV maps versus using DensePose outputs directly (Fig. 4). In the case of partial occlusion, our UV maps manage to complete the occluded regions of the person, while the DensePose UVs fail to complete the face and the left arm. In a more extreme case where the person is fully occluded, but their reflection is visible, it is crucial to generate a UV map for that person not only for the purposes of disocclusion, but because any visible effects such as their reflection or shadows will be incorrectly reconstructed in the layer of the visible person (as the fully occluded UV will be indistinguishable from the background-only UV). Thus we disocclude UVs by using keypoints as an intermediate representation that allows for more robust estimation and straightforward interpolation.

6.4.2 Layered vs. Non-Layered Neural Rendering. We compare our method to a non-layered model, i.e., a model that takes as input a single sampled texture map representing all the people in the frame as well as the background, and directly outputs an RGB reconstruction of the frame. This baseline follows the deferred neural rendering approach [Thies et al. 2019] .

Fig. 11 shows a reconstruction and editing result produced by the non-layered model. As can be seen, the non-layered model can reconstruct the original frames fairly well despite the missing information and noise in the UV maps. This aligns with the results demonstrated in [Thies et al. 2019]. However, when editing is performed, the non-layered model performs poorly and is unable to generalize to new compositions of people, as evident by the partial reflection artifact. The main reason is twofold: (i) To produce editing effects, the model is required to generalize to new UV compositions of people in configurations that were never seen during training; thus it struggles to produce realistic-looking results based solely on L1 reconstruction loss. Our approach avoids this generalization issue because editing is performed as post-processing on the predicted layers—the same outputs produced during training. (ii) When the input to the model is a composition of all the people in the frame rather than separated UVs, the model can easily reconstruct the original frame without necessarily capturing meaningful correlations since it is not required to disentangle separately moving parts of the scene. This can be seen where the non-layered model struggles to learn the correct relationships between the different people and their reflections.

Another benefit of the layered representation is that we can transfer high-resolution details from the input video to each layer, as described in Section 4.4. Because the non-layered model produces the retimed result directly, there is no opportunity to transfer detail from the input video prior to retiming. Achieving comparable quality with the non-layered model would likely require extensive training time and additional loss functions.

6.5 Limitations

While our system manages to decompose scenes quite well—even surprisingly so—in some scenes the results may not be perfect. Here we point out some of the limitations that we have observed.

For a particularly difficult section of the *Splash* video, part of the large water splash caused by the child who is underwater in *Layer 1* is incorrectly placed in *Layer 2* (Fig. 12). However, this error can be fixed by manually editing the initial trimap masks for Layer 1 to include the entire splash (top right, Fig. 12). Such editing can be done very quickly when needed, requiring only rough marking on a few of the frames (a single trimap in this example).

Another type of artifact that may occur is when the scene includes time-varying background elements that are not correlated with any of the people of interest in the video. For example, since we assume a static background, the colorful blinking lights in the *Ballroom* video

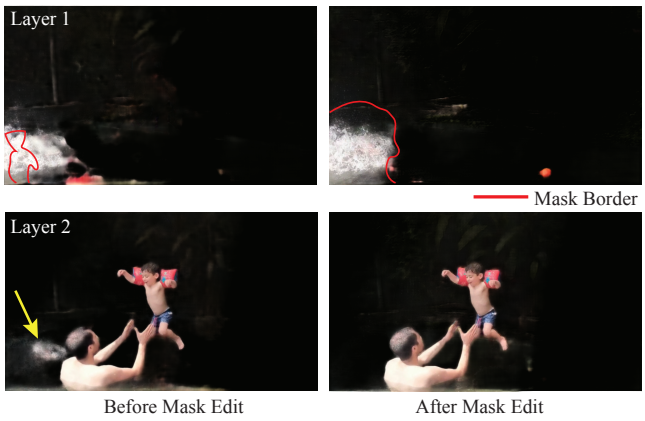


Fig. 12. **User-guided trimaps.** In some particularly difficult segments, the assignment of effects to layers may be incorrect, e.g., some parts of the splash caused by the child who is underwater are incorrectly assigned to Layer 2 (yellow arrow). The user can correct such errors by simple manual editing of the initial trimaps. Here, we expand the initial trimap to include the entire splash region (red outline); this was done for only one layer of one frame, and then duplicated for a segment of the video.

must be included in one of the foreground layers. As can be seen in Fig. 3, the purple light appears in the front couple's layer, making it impossible to retime them separately. This artifact could be remedied by adding a separate layer to represent the dynamic background elements in addition to the layers for the dancing couples.

