Hashing Neural Video Decomposition with Multiplicative Residuals in Space-Time

Cheng-Hung Chan Cheng-Yang Yuan Cheng Sun Hwann-Tzong Chen
National Tsing Hua University, Taiwan

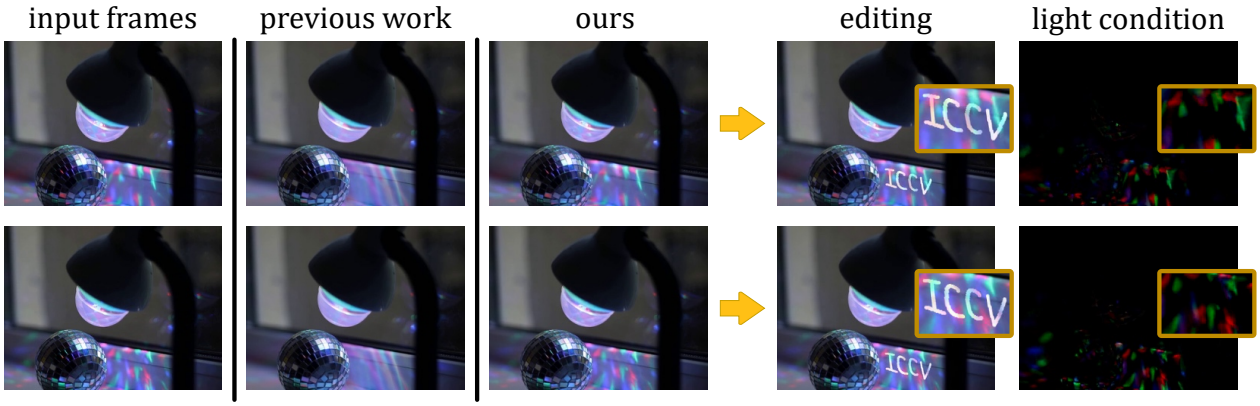

Figure 1: **Integration of layer-based video editing and spatiotemporal lighting rendering.** Our approach decomposes an input video into layers with multiplicative residuals that characterize complex spatiotemporal lighting variations. Our method efficiently fuses the user-edited components with the expected lighting conditions to produce high-quality video output.

Abstract

We present a video decomposition method that facilitates layer-based editing of videos with spatiotemporally varying lighting and motion effects. Our neural model decomposes an input video into multiple layered representations, each comprising a 2D texture map, a mask for the original video, and a multiplicative residual characterizing the spatiotemporal variations in lighting conditions. A single edit on the texture maps can be propagated to the corresponding locations in the entire video frames while preserving other contents' consistencies. Our method efficiently learns the layer-based neural representations of a 1080p video in 25s per frame via coordinate hashing and allows real-time rendering of the edited result at 71 fps on a single GPU. Qualitatively, we run our method on various videos to show its effectiveness in generating high-quality editing effects. Quantitatively, we propose to adopt feature-tracking evaluation metrics for objectively assessing the consistency of video editing. Project page: https://lightbulb12294.github.io/hashing-nvd/

1. Introduction

Unlike image editing, video editing involves modeling the frame-to-frame relationships and addressing temporal variations such as motion and illumination changes. The task of video editing becomes challenging with the added dimension of time for a user who attempts to apply edits to a video while ensuring consistency across all frames. It is more convenient if we can handle the spatiotemporally varying components and reduce video editing to image editing—The user thus only needs to do edits on images with ease, and the editing results can seamlessly propagate to the entire video.

To achieve this goal, we may consider incorporating effective representations that can model and reconstruct the static and dynamic information in videos. Furthermore, for practical concerns, the algorithm must be efficient enough in modeling and rendering to support interactive editing. Recent work on video decomposition has proposed to employ neural-based representations, such as layered neural atlases [16] and deformable sprites [39], to enable the conversion between the space-time video domain and 2D texture domain for editing. Despite their successes in showcasing impressive video editing effects, we notice that they often require longer training time or restrict to limited frame resolution. For example, it takes more than ten hours to derive the layered neural atlases [16] from a 100-frame 480p video. Deformable sprites [39] are relatively fast to derive but require more computation resources (over 24GB of GPU memory for a 480p video) and thus are unsuitable for editing

Figure 2: **The procedure of layer-based video editing.** Our model allows users to apply edits to the extracted texture for rendering the edited video.

high-resolution videos. We address the computation issue by incorporating hash grid encoding [24] into our framework and achieve fast training and rendering for high-resolution videos. Moreover, we introduce the multiplicative residual estimator to model the lighting variations across video frames, which can improve the reconstruction quality and allow illumination-aware editing unachievable by prior work, as shown in Fig. 1.

We summarize the contributions of this work as follows:

1. This work is the first to consider spatiotemporally varying lighting effects for layered-based video editing. The proposed multiplicative-residual estimator effectively decomposes the lighting conditions from the video without supervision. Our method can produce better-quality videos by fusing the edits with expected illuminations.
2. Our approach is efficient in both training and rendering. Compared with prior work, the proposed method improves the training time with fewer resources and thus enables training on higher-resolution or longer videos. The trained model can achieve fast video rendering via hashing-based coordinate inference. It takes about 40 minutes to train with a 1080p video of 100 frames on a single 3090 Ti GPU. The inference speed for rendering an edited video is 71 fps for 1080p resolution, allowing real-time video editing.
3. The experimental results demonstrate appealing video edits in various challenging contexts, such as modifying the texture of moving objects, handling occlusion, and manipulating camera motion, where all can be fused with vivid lighting and shading for better effects that are not easy to achieve by prior work.

2. Related Work

Video decomposition. Decomposing a video into layered representations is a long-standing video analysis approach in computer vision since the seminal work by Wang & Adelson [33]. Similar ideas have been continually revisited under different contexts with the development of new techniques. A typical formulation is motion segmentation that decomposes the video by motion [6, 15, 17, 35]. Video matting, on the other hand, focuses more on separating the alpha matte of the foreground object from the background for blending [9, 22, 30, 36]. Recently, neural-based methods have shown to be effective in estimating layered representations for video segmentation and video editing [3, 14, 16, 38, 39]. In this work, we also adopt neural networks to derive layered representations from videos. Further, inspired by the pioneering work in visual tracking for handling illumination changes [10], we incorporate a new multiplicative residual representation to model the lighting variations for illumination-aware video editing.

