Omnimatte: Associating Objects and Their Effects in Video

Erika Lu[1,2] Forrester Cole[1] Tali Dekel[1,3] Andrew Zisserman[2]
William T. Freeman[1] Michael Rubinstein[1]

[1]Google Research [2]University of Oxford [3]Weizmann Institute of Science

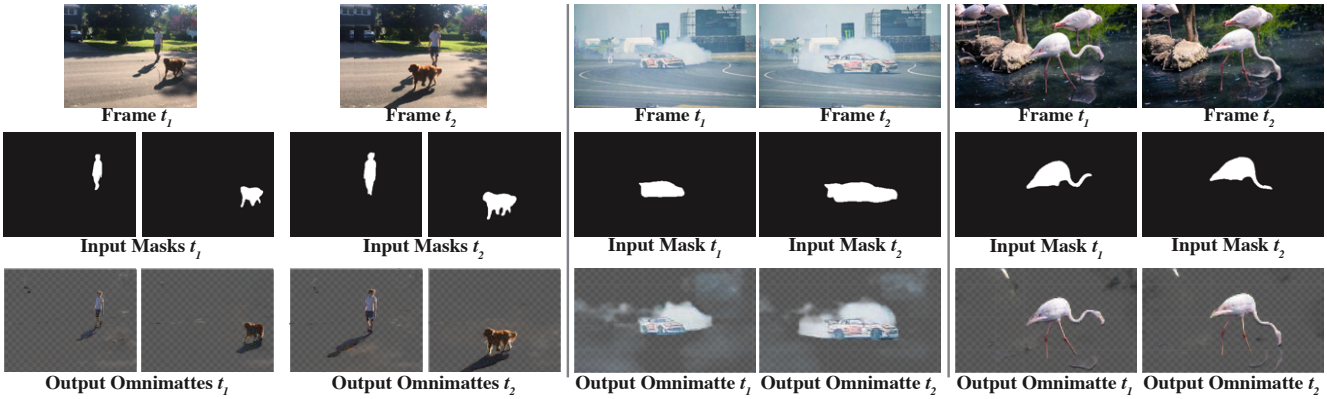


Figure 1. We pose a novel problem: automatically associating subjects in videos with 'effects' related to them in the scene. Given an input video (top) and rough masks of subjects of interest (middle), our method estimates an *omnimatte* – an alpha matte and foreground color that includes the subject itself along with all scene elements associated with it (bottom). The associated elements can be other objects attached to the subject or moving with it, or complex effects such as shadows, reflections, smoke, or ripples the subject creates in water.

Abstract

Computer vision is increasingly effective at segmenting objects in images and videos; however, scene effects related to the objects—shadows, reflections, generated smoke, etc.*—are typically overlooked. Identifying such scene effects and associating them with the objects producing them is important for improving our fundamental understanding of visual scenes, and can also assist a variety of applications such as removing, duplicating, or enhancing objects in video. In this work, we take a step towards solving this novel problem of automatically associating objects with their effects in video. Given an ordinary video and a rough segmentation mask over time of one or more subjects of interest, we estimate an* omnimatte *for each subject—an alpha matte and color image that includes the subject along with all its related time-varying scene elements. Our model is trained only on the input video in a self-supervised manner, without any manual labels, and is generic—it produces omnimattes automatically for arbitrary objects and a variety of effects. We show results on real-world videos containing interactions between different types of subjects (cars, animals, people) and complex effects, ranging from semi-transparent elements such as smoke and reflections, to fully opaque effects such as objects attached to the subject.*[1]

[1]Project page: https://omnimatte.github.io/

1. Introduction

"And first he will see the shadows best, next the reflections of men and other objects in the water, and then the objects themselves, then he will gaze upon the light of the moon and the stars and the spangled heaven ... Last of all he will be able to see the sun." – Plato

Is it possible to automatically determine all the effects caused by a subject in a video? Reflect for a moment on the difficulty of the task: a subject, such as a human wandering through a scene, can cast shadows on the floor and distant walls, and be reflected in windows and other surfaces. These 'effects' are non-local. However, they are *correlated* with the subject's shape, motion and, in the case of reflections, appearance.

Tackling this problem is the objective of this paper. More specifically, given an input video and (possibly rough) segmentations over time of subjects of interest in the video, we seek to produce an output opacity matte (alpha matte) for each subject that includes the subject and their effects in the scene (Figure 1). We call this the *"omnimatte"* of the subject. We additionally produce a color background image containing the static background elements in the video. We achieve this by proposing a network and training framework that is able to automatically determine and segment regions that are *correlated* with the given subject (Figure 2). The model is trained in a self-supervised way only on the in-