7 CONCLUSION

We have presented a system for retiming people in video, and demonstrated various effects, including speeding up or slowing down different individuals' motions, or removing or freezing the motion of a single individual in the scene. The core of our technique is learned layer decomposition in which each layer represents the *full* appearance of an individual in the video — not just the person itself but also all space-time visual effects correlated with them, including the movement of the individual's clothing, and even challenging semi-transparent effects such as shadows and reflections.

We believe that our layered neural rendering approach holds great promise for additional types of synthesis techniques, and we plan to also generalize it to other objects besides people, and to expand it to other non-trivial post-processing effects, such as stylized rendering of different video components.

ACKNOWLEDGMENTS

We thank the friends and family that appeared in our videos. The original *Ballroom* video belongs to Desert Classic Dance. This work was funded in part by the EPSRC Programme Grant Seebibyte EP/M013774/1.

REFERENCES

Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. 2019. Deep Video-Based Performance Cloning. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 219–233.

Aseem Agarwala, Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. 2005. Panoramic Video Textures. In *SIGGRAPH*.

Jean-Baptiste Alayrac, João Carreira, and Andrew Zisserman. 2019a. The Visual Centrifuge: Model-Free Layered Video Representations. In *CVPR*.

Jean-Baptiste Alayrac, Joao Carreira, Relja Arandjelovic, and Andrew Zisserman. 2019b. Controllable Attention for Structured Layered Video Decomposition. In *ICCV*.

Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. 2009. Video SnapCut: robust video object cutout using localized classifiers. *TOG* (2009).

Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. 2010. Video Tapestries with Continuous Temporal Zoom. *SIGGRAPH* (2010).

Eric P Bennett and Leonard McMillan. 2007. Computational time-lapse video. In *ACM SIGGRAPH 2007 papers*. 102–es.

Daniel Castro, Steven Hickson, Patsorn Sangkloy, Bhavishya Mittal, Sean Dai, James Hays, and Irfan Essa. 2018. Let's Dance: Learning From Online Dance Videos. In *eprint arXiv:2139179*.

Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. 2019. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision*. 5933–5942.

Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David Salesin, and Richard Szeliski. 2002. Video matting of complex scenes. In *SIGGRAPH*.

Abe Davis and Maneesh Agrawala. 2018. Visual Rhythm and Beat. *ACM Trans. Graph.* 37, 4 (2018), 122–1.

Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. 2017. RMPE: Regional Multi-person Pose Estimation. In *ICCV*.

Oran Gafni, Lior Wolf, and Yaniv Taigman. 2020. Vid2Game: Controllable Characters Extracted from Real-World Videos. In *ICLR*.

Yossi Gandelsman, Assaf Shocher, and Michal Irani. 2019. "Double-DIP": Unsupervised Image Decomposition via Coupled Deep-Image-Priors. In *CVPR*.

Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. 2008. Video Object Annotation, Navigation, and Composition. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA) *(UIST âĂŹ08)*. Association for Computing Machinery, New York, NY, USA, 3âĂŞ12. https://doi.org/10.1145/1449715.1449719

Matthias Grundmann, Vivek Kwatra, and Irfan Essa. 2011. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR 2011*. IEEE, 225–232.

Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7297–7306.

Qiqi Hou and Feng Liu. 2019. Context-Aware Image Matting for Simultaneous Foreground and Alpha Estimation. In *ICCV*.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *CVPR*.

Njegica Jojic and B.J. Frey. 2001. Learning flexible sprites in video layers. In *CVPR*.

Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F Cohen. 2015. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.

H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, M. Nießner, P. Péerez, C. Richardt, M. Zollhöfer, and C. Theobalt. 2018. Deep Video Portraits. *ACM Transactions on Graphics 2018 (TOG)* (2018).