Video editing. Layered representations can benefit video editing in various ways. For example, the layered representations can serve as a visual proxy for intuitive video editing [16] or can be used to create the retiming effects [21]. Editing can be more easily done on a unified texture map built from the video's background, such as background mosaics [7], tapestries [4], and layered neural atlases [16]. Building a unified texture map for the foreground object is also useful, *e.g.*, flexible sprites [15], unwrap mosaics [28], and deformable sprites [39]. Another type of decomposition is to derive temporally-consistent intrinsic components from videos [27, 20] so that the edits can be performed on the reflectance for recoloring or texture transfer. Recent deep-learning-based methods mainly address a single task of video editing, *e.g.*, video style transfer [34, 19, 18, 12, 29]. or category-specific GAN-based video editing [2, 25, 32, 37]. Regarding achieving consistent video editing, it is crucial to know the temporal correspondences [13, 5] so typical animation techniques like Rotoscoping [1] can be applied.

3. Approach

We formulate the problem of consistent video decomposition and editing as follows: The goal is to decompose an input video into multiple layers, where one layer comprises the background, and each of the remaining layers represents a foreground object. We aim to facilitate user-friendly and intuitive editing of video components on time-independent static texture maps while ensuring the quality and consistency of the edited results in the output video. Fig. 2 illustrates an editing procedure. The model may rely on additional information obtained during pre-processing, such as backward and forward optical flow and rough object masks, to improve the training efficiency and consistency.

3.1. Layer-Based Video Decomposition

A video of a scene typically contains multiple objects with distinct motions, appearances, shapes, and other attributes. Achieving consistent editing on the video scene and objects can be challenging. An efficient way to group objects is by decomposing the video into different motion layers. Static objects can be considered part of the scene's background, while only a few foreground objects move differently. Editing the objects on decomposed layers becomes more intuitive and straightforward since we can modify each object separately without touching the others. We decompose videos into layers using backward and forward optical flow and coarse object masks, resulting in $N+1$ layers where layer N represents the background, and the rest is for foreground objects. Each layer n comprises a predicted mask α_n across different frames, a time-independent texture map M_n, and the multiplicative residual $\ell_{n,t}$ characterizing the illumination for each frame t built on the texture map. Fig. 3 shows an overview of our model.

Layer hierarchy. Our alpha network $\mathcal{A}$ predicts the object mask probability for every layer. Rather than simply normalizing the layer probabilities of each pixel and directly using them as layer masks, we build a front-to-back hierarchy. That is, for layer n, its probability of object mask is computed by

$$\alpha_n = a_n \cdot \prod_{i=n+1}^{N} (1 - a_i)\,, \tag{1}$$

where $a \in [0, 1]$ is the raw output of $\mathcal{A}$ with a_N fixed and equal to 1. This hierarchy assumes that objects follow a strict front-to-back ordering, which also implies that layer N represents the background. Here, we specifically focus on scenes with only one foreground object. That is, $N = 1$. We show an example of decomposing multiple foreground objects in the supplementary material.

3.2. Texture Mapping

We associate each object with its own texture, such that modifying a single pixel in the texture will impact the object's appearance in the rendered video scene. We translate the spatiotemporal locations in the video into 2D UV coordinates for different layers by a mapping network $\mathcal{M}$. This allows us to sample colors on a texture with the translated coordinates. We follow the approach of coordinate-based neural rendering [16, 23] and use the multi-layer perceptrons as our coordinate translator instead of a textured grid.

Note that the appearance of an object in the scene may change throughout the video due to camera motion and illumination variations. We will explain how we address this issue with the proposed multiplicative residual in Sec. 3.3.

The textures can be further obtained by grid-sampling the texture network $\mathcal{T}$. Once the texture is extracted, the texture network can be replaced by the extracted texture using UV coordinates from the mapping network, achieving the same effect via bilinear interpolation. This process enables users to modify and apply the extracted texture to the entire video.

3.3. Multiplicative-Residual Estimator

In order to apply a single modification to all video frames, the texture must remain constant throughout the entire video. However, the color at the same location on an object may change over time due to variations in lighting conditions, resulting in low reconstruction quality or noisy textures. To address this issue and ensure high-quality reconstruction, we have developed a new method, *multiplicative-residual estimator*, to achieve constant texture across all frames while preserving illumination variations.

With this design, we can also synthesize illumination coefficients not present in the original video at specific time points, allowing us to manipulate camera motion. Note that the prediction of the residual estimator is sharp. Therefore, we have designed losses to constrain the seen and unseen areas, as will be described in Sec. 3.5. The multiplicative-residual estimator predicts the illumination coefficient of each color channel on the texture map, which is multiplied by each channel to obtain the shaded color.

To provide more insight into our decision to use a multiplicative residual instead of an additive residual, we examine the process of diffuse shading:

$$C = \rho L\,, \tag{2}$$

where ρ is the diffuse albedo term determined by surface material and L is the lighting term determined by the environment. When the lighting condition changes or a shadow is cast onto the surface, the new shaded color then becomes $C' = \rho \cdot L'$. With multiplicative residual, we can reproduce C' by scaling C with $\frac{L'}{L}$. In contrast, we have to add C by $\rho(L' - L)$ if we use additive residual. One can observe that the additive residual is more difficult to model as it also depends on the diffuse albedo term. We detail the practical result of the two choices in Sec. 4.6.

3.4. Network Architecture

We design our model as an end-to-end framework trained via self-supervision. We take a spatiotemporal coordinate $p = (x, y, t)$ in the video as the model input and predict the reconstructed color of the corresponding pixel. Fig. 3 shows the pipeline of our model, which includes a mapping network, a texture network, and a multiplicative-residual estimator for each layered representation, with a shared alpha network generating the soft masks for different layers.