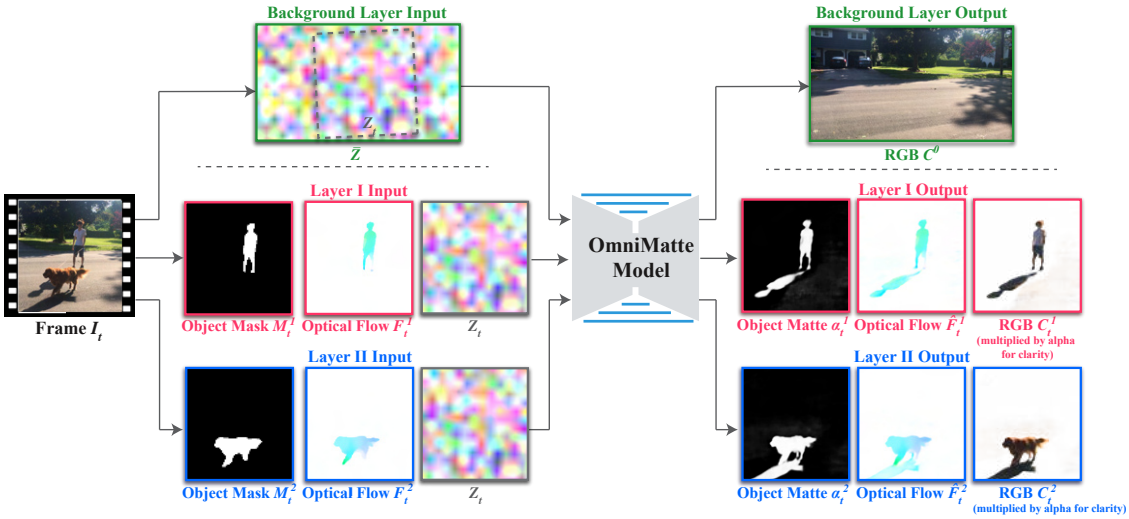


Figure 2. **Estimating omnimattes from video.** The input to the model is an ordinary video with multiple moving objects, and a rough segmentation mask M for each object (left). In a pre-processing step, we compute an optical flow field F between consecutive frames using [28]. For each object, we pass the mask, estimated flow in the object's region, and a sampled noise image Z_t (representing the background) to our model, producing an omnimatte (color + opacity) and an optical flow field for the object (right). In addition, the model predicts a single background color image for the entire video (top), given a spatial texture noise image $\bar{Z}$ as input. See Sec. 3 for details.

put video, without observing any additional examples. Our solution is inspired by the recent work of Lu et al. [17] that presented a method to decompose a video into a set of human-specific RGBA layers. We generalize this technique to support arbitrary objects, by relying only on binary input masks (no object-specific representation or processing) and incorporating general optical flow to account for motion and frame-to-frame correspondence.

Associating objects with their effects not only improves our fundamental understanding of visual scenes and events captured in video, it can also support a range of applications. Consider for example the problem of removing a person or other types of objects from a video. As is well known, a common error in person removal, e.g., by inpainting, is that a shadow or reflection of the person remains, resulting in a video left with just a 'shadow of the former self'. The erroneous missing of a reflection is a central plot point in the film 'Rising Sun' (1993), and the converse, a lack of reflection, a common trope of vampire movies. The important point is that manipulating an object in a video requires dealing not only with the object; its *effects* in the scene need to be adjusted together with the object in order to create realistic and faithful renditions.

We demonstrate results of inferring omnimattes for different objects such as animals, cars, and people, capturing a variety of complex scene effects including shadows, reflections, dust and smoke. We evaluate the resulting omnimattes qualitatively and quantitatively, and also demonstrate how omnimattes can be useful for video editing applications such as object removal, background replacement, "color pop", and stroboscopic photography.

2. Related Work

Video layer decomposition Our work is inspired by seminal works on layered video decomposition such as [30] and [6]. Layered representations of images and videos have been applied widely in computer vision and graphics, for example, for inferring occlusion relationships (*e.g.* [6]), depth (*e.g.* [39, 27]), and synthesizing novel views (*e.g.* [26, 29]). In particular, our work builds on the recent work of Lu et al. [17] that presented a method for decomposing a video into a set of human-specific RGBA layers, where each layer represents a person and their associated scene elements. They take a neural rendering approach and represent the geometry and texture of people explicitly, using a dedicated, human-specific pipeline. We demonstrate that the neural rendering component is in fact unnecessary, and that comparable results can be achieved by providing only rough segmentation masks and flow as input to the network. The result is a simpler and more efficient setup that allows the model to handle arbitrary moving objects. We compare the results with [17] in Sec. 4.5.

Image and video matting Image and video matting traditionally deals with the problem of estimating a foreground layer (color + opacity) and a background color image from a given image or a video (e.g., [5, 31, 15, 7, 37, 13, 25]). The novel problem we propose—estimating an omnimatte—also aims at estimating color + opacity layers from an input video. However, the key fundamental difference is that omnimattes capture not only an object but also all the various scene effects that are correlated with the object. None of the existing matting methods is suitable for performing this task: they cannot handle well entirely semi transparent objects, typically require accurate trimaps that are generated manually, and they are often restricted to estimating two layers (background/foreground). In practice, film production uses manual or semi-automatic rotoscoping to create mattes with such effects [14]. Our method works automatically and generically on natural, ordinary videos that contain arbitrary moving objects and scene effects, and requires only rough object masks (e.g., see the flamingo example in Fig. 1).