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *ICLR* (2014).

M. Pawan Kumar, Philip H. S. Torr, and Andrew Zisserman. 2008. Learning Layered Motion Segmentations of Video. *IJCV* (2008).

Shuyue Lan, Rameswar Panda, Qi Zhu, and Amit K Roy-Chowdhury. 2018. FFNet: Video fast-forwarding via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6771–6780.

Jessica Lee, Deva Ramanan, and Rohit Girdhar. 2020. MetaPix: Few-Shot Video Retargeting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJx1URNKwH

Yin Li, Jian Sun, and Heung-Yeung Shum. 2005. Video Object Cut and Paste. In *SIGGRAPH*.

Lingjie Liu, Weipeng Xu, Michael Zollhöfer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. 2019. Neural Rendering and Reenactment of Human Actor Videos. *ACM Trans. Graph.* 38, 5, Article 139 (Oct. 2019), 14 pages. https://doi.org/10.1145/3333002

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, and et al. 2018. LookinGood: Enhancing Performance Capture with Real-Time Neural Re-Rendering. *ACM Trans. Graph.* 37, 6, Article 255 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275099

James McCann, Nancy S Pollard, and Siddhartha Srinivasa. 2006. Physics-based motion retiming. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 205–214.

Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. 2019. Neural rerendering in the wild.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6878–6887.
Ajay Nandoriya, Elgharib Mohamed, Changil Kim, Mohamed Hefeeda, and Wojciech Matusik. 2017. Video Reflection Removal Through Spatio-Temporal Optimization. In *ICCV*.
Yair Poleg, Tavi Halperin, Chetan Arora, and Shmuel Peleg. 2015. Egosampling: Fast-forward and stereo for egocentric videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4768–4776.
Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 253âĂŞ259. https://doi.org/10.1145/964965.808606
Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. 2008. Nonchronological video synopsis and indexing. *IEEE transactions on pattern analysis and machine intelligence* 30, 11 (2008), 1971–1984.
Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*. Ieee, 2564–2571.
Michel Silva, Washington Ramos, Joao Ferreira, Felipe Chamone, Mario Campos, and Erickson R Nascimento. 2018. A weighted sparse sampling and smoothing frame transition approach for semantic fast-forward first-person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2383–2392.
V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and Zollhöfer. 2019. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.
Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the Boundaries of View Extrapolation with Multiplane Images. In *CVPR*.
Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Trans. Graph.* 38, 4 (2019), 66:1–66:12.
Giorgio Tomasi, Frans Van Den Berg, and Claus Andersson. 2004. Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *Journal of Chemometrics: A Journal of the Chemometrics Society* 18, 5 (2004), 231–241.
Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9446–9454.
John Wang and Edward Adelson. 1994. Representing Moving Images with Layers. *IEEE Trans. on Image Processing* (1994).
Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. 2005. Interactive Video Cutout. *TOG* (2005).
Yonatan Wexler, Eli Shechtman, and Michal Irani. 2007. Space-Time Completion of Video. *PAMI* (2007).
Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. 2018. Pose Flow: Efficient Online Pose Tracking. In *BMVC*.
Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. 2017. Deep Image Matting. In *CVPR*.
Tianfan Xue, Michael Rubinstein, Ce Liu, and William T. Freeman. 2015. A Computational Approach for Obstruction-Free Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (2015).
Innfarn Yoo, Michel Abdul Massih, Illia Ziamtsov, Raymond Hassan, and Bedrich Benes. 2015. Motion retiming by using bilateral time control surfaces. *Computers & Graphics* 47 (2015), 59–67.
Feng Zhou, Sing Bing Kang, and Michael F Cohen. 2014. Time-mapping using space-time saliency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3358–3365.
Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis using Multiplane Images. In *SIGGRAPH*.
Yipin Zhou, Zhaowen Wang, Chen Fang, Trung Bui, and Tamara Berg. 2019. Dance dance generation: Motion transfer for internet videos. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 0–0.
C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality video view interpolation using a layered representation. *TOG*.