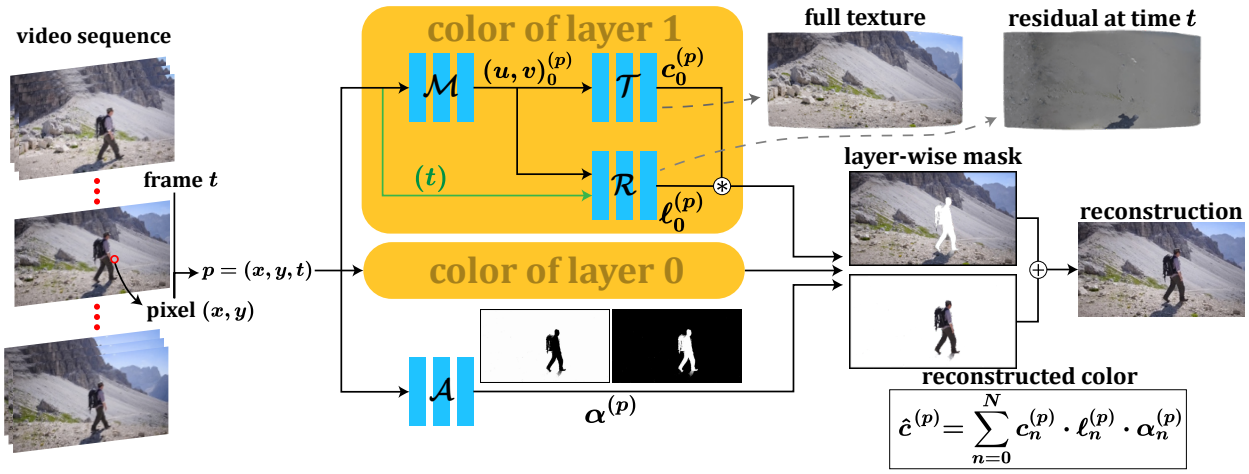

Figure 3: **Model pipeline.** Our model takes a video coordinate $p = (x, y, t)$ as input and decomposes the video into multiple layers. A representation for layer n is modeled by a mapping network $\mathcal{M}$, a texture network $\mathcal{T}$, and a multiplicative-residual estimator $\mathcal{R}$; the three modules jointly convert a video coordinate p into layered color $c_n^{(p)}$ and lighting coefficient $l_n^{(p)}$. An alpha network $\mathcal{A}$ also takes p as input and predicts soft object masks $\boldsymbol{\alpha}^{(p)}$ for each layer. The final video color is reconstructed by the bottom-right equation. The multiplicative residuals are critical to our model for handling illumination variations.

Mapping network. Following the design choice of recent coordinate-based approaches [16, 23], we build our network architecture with multi-layer perceptrons. For each layer n, we pass a spatiotemporal coordinate p through a mapping network to translate the video coordinate into a time-independent 2D texture-map coordinate (u, v) by

$$(u, v)_n^{(p)} = \mathcal{M}_n(p), \quad (u, v)_n^{(p)} \in [-1, 1]^2, \tag{3}$$

where $\mathcal{M}_n$ is the mapping network of layer n.

Texture network and multiplicative-residual estimator. After obtaining the UV coordinates, the texture network takes the UV coordinates $(u, v)_n^{(p)}$ and produces the corresponding color at each position, while the multiplicative-residual estimator uses both the UV coordinates and the time t as inputs to predict the corresponding illumination coefficients at each position:

$$\begin{aligned} c_n^{(p)} &= \mathcal{T}_n\left((u, v)_n^{(p)}\right), \quad c_n^{(p)} \in [0, 1]^3, \\ \ell_{n,t}^{(p)} &= \mathcal{R}_n\left((u, v)_n^{(p)}, t\right), \quad \ell_n^{(p)} \in \mathbb{R}_+^3, \end{aligned} \tag{4}$$

where $\mathcal{T}$ is the texture network and $\mathcal{R}$ is the multiplicative-residual estimator.

Alpha network. We also predict an object mask for each layer to indicate the layer to which each pixel belongs:

$$\boldsymbol{\alpha}^{(p)} = \mathcal{A}(p), \quad \boldsymbol{\alpha}^{(p)} \in [0, 1]^{N+1}, \tag{5}$$

where $\alpha_n^{(p)}$ is the probability that pixel p belongs to layer n, *i.e.*, $\sum_{n=0}^{N} \alpha_n^{(p)} = 1$. The derivation of the object mask is detailed in Sec. 3.1.

Pixel color reconstruction. After obtaining the information described above, we can reconstruct the pixel color at position p as

$$\hat{c}^{(p)} = \sum_{n=0}^{N} c_n^{(p)} \cdot \ell_n^{(p)} \cdot \alpha_n^{(p)}. \tag{6}$$

Inspired by [24], we use hash grid encoding for all networks except for the mapping network, as texture coordinates are expected to be smooth. This approach is adopted to facilitate better convergence during training.

3.5. Loss Terms

Losses inherited from previous work. We incorporate several losses from Layered Neural Atlases [16] to facilitate the training process. These losses include 1) *optical flow loss*, which provides the model with the supervision of optical flow, and 2) *alpha bootstrapping loss*, which offers supervision of initial coarse masks; additionally, we use 3) *reconstruction loss* to stabilize the training and improve reconstruction quality, and 4) *sparsity loss* to avoid duplicate foreground objects in the texture area. Further details can be found in the supplementary material.

Residual consistency loss. In order to produce significant yet smooth lighting conditions, the multiplicative residuals of different times t_1 and t_2 of the same position on the texture coordinate (u, v), where (u, v, t_1) corresponds to a visible area and (u, v, t_2) corresponds to an invisible area, should be close in distribution and can have different intensities of illumination. Formally, we have a small $k \times k$ patch P

at time t_1 on the video centered at (x, y). We then sample their texture coordinates $(u, v)^{(P)} = \mathcal{M}(P)$. The texture coordinates are then combined with different times t_1 and t_2 to get the corresponding multiplicative residuals:

$$\begin{aligned} \ell^{(P)} &= \mathcal{R}\left((u, v)^{(P)}, t_1\right) , \\ \ell'^{(P)} &= \mathcal{R}\left((u, v)^{(P)}, t_2\right) . \end{aligned} \tag{7}$$

To this end, we encourage multiplicative residuals of the same position on texture coordinate should be positively correlated. Therefore, we define the residual consistency loss through normalized cross-correlation by

$$\psi(P, t_2) = \frac{\left(\ell^{(P)} - \mu_{\ell^{(P)}}\right)\left(\ell'^{(P)} - \mu_{\ell'^{(P)}}\right)}{\sigma_{\ell^{(P)}}\,\sigma_{\ell'^{(P)}}} . \tag{8}$$

We further introduce a variance-smoothness term that smooths the changes in lighting conditions as

$$\mathbb{E}(P, t_2) = \sigma^2_{\ell'^{(P)}} . \tag{9}$$

We apply the above terms to all layers and get the overall loss as

$$\mathcal{L}_{\mathcal{R}\text{con}} = \lambda_{\mathcal{R}\text{con}}(\psi + \beta\mathbb{E}) , \tag{10}$$

where we choose $k = 3$ since our patch size is 3×3 and $\beta = 16$.