Background subtraction Change detection using background subtraction [20, 9] typically does not produce alpha mattes, but binary masks containing all objects and effects (such as shadows). Qian and Sezan's "difference matting" work [23] attempts to use background subtraction and thresholding to produce a foreground matte with a known background image, but the results are very sensitive to the thresholding value. [25] recently modernized that approach, producing nice quality, continuous-valued alpha mattes but still require a known, clean image of the background. More importantly, background subtraction and difference matting cannot solve the omnimatte problem when the video has *multiple objects with effects*. In such cases it is not enough to detect the effects, each effect must also be associated its subject. We evaluate our method numerically and compare it with background subtraction using a change detection dataset [34] with pixel-level labels for objects and shadows.

Shadow and reflection detection Specialized methods also exist for detecting, removing, or modifying specific types of effects, such as shadows and reflections. For example, [8] acquire a shadow displacement map by waving a stick over different parts of a scene, then use it to synthesize realistic shadows that match an object's shape. [2, 3] decompose natural videos to remove reflections, shadows, and smoke. More recently, Wang et al. [33] proposed to analyze the motion of people in video and use it to predict depth, occlusion, and lighting/shadow information, to increase realism of 2D object insertion. Our goal in this work is to provide a general technique for inferring all of a subject's associated effects. However, as shadows are a particularly common effect, we compare our results with a state-of-the-art shadow detector [32] in Sec. 4.1.

Video Effects Although omnimattes are not explicitly optimized for editing, they can facilitate various video editing effects that rely on input object masks, including object removal and video completion (*e.g.* [10, 35]), object cut-and-paste (*e.g.* [15, 31]), color pop, and creation of stroboscopic images from video (*e.g.* [1]). All of these effects can be achieved via simple manipulations of the estimated omnimattes in a post-processing step, or alternatively, by using the omnimattes as input to existing methods such as video completion (*e.g.* [10]), to save the manual work required for marking the object's effects. We demonstrate these results in Sec. 4.1.

3. Estimating Omnimattes from Video

The input to our method is an ordinary video of moving objects, and one or more layers of rough segmentation masks that mark the subjects of interest. The output is an *omnimatte* for each input mask layer, consisting of an alpha matte (opacity map) and a color image. The model is trained per-video to reconstruct the input in a self-supervised manner, without observing any additional examples.

To accurately reconstruct the input video, the model must infer all the time-varying effects (*e.g.* shadows, reflections) from the input object masks, which do *not* represent those effects. Our goal is to steer the model to place the associated effects in the layer of the subject causing them. Lu, et al. [17] showed that this association can be achieved by showing the network one mask at a time, leveraging the fact that an effect is easier to predict from the object mask most correlated with it. For example, the mask of the person in Figure 1 provides more information about its shadow (more similar to it in shape, in motion) compared to the mask of the dog. Therefore (as shown in [17]) the network tends to learn to predict the person's shadow from the person's mask (thus associating it with the correct layer). We build on this training strategy, but design network inputs and losses to encourage this solution for general objects.

3.1. Overview

Figure 2 illustrates our pipeline. Our model is a 2D U-Net [24] that processes the video frame by frame. For each frame, we compute rough object masks using off-the-shelf techniques to mark the major moving objects in the scene. We group the objects into N mask layers $\{M_t^i\}_{i=1}^N$ and define a (possibly time-varying) ordering o_t for the layers. For example, in a scene with a rider, a bicycle, and several people in a crowd, we might group the rider and bicycle into one layer, while grouping the crowd into a second layer. To equip our model with explicit information about *object motion* and frame-to-frame correspondence, we also compute a dense optical flow field, F_t, between each frame and the consecutive frame in the video. This flow field is masked by the input masks M_t^i to provide the network only flow information related to the layer's subject. We additionally align all frames onto a common coordinate system using homographies, and represent the background as a single unwrapped image on a separate layer.

From this rough yet explicit representation of moving objects, the model has to infer: (i) *omnimattes* – pairs of continuous-valued opacity maps (mattes) and RGB images that capture not only the i^{th} moving object but also all the scene elements that are correlated with it in space and time (e.g., reflections, shadows, attached objects, etc.), (ii) a refined optical flow field for each layer, and (iii) a background RGB image. Formally,

$$\text{Omnimatte}(I_t, H_t, M_t^i, F_t^i) = \mathcal{L}_t = \{\alpha_t^i, C_t^i, \hat{F}_t^i\}, \quad (1)$$

where I_t, H_t, M_t^i, F_t^i are the input video RGB frame, estimated camera homography, the initial input mask, and the pre-computed flow field of the i^{th} object in time t, respectively. α_t^i and C_t^i are the alpha and color buffers of the output omnimatte, and $\hat{F}_t^i$ is the predicted object flow.

The training loss consists of terms on the RGBA outputs (Sec. 3.2) and the predicted flow (Sec. 3.3). The main loss is a reconstruction loss $\mathbf{E}_{\text{rgb-recon}}$, but as reconstruction is underconstrained with multiple layers, we add a sparsity regularization $\mathbf{E}_{\text{reg}}$ to the alpha layers and an initialization loss $\mathbf{E}_{\text{mask}}$ to the masks, similar to [17]. We encourage the *motion* of the result to match the input by adding a flow-reconstruction loss $\mathbf{E}_{\text{flow-recon}}$ and a temporal consistency term to the alpha mattes $\mathbf{E}_{\text{alpha-warp}}$.