A APPENDIX

A.1 Implementation Details

Neural texture. We use a neural texture with 16 channels, where each body part and the background are represented by an atlas with 16×16 pixel dimensions. As in the DensePose work, we adopt the SMPL representation, which uses 24 body parts. Thus the dimensions of our neural texture, for a video with N people, is $16 \times 16 \times 16 * (24N + 1)$. Empirically we found that increasing the texture resolution did not improve results.

Training on low-resolution. For each video, we first train the neural renderer and neural texture for 2500 epochs at a video size of 448×256 (or 352×256 for the *Ballroom* sequence only). The input to the neural rendering network is the sampled neural texture concatenated with a map representing the person ID at each pixel (or 0 for background). We apply brightness and spatial jittering to the video frames as data augmentation. Brightness jittering is turned off after 400 epochs to allow the network to faithfully reconstruct the original video. We also employ curriculum learning, where we train only on the easier half of the frames for the first 1k epochs. We rank difficulty by computing the IoU of every pair of layers' positive trimap regions, using distance of bounding box centers as a tie-breaker. The goal of this curriculum is to withhold the more difficult frames that contain overlapping people, because of the greater ambiguity in learning correlations.

Upsampling low-resolution results. After training the neural rendering module, we obtain a high-resolution result by freezing the trained parameters and training an additional upsampling network. This lightweight upsampling network is trained for 500 epochs with only L1 reconstruction loss, sampling random 256×256 crops for upsampling due to memory constraints. The final output of the upsampling network has dimensions double the size of the low-resolution output. We apply the detail transfer step described in Section 4.4 to these final outputs.

A.2 Network Architectures

In all networks, zero-padding is used to preserve the input shape. 'bn' refers to batch normalization. 'in' refers to instance normalization. 'convt' refers to convolutional transpose. 'leaky' refers to leaky relu with slope -0.2. 'skip$k$' refers to a skip connection with layer k. 'resblock' denotes a residual block consisting of conv, instance norm, relu, conv, instance norm.

A.2.1 Neural renderer. The neural renderer architecture is a modified pix2pix network [Isola et al. 2017]:

	layer type(s)	out channels	stride	activation
1	4×4 conv	64	2	leaky
2	4×4 conv, bn	128	2	leaky
3	4×4 conv, bn	256	2	leaky
4	4×4 conv, bn	256	2	leaky
5	4×4 conv, bn	256	2	leaky
6	4×4 conv, bn	256	1	leaky
7	4×4 conv, bn	256	1	leaky
8	skip5, 4×4 convt, bn	256	2	relu
9	skip4, 4×4 convt, bn	256	2	relu
10	skip3, 4×4 convt, bn	128	2	relu
11	skip2, 4×4 convt, bn	64	2	relu
12	skip1, 4×4 convt, bn	64	2	relu
13	4×4 conv	4	1	tanh

A.2.2 Upsampling network. The upsampling network predicts a residual image for each bilinearly upsampled RGBA layer predicted

by the neural renderer. The network input is the bilinearly upsampled (to the desired output size) concatenation of (1) the predicted low-resolution RGBA, (2) the sampled texture input, and (3) the final feature maps output by the neural renderer preceding the RGBA output layer. The RGBA outputs of the upsampling network are then composited according to o_t. The upsampling network architecture is as follows:

	layer type(s)	output channels	stride	activation
1	3×3 conv, in	64	1	relu
2	3×3 resblock	64	1	relu
3	3×3 resblock	64	1	relu
4	3×3 resblock	64	1	relu
5	3×3 conv	4	1	none

A.2.3 Keypoints-to-UVs. The keypoint-to-UV network is a fully convolutional network that takes in an RGB image of a skeleton and outputs a UV map of the same size. The architecture is the same as the neural renderer architecture, with the exception of the final layer, which is replaced by two heads: 1) a final convolutional layer with 25 output channels to predict body part and background classification, and 2) a convolutional layer with 48 output channels to regress UV coordinates for each of the 24 body parts. As in the DensePose work [Güler et al. 2018], we train the body part classifier with cross-entropy loss and train the predicted UV coordinates with L1 loss. The regression loss on the UV coordinates is only taken into account for a body part if the pixel lies within the specific part, as defined by the ground-truth UV map.