Residual regularization. Minimizing the current losses can collapse to a trivial solution because the multiplicative-residual estimator absorbs all colors. We regularize the residuals equal to 1 since "true color" of an object in the video depends on the overall light conditions. That is, if the light source is a pure blue light in the entire video shining on a white wall, we can only say that the steady color of the wall is "blue", as we do not have enough information to determine the true color of the wall.

$$\mathcal{L}_{\mathcal{R}\text{reg}} = \lambda_{\mathcal{R}\text{reg}}||\mathcal{R}(\cdot) - 1||_2^2 . \tag{11}$$

Alpha regularization. We introduce an additional regularization term to address potential issues with the alpha network. Specifically, the sparsity loss used in the alpha network tends to assign a value of zero (*i.e.*, black) to regions that are not visible in the input video, which can result in noisy masks and incorrect shadow embedding. To mitigate this problem, we enforced a constraint that each pixel should contribute to at most one layer. This ensures that the mask for each layer is clean and reliable and that the lighting conditions are properly embedded in each layer.

$$\mathcal{L}_{\alpha\text{reg}} = \lambda_{\alpha\text{reg}}\text{BCE}\left(\max_{n\in\{0,\cdots,N\}} \alpha_n\right) , \tag{12}$$

where BCE is binary cross entropy.

3.6. Hash Grid Encoding

Like [16, 23], we have considered employing positional encoding as the input encoding for the multi-layer perceptrons. The purpose of including the positional encoding is to enrich the discriminative power of the coordinates and ensure that high-frequency details can be adequately represented. Inspired by [24], we choose to adopt hash grid encoding as the input encoding method for our model. By leveraging this technique, the input features are encoded into a set of intermediate representations that span a broad spectrum of spatiotemporal resolutions, ranging from coarse- to fine-grained, and can flexibly adjust to distinct regions while maintaining high levels of accuracy and consistency. Formally, the 2D or 3D input feature is treated as a coordinate of multi-resolution grids and then used to sample data from the grids using interpolation techniques. The resulting sampled data are concatenated and passed into multi-layer perceptrons to obtain the final output.

4. Experiments

We conduct our experiments on the DAVIS dataset [26] and various internet videos to demonstrate the effectiveness of our approach in video reconstruction and consistent video editing. We also design an evaluation metric of *edit consistency* on the TAP-Vid-DAVIS dataset [8]. In addition, we use our method to generate videos with different camera motions from those in the original input video. Furthermore, we conduct ablation studies on the multiplicative-residual estimator and encoding type to assess the impact on the performance of different architecture choices.

4.1. Qualitative Results

Fig. 4 shows four qualitative results of our work on the DAVIS dataset [26]. For each video frame, we show the predicted masks (second and third columns), the m-residuals (fourth column), the corresponding layered textures (fifth and sixth columns), and the reconstruction result (last column). Please also refer to Fig. 1 for an example result highlighting the robustness of our m-residual estimation. More results and higher resolution reconstruction can be found in the supplementary material.

With merely the guidance from the coarse object mask provided by Mask R-CNN [11], our model successfully separates the object from the background yielding a clean and precise mask. In the second video in Fig. 4, we showcase an example of a video occluded by complex water splashes. Our model can accurately reconstruct the video without introducing noticeable texture distortions. We superimpose a rainbow checkerboard on the mask to visualize the transformation of each layer. It is evident that each coordinate's component remains at the same position in the texture.

Since the multiplicative residual may brighten or darken

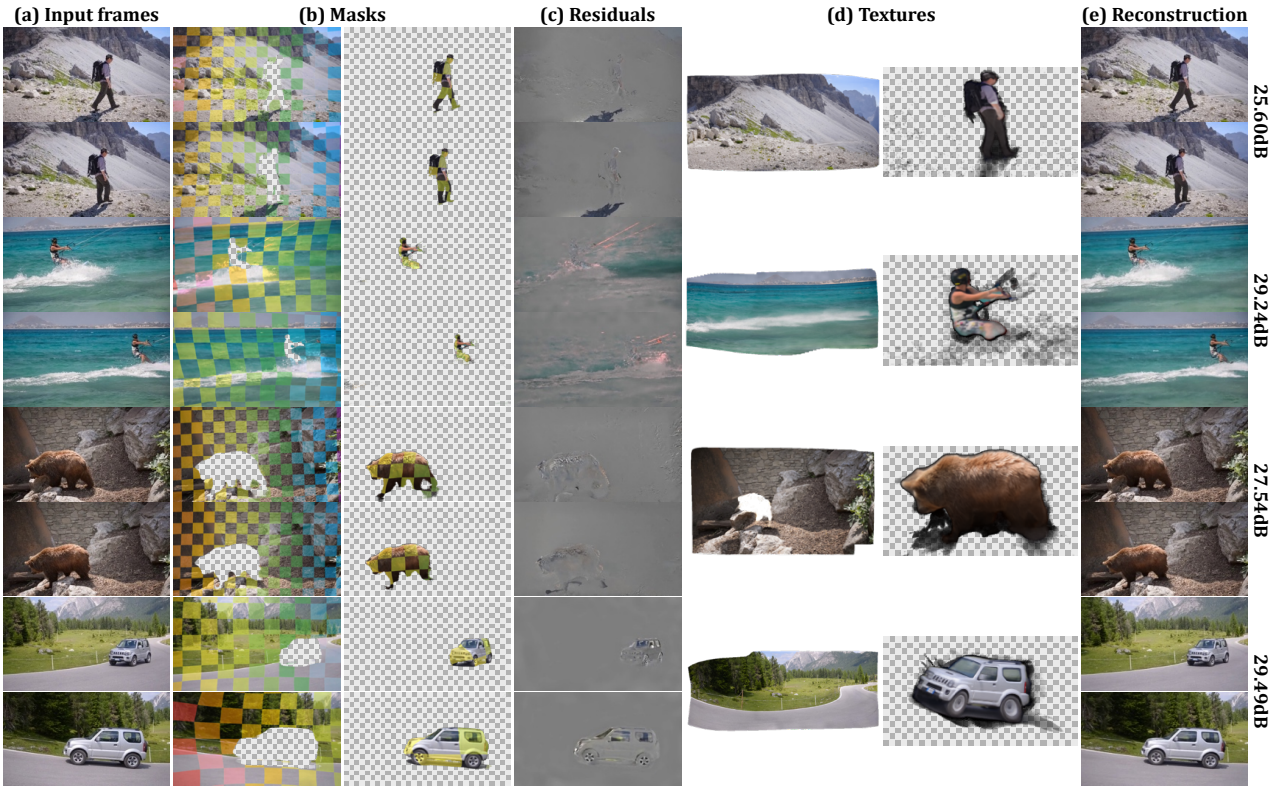

Figure 4: **Qualitative results on the DAVIS dataset.** We show our predicted masks, textures, m-residuals, reconstructed results, and PSNR of different videos. We overlay a color checkerboard on the masked areas to visualize the texture transformations.