The total loss is:

$$\mathbf{E}_{\text{rgb-recon}} + \lambda_{\text{r}}\mathbf{E}_{\text{reg}} + \lambda_{\text{m}}\mathbf{E}_{\text{mask}} + \mathbf{E}_{\text{flow-recon}} + \lambda_{\text{w}}\mathbf{E}_{\text{alpha-warp}}, \quad (2)$$

where λ_{r}, λ_{m}, and λ_{w} are weighting coefficients (see supplementary material (SM)). As the background is assumed to be static, we factor out camera motion and treat the background with a special, fixed layer (Sec. 3.4).

3.2. RGBA Losses

The main loss in our optimization is a *reconstruction loss*. Formally, we composite the set of estimated layers for each frame and the predicted background layer using standard back-to-front compositing [22], and encourage the composite image to match the original frame:

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

where $\mathcal{L}_t = \{\alpha_t^i, C_t^i\}_{i=1}^N$ are the predicted layers for frame t, and o_t is the compositing order.

To prevent a trivial solution where a single layer reconstructs the entire frame, we further apply a regularization loss to the α_t^i to encourage them to be spatially sparse. We use a mix of L_1 and an approximate-L_0:

$$\mathbf{E}_{\text{reg}} = \frac{1}{T}\frac{1}{N}\sum_t\sum_i \gamma \left\|\alpha_t^i\right\|_1 + \Phi_0(\alpha_t^i), \quad (4)$$

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.

To guide the optimization to convergence from a random initialization, we therefore adopt a "bootstrap" loss to coerce the alpha maps α_t^i to match the input masks M_t^i:

$$\mathbf{E}_{\text{mask}} = \frac{1}{T}\frac{1}{N}\sum_t\sum_i \left\|d_t^i \odot (M_t^i - \alpha_t^i)\right\|_2 \quad (5)$$

where $d_t^i = 1 - \texttt{dilate}(M_t^i) + M_t^i$ is a boundary erosion mask to turn off the loss near the mask boundary, and $\odot$ is element-wise product. This loss is turned off after its value reaches a fixed threshold (see SM).

3.3. Flow Losses

Our model additionally predicts a set of *flow layers*. Predicting flow layers serves as an auxiliary task that injects information about motion to our model and improves our decomposition (as demonstrated by our experiments). To achieve that we apply a *flow reconstruction loss* and a *photometric warping loss* defined below:

$$\mathbf{E}_{\text{flow-recon}} = \frac{1}{T}\sum_t W_t \cdot \|F_t - Comp(\mathcal{F}_t, o_t)\|_1, \quad (6)$$

where $\mathcal{F}_t = \{\hat{F}_i^t\}$ is the set of predicted flow layers, F_t is the original, pre-computed flow, and W_t is a spatial weighting map that lowers the impact of pixels with inaccurate flow. W_t is computed based on standard left-right flow consistency error and photometric warping error (see full details in SM).

We additionally encourage temporal consistency within layers using an *alpha warping loss*:

$$\mathbf{E}_{\text{alpha-warp}} = \frac{1}{T}\frac{1}{N}\sum_t\sum_i \|\alpha_t^i - \alpha_{wt}^i\|_1, \quad (7)$$

where $\alpha_{wt}^i = Warp(\alpha_{t+1}^i, \mathcal{F}_t^i)$ is the alpha for layer i at time $t+1$ warped to time t using the predicted flow.

3.4. Camera Motion and Background

We assume the background scene is stationary and camera motion can be modeled by a time-varying homography from an unwrapped "canvas" image, as in [30]. The homographies H_t from frame t to the canvas are estimated via feature tracking (using [11]) on the original RGB video frames and are held fixed. For input to the network, the background canvas is represented by a single spatial noise image $\bar{Z}$ (see Fig. 2). The background color layers C_t^0 are produced by feeding $\bar{Z}$ through the network to form a static color image $\bar{C}^0$, which is then sampled using H_t^{-1} to form time-varying background images $\{C_t^0\}$.

To make the foreground layers aware of the camera motion, the input mask layers M_t^i are concatenated with a noise image that tracks the camera. The background noise image $\bar{Z}$ is sampled using H_t^{-1} to form time-varying noise images $\{Z_t\}$. This is a similar approach to Lu, et al. [17], but in our case the noise image is not trainable.

Minor stabilization errors, as well as exposure changes, vignetting, and radial distortion, usually cause slight changes in appearance even for a stationary background. If the background is assumed to be entirely static, these subtle shifts in appearance will show up as noise in our omnimatte. Such effects, however, tend to have low spatial and temporal frequency relative to the subject's effects and can be safely captured by applying a *refinement warp* to the background layer. The refinement warp consists of a spatially and temporally coarse grid-based warp. We additionally apply a grid-based brightness adjustment to the final composite $Comp(\mathcal{L}_t, o_t)$. The parameters of the warp and brightness adjustment are optimized together with the network parameters (see SM for additional details).