the texture, we display the m-residual output on a gray canvas to visualize its effect. In the first video, our m-residual estimator successfully models the shadow of the hiker while the background texture retains its original color. The m-residual estimator takes both the texture coordinate and time as the input, allowing it to learn how to model the changes in lighting conditions through space-time and adjust the color accordingly. Such a mechanism enables our model to generate realistic and consistent shading effects for different objects in the video, such as the shading on the bear's fur or the reflection on the hiker's hat. We also report the PSNR of each video as evidence of high-quality reconstruction.

4.2. Comparison with Previous Work

We report the PSNR values and various settings for every approach in Table 1. We also provide more quantitative comparisons with Deformable Sprites (DS) [39] and Layered Neural Atlases (LNA) [16] under multiple metrics in Table 2. Our method demonstrates superior performance of PSNR, rendering speed, and GPU memory consumption compared to previous methods on all three videos while maintaining the same or better resolution. Note that we could not run Deformable Sprites for the 768×432 resolution, as Deformable Sprites failed to finish due to their GPU memory usage exceeding the limit of a single GPU with 24 GB memory. In contrast, our method can efficiently decompose higher-resolution videos into full-HD layers using affordable computation resources while maintaining comparable reconstruction results. To compute the PSNR, we scale the input video up to the reconstructed resolution. Therefore, the PSNR of our 1080p model might be slightly lower than that of the 768×432 model, as the PSNR of the 1080p model is calculated at the corresponding resolution and considered harder due to having more details to reconstruct than the low-resolution counterpart.

4.3. Consistent Video Editing

For editing purposes, we use a 1000×1000 grid sampling to render the texture networks. We then modify the rendered texture and sample color via bilinear interpolation. As our key advantage lies in the m-residual representation of illumination, we have focused our editing on videos or components that exhibit changes in lighting.

We show our editing results in Fig. 5. We choose three videos to present the rendering quality. The first video is sourced from internet videos, while the other two are taken from the DAVIS dataset [26]. For evaluating the editing quality, we modify the background texture of the first two videos by adding handwritten characters of "ICCV". We also adjust the color of the shirt of the hiker in the second video.

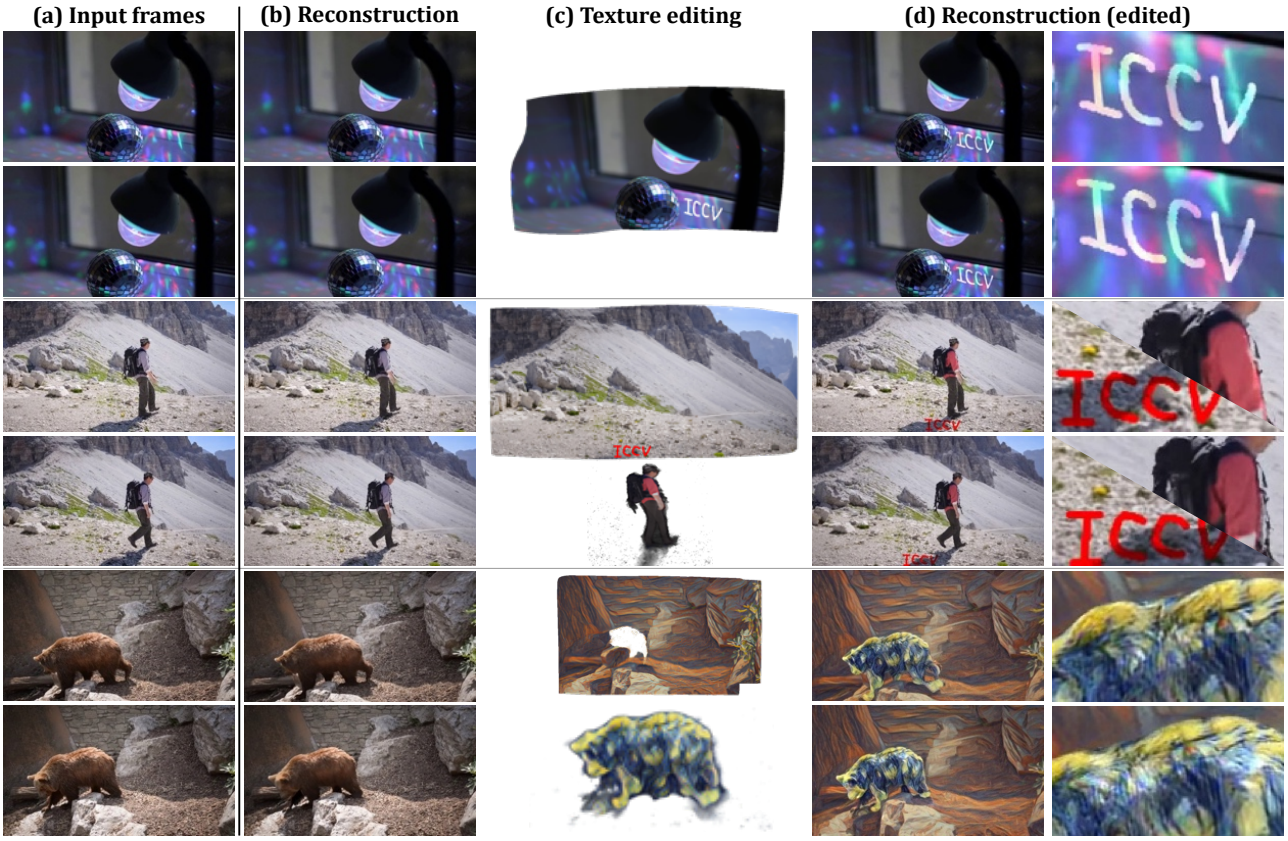

Figure 5: **Edits on videos with varying lighting conditions.** We apply various editing techniques on different components of the videos to observe the corresponding changes in the lighting conditions. The textures with edits are shown in (c), while the editing results are enlarged in the right column of (d). The lighting and shading fuse with the edits flawlessly.