3.5. Implementation Details

For all our results, we used Mask R-CNN [12] to segment the input objects, and STM [18] (a video object segmenter trained on the DAVIS dataset [21]) to track objects across frames. Optical flow between consecutive frames was computed using RAFT [28]. When dynamic background elements such as tree branches are present, we use panoptic segmentation [36] to segment them and treat the segment as additional objects. To increase the detail of the color buffers C_t^i, we apply a similar detail-transfer technique to Lu, et al. [17]. See SM for training details.

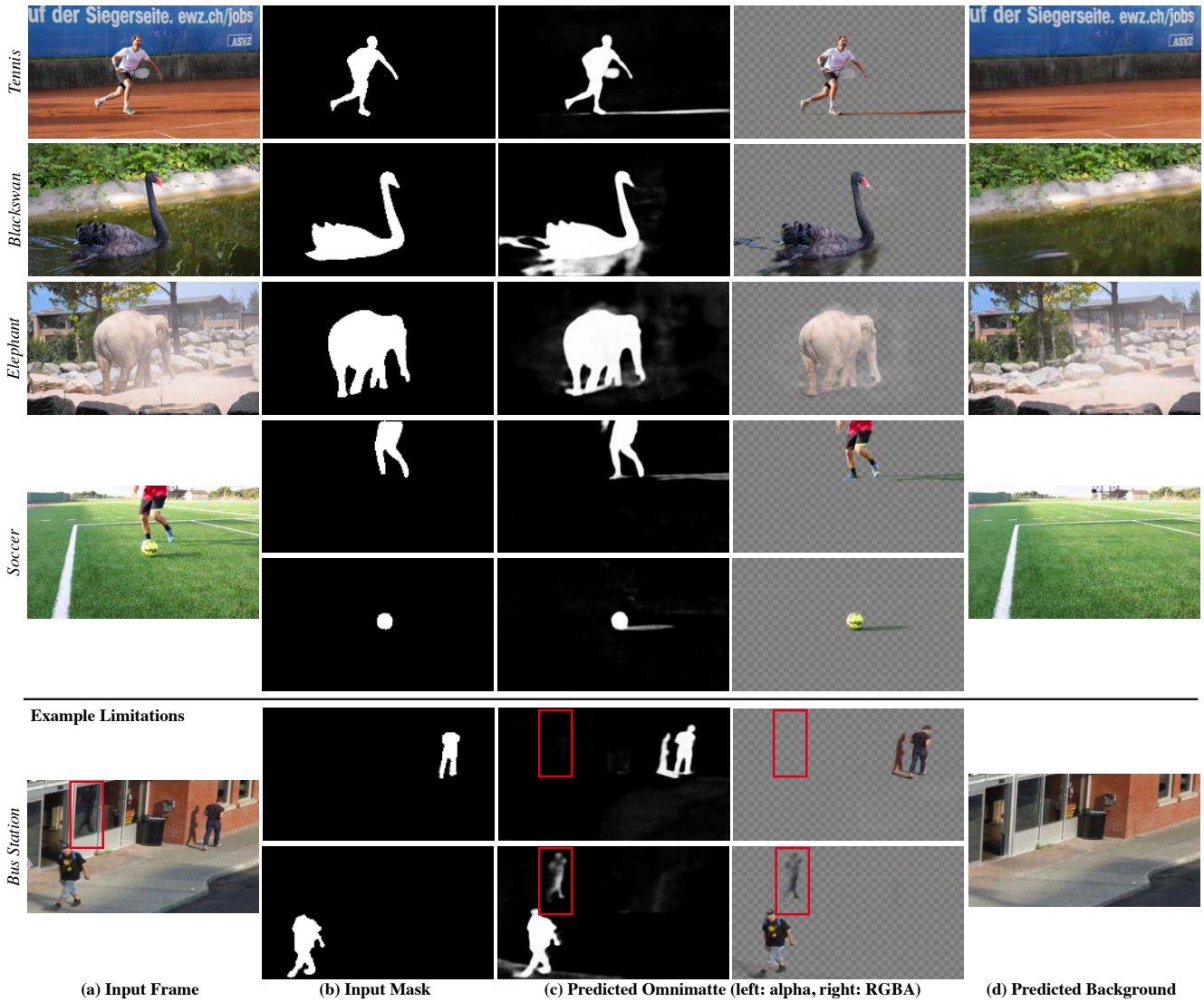


Figure 3. **Results on natural videos.** For each example, we show: **(a)** input frame; **(b)** input mask(s) computed by Mask R-CNN [12]; **(c)** our resulting omnimatte (left: alpha matte, right: RGBA); **(d)** our estimated background layer. The bottom example (*Bus Station*) shows a failure case: while the shadows are correctly associated with the people, the reflection cast on the window by the person in the top-right corner (marked by the red rectangle) is mistakenly grouped with the person in the bottom-left corner.

4. Results

4.1. Qualitative examples on real videos

Figure 3 shows examples of our estimated omnimattes on a variety of real-world videos from DAVIS [21], CDW-2014 [34] (see Sec. 4.4), and videos downloaded from YouTube. These examples span a wide range of dynamic subjects (e.g., people, animals or general moving objects such as a soccer ball), performing complex actions and generating various scene effects including shadows, reflections, water ripples, dust and smoke. None of the input object masks include these effects (see Fig. 3(b)).

As seen in Fig. 3(c-d) top, our method successfully associates the subjects with the scene effects that are related to them. In *Blackswan*, the omnimatte of the swan captures its reflection and the water ripples it causes. In *Elephant*, our omnimatte captures the semi-transparent cloud of dust sprayed by the elephant, as well as the shadow the elephant casts on the ground. In *Tennis*, the running player casts thin shadows, which the omnimatte correctly separates from the shadows in the background. Additionally, although the player's racket is not included in the input mask (b), it is reconstructed in our omnimatte result; this demonstrates our model's ability to reconstruct objects that are attached to the main subject even when given incomplete input masks.

In *Soccer*, we show a two-subject example where our model estimates a separate omnimatte for each of the subjects: a person (top row) and a soccerball (bottom row). Our model successfully separates the person's shadow from that of the soccerball up until the final few frames of the video, where part of the person's shadow appears in the soccerball's omnimatte (full video in SM).

Another two-subject example is shown in *Bus Station* where two people walk away from each other. Our model

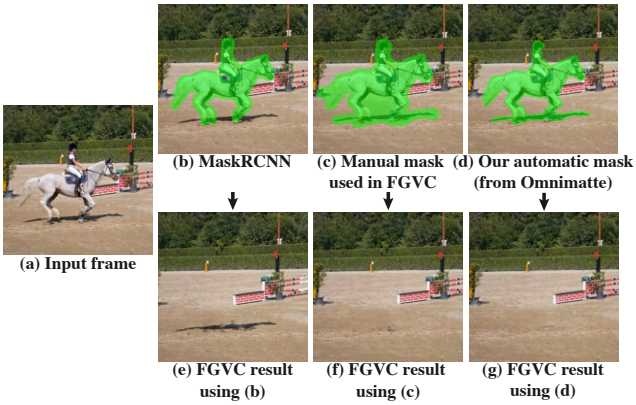


Figure 4. **Omnimattes as input to state-of-the-art object removal.** Results by FGVC [10] using different types of input masks: **(b)** Raw masks from MaskRCNN do not capture shadows, and produce unrealistic results **(e)**. By using manually annotated masks that include the shadow **(b)**, both the horse and the shadow are removed **(f)**. **(d)** binary mask *automatically derived from omnimatte* produces comparable result **(g)** when inputted to FGVC.

correctly associates each person with their shadow in this challenging case. However, the reflection cast on the window by the person on the right (top) is incorrectly placed in the left person's layer (bottom). The challenge of this scene lies in both the spatial proximity of the reflection to the incorrect person, and the similar motions (both people in the scene are moving consistently at the same speed). Lu, et al. [17] showed that layers tend to 'grab' spatially proximal effects; in this case, the reflection is actually closer to the person who is *not* casting the reflection. [17] additionally showed that correlated motions are grouped in the same layers; as both people are walking in synchronization, the network places the reflection in the incorrect person's layer.

4.2. Object Removal

Our method can be applied to remove a dynamic object from a video by either: (i) binarizing our omnimatte and using it as input to a separate video-completion method such as FGVC [10], or by (ii) simply excluding the object's omnimatte layer from our reconstruction.

As shown in Fig. 4(b,e), removing an object but not its correlated effects produces an unrealistic result (object removed but its shadow remains). Typically such effects are manually annotated to create a conservative binary mask of the regions to remove (Fig. 4(c)). To show that an omnimatte can replace manual editing, we derive a binary mask by thresholding our soft alpha at 0.25 and dilating by 20 pixels, and inputting it to FGVC [10]. Fig. 4(c) shows both the horse and its shadow are removed, demonstrating that our derived mask is comparable to a manually annotated mask.

Fig. 5 shows a comparison between omnimatte removal (approach (ii) above) and FGVC using manual masks. In the *flamingo* example, our method removes not only the flamingo but also its reflection in the water beneath it. FGVC relies on a mask that does not include the reflection, thus the reflection remains intact in their result. Our omnimattes bypass the need to manually label such semi-

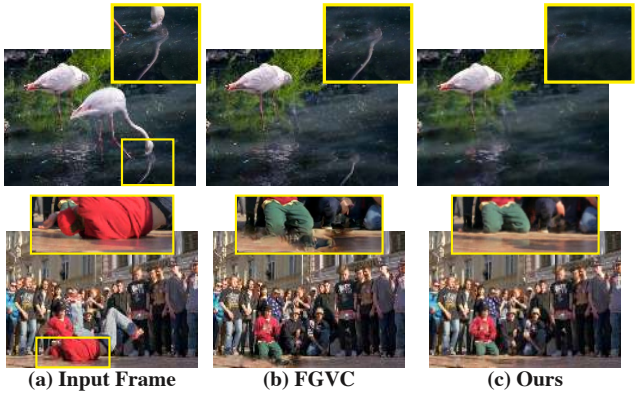


Figure 5. **Direct omnimatte-based removal.** For each input frame **(a)** we show the result of removing a foreground object by excluding its omnimatte from the reconstruction **(c)**, compared to FGVC [10] **(b)**.

transparent effects. In the *breakdance* example, both the crowd and the dancer are moving. To handle this case, we assign an omnimatte to the dancer and a separate single omnimatte to the crowd. Fig. 5(c) shows the crowd omnimatte composited with the background layer. The FGVC result (b) shows artifacts on the ground where the dancer is removed, whereas our result is seamless and realistic.