Finally, we perform style transfer on the third video on both background and object textures. In the first video, despite the complexity of the lighting conditions caused by the Disco ball, our approach can successfully handle and diffuse the light onto the edited region, resulting in high-quality output. Previous approaches fail to achieve such a representation of complex lighting conditions. In the second video, the color of the shirt changes convincingly according to the different lighting conditions in two different frames, while our edit on the background is successfully occluded by the shadow. We can see that in the third video, our edited reconstruction follows the textures and the modified lighting conditions fit seamlessly with our modifications, as shown in (d) of the fifth and sixth rows. In the supplementary material, we also provide additional editing results of various types to further demonstrate the versatility and effectiveness of our method.

4.4. Quantitative Results for Editing Consistency

Our work mainly focuses on consistent video editing. However, famous metrics such as PSNR (as we show in Fig. 4) or other reconstruction quality measures do not necessarily reflect the quality of consistent video editing. We take advantage of TAP-Vid [8], which provides several evaluation metrics and the ground-truth feature tracking on the foreground object. We use the feature points of the first unobstructed frame $(f_{0,x}, f_{0,y})$ as our base point and convert it into texture coordinate (u_0, v_0). For all other frames t, we select the video coordinate whose mapped texture coordinate is closest to (u_0, v_0) as our prediction for frame t. Therefore, the feature points on the same track should be mapped to the same coordinate on the foreground texture for an ideal rendering result.

We report our results in Table 3. We select a subset of the TAP-Vid DAVIS dataset for our quantitative analysis. For reference, we also report the results of RAFT [31], as a baseline, on the same videos since our optical flow supervision comes from it. In general, the correspondences obtained by our method are more consistent than those by RAFT, as RAFT considers only neighboring frames while ours works on a unified foreground representation. Specifically, our method achieves higher scores than RAFT on the "black swan" and "kite-surf" videos and comparable results on the other two videos. The low score on the "parkour" video is due to the complex geometric changes in the video, which

Method	Resolution	Training time	Rendering speed (fps)	GPU memory	PSNR bear	PSNR disco ball	PSNR hike
Deformable Sprites [39]	213 × 120	10 minutes	5	5 GB	22.7	26.2	21.5
Deformable Sprites [39]	427 × 240	20 minutes	1.6	12 GB	23.6	26.4	22.0
Layered Neural Atlases [16]	768 × 432	5.5 hours	0.5	3 GB	27.3	29.0	25.2
Ours	768 × 432	40 minutes	787	3 GB	27.5	37.7	25.6
Ours	1920 × 1080	40 minutes	71	5 GB	26.9	35.6	25.4

Table 1: **Comparison results.** We report the PSNR for each video, along with the corresponding training time, rendering speed (frames per second), and GPU memory usage under different resolutions. Our work achieves better results than prior work on the three videos, achieving faster rendering, lower GPU memory consumption, and higher resolution. Note that we measure the PSNR at the corresponding reconstructed resolution. Therefore the PSNR tends to favor the evaluation at low resolution, explaining why our 1080 reconstructions have slightly lower PSNR than our 480p reconstructions.

Method	bear PSNR	bear LPIPS	bear SSIM	disco ball PSNR	disco ball LPIPS	disco ball SSIM	hike PSNR	hike LPIPS	hike SSIM
DS	23.6	0.23	0.78	26.4	0.21	0.89	22.0	0.29	0.68
LNA	27.3	0.19	0.85	29.0	0.11	0.95	25.2	0.19	0.80
Ours	**27.5**	**0.16**	**0.87**	**37.7**	**0.04**	**0.97**	**25.6**	**0.16**	**0.82**
	kite-surf			car-turn			libby		
DS	21.2	0.34	0.62	22.2	0.32	0.65	21.6	0.41	0.57
LNA	28.2	0.30	0.76	27.5	0.30	0.88	29.4	0.31	0.91
Ours	**29.2**	**0.26**	**0.78**	**29.5**	**0.23**	**0.91**	**29.6**	**0.26**	**0.92**

Table 2: **More comparison results.** Our model outperforms previous methods in all videos and achieves better scores in all evaluation metrics.

	ours AJ	ours $< \delta^x_{avg}$	ours OA	RAFT AJ	RAFT $< \delta^x_{avg}$	RAFT OA
black swan	0.81	0.87	1.00	0.39	0.52	1.00
parkour	0.04	0.10	0.48	0.07	0.17	0.48
kite-surf	0.48	0.58	0.99	0.25	0.35	0.86
cows	0.51	0.61	0.99	0.55	0.65	0.99

Table 3: **Quantitative results on TAP-Vid.** We test the editing consistency on a subset of TAP-Vid DAVIS and show the result for each video. The reported metrics, the higher the better, are average Jaccard (AJ), average position accuracy ($< \delta^x_{avg}$), and binary occlusion accuracy (OA). Compared to RAFT [31], our method performs better or reaches comparable results on all videos.

our current architecture can reconstruct well but edit poorly on the foreground parkour runner. Nevertheless, the editing on the background still exhibits comparable quality.

4.5. Manipulating Camera Motion

With our multiplicative residual estimator, we are able to synthesize realistic views that are not shown in the input video. A different camera view can be synthesized by scaling, shifting, or rotating the original video coordinates (x, y) to get (x', y') and concatenating it with the temporal information t. Once the new set of video coordinates $p' = (x', y', t)$ is obtained, we can use the mapping network $\mathcal{M}$ to obtain the texture coordinate $(u, v)^{(p')}$ for each layer. The color $c^{(p')}$ and the lighting $\ell^{(p')}$ of all layers can be obtained based on these texture coordinates. For each non-background layer n, we use the alpha network $\mathcal{A}$ to obtain the original soft mask $\alpha^{(p)}$ for each $p = (x, y, t)$ and establish a relation as $(u, v)_0 \rightarrow \alpha_n^{(p)}$ if $(u, v)_0 = \mathcal{M}_0(p)$. Then, we linearly interpolate the value of the mask at each location p' by triangulating the input data. That is, we establish a correspondence between the coordinates of the object textures and the background texture and then transfer the mask values from the object onto the background texture.

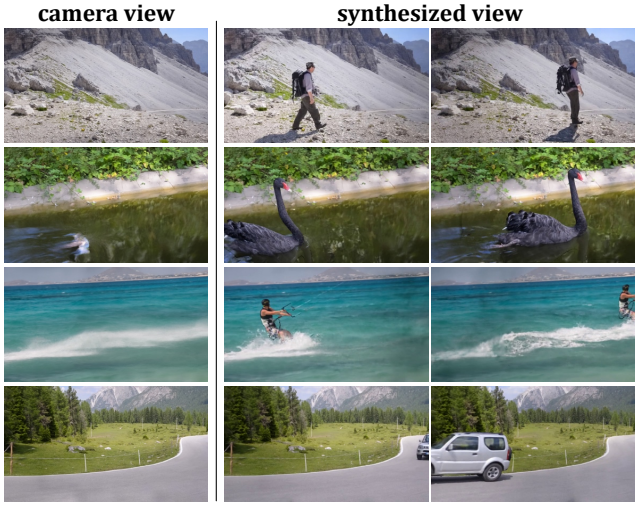

Figure 6: **Camera motion manipulation.** The multiplicative residual serves as a smooth representation that effectively adjusts the lighting conditions with a fixed camera.

We set up a virtual camera in the background to capture the entire video scene from a fixed position. Fig. 6 shows our result. The camera is fixed (*i.e.*, the background remains static) while the object keeps moving in the scene. In the fourth video, the car is partially visible in the beginning frames and fully visible later, with its side facing the camera. The third video demonstrates that the splashes consistently follow the entire path of the original camera view. With our multiplicative-residual estimator, we can successfully reconstruct the surfing scene while accurately representing the water splashes. We show more results with varying camera motions in the supplementary material.

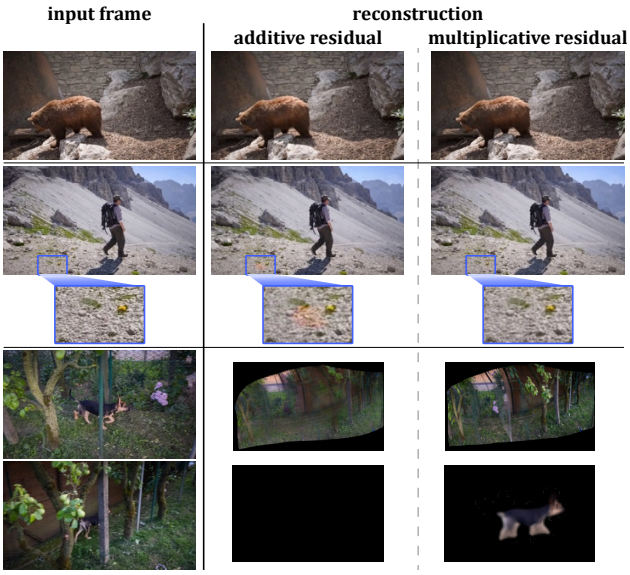

Figure 7: **Choice of the residual estimator architecture.** Using an additive residual estimator architecture may introduce artifacts or degrade the quality of textures, while a multiplicative one works correctly.

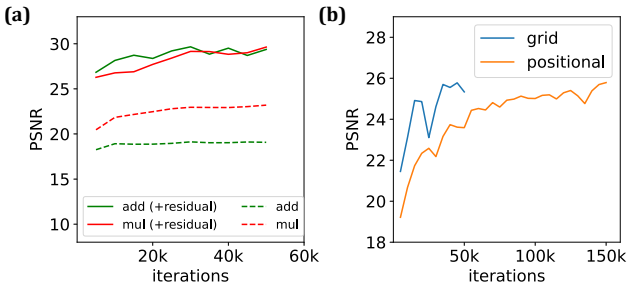

Figure 8: **Quantitative ablation studies.** (a) The reconstruction PSNR degradation of the additive residual is significantly higher than that of the multiplicative residual. (b) Hash grid encoding can achieve better PSNR in fewer iterations.

4.6. Ablations

Residual type. We present an ablation study to verify the choice of the architecture of our residual estimator. As mentioned in Sec. 3.3, we adopt the multiplicative residual instead of the additive residual. We show the comparisons of the two designs in Fig. 7 and Fig. 8a. The additive residual estimator is conducted, and the output is normalized to $[-1, 1]$ to ensure they have the same representation capability. In Fig. 7, both additive and multiplicative residuals provide promising reconstruction results for the first video. However, when it comes to the complex texture, like the background of the second video, the additive residual fails to represent it and compensates for the wrong illuminations. In the third video, the additive residual fails to represent precise background texture, and the dog vanishes from the texture. Fig. 8a demonstrates a significant drop in PSNR when ablating the residual module. This finding reflects that additive residual is prone to overfit the color that should be attributed to the editable texture. In contrast, the multiplicative residual provides complete background and object textures.

Hash grid encoding. We evaluate the effectiveness of hash grid encoding in Fig. 8b. Positional encoding needs more MLP layers to achieve good quality, which takes 1.5× processing time per iteration compared to hash encoding. However, hash grid encoding still achieves better PSNR in fewer iterations.

5. Discussion

We have observed several interesting points that are worth discussing. Our method relies on Mask-RCNN and RAFT to provide the foreground mask and optical flow as external priors. As a result, our method may be biased by the inaccurate learning-based priors on some difficult cases like the spinning lighting in disco ball or the object boundary in bear. The reconstructed objects in the video may exhibit edge artifacts due to referencing the incorrect texture layer. Our method with reconstruction loss could improve the initial external prior but is still not perfect to solve all the artifacts. Adopting better external priors or designing strong internal priors are both good future improvements. The problem of object edge artifacts is similar to the seam artifact in mesh uv-parameterization from which future exploration may take inspiration.

6. Conclusion

We have presented a neural layer decomposition method to facilitate illumination-aware video editing. The proposed multiplicative-residual estimator can effectively derive the layered representation that characterizes the spatiotemporally varying lighting effects. We have also implemented hash grid encoding for fast coordinate inference. Our model, therefore, significantly reduces the training time and achieves real-time rendering speed with a low requirement of computation resources, enabling interactive editing on high-resolution videos. We use our model to generate high-quality video editing results, where, in particular, the varying illumination effect can only be achieved by ours rather than the previous methods.