4.3. Comparison with Shadow Detection

We show qualitative comparisons with a recent state-of-the-art shadow detection method, ISD [32], a deep-learning based method that takes an RGB image as input and produces segments for object-shadow pairs. ISD integrates a MaskRCNN-like object detection stage (Detectron2 [36]), hence it does not require or allow an input mask.

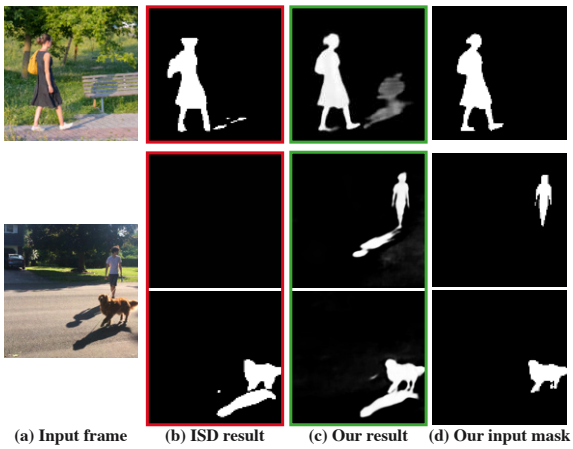
Figure 7 compares our result with ISD on two challenging scenes, where a person casts a shadow onto another object (a bench), and where a person's shadow is occluded by another object (a dog). Our method successfully handles and outperforms ISD in both cases. Occlusions and shadows cast on other objects present particularly difficult cases for purely data-driven methods such as ISD, since the appearance of the shadow depends on the relative configuration of multiple objects in the scene, presenting a combinatorial explosion of scenarios for training. In contrast, our method analyzes and leverages space-time information throughout the entire video to perform these complex object-effects associations.

4.4. Comparison with Background Subtraction

We quantitatively evaluate our approach on the task of background subtraction using a change detection dataset, CDW-2014 [34], which has ground-truth pixel-level labels for objects and hard shadows. We selected a subset of videos that contain objects and their shadows (see Fig. 8 for sample frames and labels). We manually excluded "bad weather" and "low framerate" categories to avoid low quality videos, and selected short clips of up to 5 moving objects. The selected subset contains 12 clips, each with 40 - 115 frames, for approximately 950 frames in total. While the background subtraction task requires only sep-

Color Pop **Background Replacement** **Stroboscopic Photograph**

Figure 6. **Video editing with Omnimattes.** Effects such as "color pop" (left; subject in color, background in grayscale), background replacement (center), and stroboscopic photography (right) all benefit from capturing the subjects' associated effects with an Omnimatte. Note the color pop present in the flamingo's reflection and the correct placement of shadows in the background replacement and stroboscopic photograph examples. See the SM for the full details and before/after videos.



(a) Input frame **(b) ISD result** **(c) Our result** **(d) Our input mask**

Figure 7. **Comparison with shadow detection.** **(b)** Results produced by ISD [32], a recent state of the art, single-image shadow detection method, and **(c)** our results when using **(d)** MaskRCNN masks as input. See SM for additional comparisons.

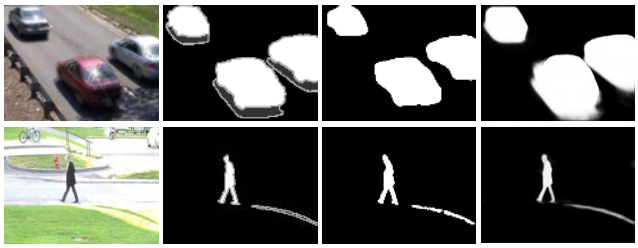
arating foreground from background, we demonstrate the additional capabilities of our method by also segmenting the effects for individual object instances.

We convert our soft omnimattes into a single, hard segmentation mask using a fixed threshold value and report the Jaccard index ($\mathcal{J}$) and Boundary measure ($\mathcal{F}$) [19] in Table 1. We compare with two top-performing methods on CDW-2014, FgSegNet [16] and BSPVGAN [38], which were trained on subsets of CDW-2014. Our method outperforms FgSegNet and matches the performance of BSPVGAN, despite not being trained supervised on CDW-2014.

4.5. Comparison with Layered Neural Rendering

Fig. 10 shows a qualitative comparison with the human-specific, layered neural rendering method by Lu et al. [17]. In [17], people are parameterized explicitly using per-frame UV maps that represent each individual's geometry, and a per-person trainable texture map that represents appearance. Instead, we use binary masks and pre-computed optical flow to represent object regions (see Sec. 3). For comparison, we used binary masks extracted from their UV maps.