7. Acknowledgments

This work was supported in part by NSTC grants 111-2221-E-001-011-MY2 and 112-2221-E-A49-100-MY3 of Taiwan. We are grateful to National Center for High-performance Computing for providing computational resources and facilities.

References

[1] Aseem Agarwala, Aaron Hertzmann, David Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004. 2

[2] Yuval Alaluf, Or Patashnik, Zongze Wu, Asif Zamir, Eli Shechtman, Dani Lischinski, and Daniel Cohen-Or. Third time's the charm? image and video editing with stylegan3. In *Computer Vision - ECCV 2022 Workshops - Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part II*, volume 13802 of *Lecture Notes in Computer Science*, pages 204–220. Springer, 2022. 2

[3] Jean-Baptiste Alayrac, João Carreira, Relja Arandjelovic, and Andrew Zisserman. Controllable attention for structured layered video decomposition. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 5733–5742. IEEE, 2019. 2

[4] Connelly Barnes, Dan B. Goldman, Eli Shechtman, and Adam Finkelstein. Video tapestries with continuous temporal zoom. *ACM Trans. Graph.*, 29(4):89:1–89:9, 2010. 2

[5] Zhangxing Bian, Allan Jabri, Alexei A. Efros, and Andrew Owens. Learning pixel trajectories with multiscale contrastive random walks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 6498–6509. IEEE, 2022. 2

[6] Gabriel J. Brostow and Irfan A. Essa. Motion based decompositing of video. In *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pages 8–13. IEEE Computer Society, 1999. 2

[7] Carlos D. Correa and Kwan-Liu Ma. Dynamic video narratives. *ACM Trans. Graph.*, 29(4):88:1–88:9, 2010. 2

[8] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens Continente, Kucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. In *NeurIPS Datasets Track*, 2022. 5, 7

[9] Zeqi Gu, Wenqi Xian, Noah Snavely, and Abe Davis. Factormatte: Redefining video matting for re-composition tasks. *ArXiv*, abs/2211.02145, 2022. 2

[10] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1025–1039, 1998. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, 2020. 5

[12] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7044–7052. IEEE Computer Society, 2017. 2

[13] Allan Jabri, Andrew Owens, and Alexei A. Efros. Space-time correspondence as a contrastive random walk. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[14] Varun Jampani, Raghudeep Gadde, and Peter V. Gehler. Video propagation networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3154–3164. IEEE Computer Society, 2017. 2

[15] Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, pages 199–206. IEEE Computer Society, 2001. 2

[16] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Trans. Graph.*, 40(6):210:1–210:12, 2021. 1, 2, 3, 4, 5, 6, 8

[17] M. Pawan Kumar, Philip H. S. Torr, and Andrew Zisserman. Learning layered motion segmentations of video. *Int. J. Comput. Vis.*, 76(3):301–319, 2008. 2

[18] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 179–195. Springer, 2018. 2

[19] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3809–3817. Computer Vision Foundation / IEEE, 2019. 2

[20] Sharon Lin, Matthew Fisher, Angela Dai, and Pat Hanrahan. Layerbuilder: Layer decomposition for interactive image and video color editing. *ArXiv*, abs/1701.03754, 2017. 2

[21] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. *ACM Trans. Graph.*, 39(6):256:1–256:14, 2020. 2

[22] Erika Lu, Forrester Cole, Tali Dekel, Andrew Zisserman, William T. Freeman, and Michael Rubinstein. Omnimatte: Associating objects and their effects in video. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4507–4515. Computer Vision Foundation / IEEE, 2021. 2

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2020. 3, 4, 5

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 2, 4, 5

[25] William S. Peebles, Jun-Yan Zhu, Richard Zhang, Antonio Torralba, Alexei A. Efros, and Eli Shechtman. Gan-supervised dense visual alignment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Or-*

leans, LA, USA, June 18-24, 2022, pages 13460–13471. IEEE, 2022. 2

[26] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 724–732. IEEE Computer Society, 2016. 5, 6

[27] Hanspeter Pfister. Interactive intrinsic video editing. *ACM Trans. Graph.*, 33(6):197:1–197:10, 2014. 2

[28] Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew W. Fitzgibbon. Unwrap mosaics: a new representation for video editing. *ACM Trans. Graph.*, 27(3):17, 2008. 2

[29] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *Pattern Recognition - 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings*, volume 9796 of *Lecture Notes in Computer Science*, pages 26–36. Springer, 2016. 2

[30] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2288–2297. Computer Vision Foundation / IEEE, 2020. 2

[31] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 402–419. Springer, 2020. 7, 8

[32] Rotem Tzaban, Ron Mokady, Rinon Gal, Amit Bermano, and Daniel Cohen-Or. Stitch it in time: Gan-based facial editing of real videos. In *SIGGRAPH Asia 2022 Conference Papers, SA 2022, Daegu, Republic of Korea, December 6-9, 2022*, pages 29:1–29:9. ACM, 2022. 2

[33] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Trans. Image Process.*, 3(5):625–638, 1994. 2

[34] Wenjing Wang, Shuai Yang, Jizheng Xu, and Jiaying Liu. Consistent video style transfer via relaxation and regularization. *IEEE Trans. Image Process.*, 29:9125–9139, 2020. 2

[35] Josh Wills, Sameer Agarwal, and Serge J. Belongie. What went where. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, pages 37–44. IEEE Computer Society, 2003. 2

[36] Ning Xu, Brian L. Price, Scott Cohen, and Thomas S. Huang. Deep image matting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 311–320. IEEE Computer Society, 2017. 2

[37] Yiran Xu, Badour AlBahar, and Jia-Bin Huang. Temporally consistent semantic video editing. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XV*, volume 13675 of *Lecture Notes in Computer Science*, pages 357–374. Springer, 2022. 2

[38] Yanchao Yang, Brian Lai, and Stefano Soatto. Dystab: Unsupervised object segmentation via dynamic-static bootstrapping. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 2826–2836. Computer Vision Foundation / IEEE, 2021. 2

[39] Vickie Ye, Zhengqi Li, Richard Tucker, Angjoo Kanazawa, and Noah Snavely. Deformable sprites for unsupervised video decomposition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 2647–2656. IEEE, 2022. 1, 2, 6, 8