In both examples, our method achieves comparable results to [17], successfully capturing the trampoline deformations, shadows and reflections, yet with a generic, much simpler input. Note that the input masks derived from the UV maps provided by [17] represent the full body of a person even if they are occluded in the original frame. This



(a) Input Frame **(b) Ground truth** **(c) BSPVGAN** **(d) Ours**

Figure 8. **Comparison with background subtraction.** We used selected videos from the CDW-2014 change detection dataset [34] (examples input frames in **(a)**), with ground truth, manually segmented objects and shadows (**b**, white pixels = moving objects, dark gray pixels = shadows, light gray pixels = 'unknown', typically at boundaries). **(c)** Result by a top-performing (on this dataset) background subtraction method [38]. **(d)** Our result (alpha mattes of estimated omnimattes). Numerical experiments are summarized in Table 1. More results can be found in the SM.

Method	$\mathcal{J}\&\mathcal{F}$(Mean) ↑	$\mathcal{J}$(Mean) ↑	$\mathcal{F}$(Mean) ↑
FgSegNet [16]	0.675	0.631	0.719
BSPVGAN [38]	**0.756**	**0.718**	0.793
Ours	0.754	0.711	**0.797**

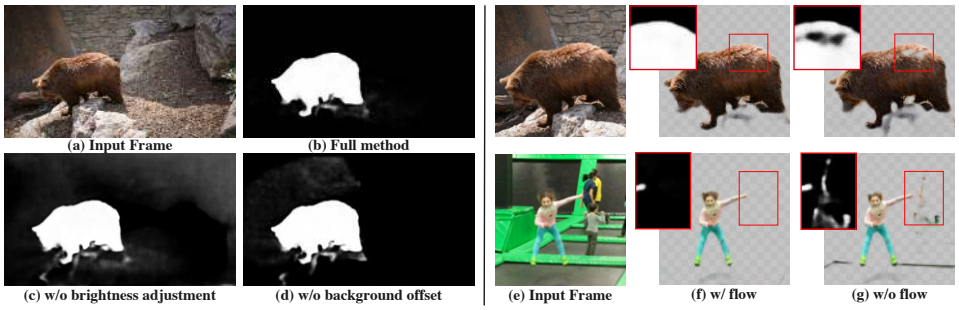
Table 1. We compare our method to the two top-performing methods on CDW-2014 [34]. We report the Jaccard index ($\mathcal{J}$) and Boundary measure ($\mathcal{F}$) on a subset of the data that includes objects and their shadows. Our method performs at par or better than the two background subtraction methods.

allows our model to inpaint object regions and scene effects that are occluded in some frames but visible in others, as in [17]. The ability of our model to inpaint occluded regions even when using incomplete masks is also evident in the person-dog example in Fig. 1, where the person and their shadow are reconstructed in our omnimatte. However, we note that in cases where the input mask is substantially occluded, the output omnimatte will show occlusion as well; thus in order to deal with large occlusions, a full-object mask should be inputted to the model, as done in [17].

4.6. Additional Video Editing Effects

The additional information present in an omnimatte compared to a standard matte that includes only the subject allow simple creation of various video effects such as color pop, background replacement, or object duplication (Fig. 6). Previously, creating these effects for videos containing shadows or reflections required extensive manual editing effort. Since the omnimatte is a standard RGBA im-

Figure 9. **Ablations.** We ablate several components of our method: **Left: (a-b)** sample input frame and our result using our full method, and **(c)** removing the brightness adjustment, and **(d)** removing the background offset (Section 3.4). **Right:** we show two examples comparing our method with and without the flow component **(f-g)**.



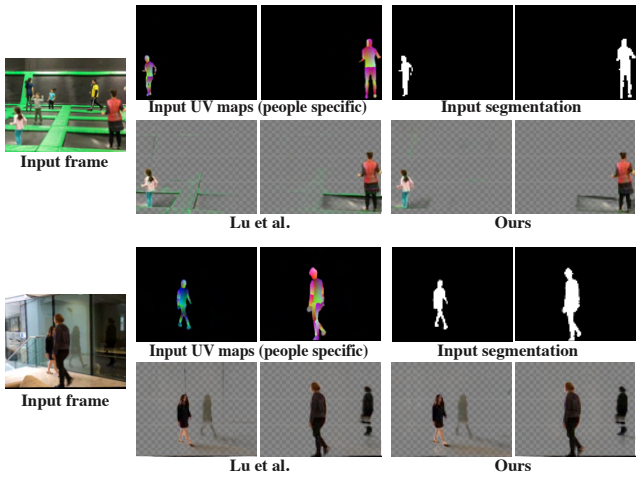


Figure 10. **Comparison with Lu et al. [17].** We achieve comparable results to [17] using just binary masks instead of the people-specific UV maps used in [17]. These binary segmentation masks are easier to obtain and are *general* – allowing our method to support arbitrary objects ([17] is applicable just to people). Notice how our omnimattes capture trampoline deformation well (top two rows), and reflections in the glass (bottom two rows).

age, these edits may be applied using standard video editing software. The omnimattes for color pop and background replacement were used unchanged, the horse jump alpha matte was adjusted with a linear contrast ramp. Please see SM for full details on creating these effects.

4.7. Ablations

In Fig. 9 we ablate several components of our method. Removing the brightness adjustment (c) and removing the background offset (d) both result in undesirable nonzero alpha values in the bear's omnimatte, due to lighting changes, vignetting, and homography inaccuracies that break the static background assumption (Sec. 3.4). Including both components results in a clean alpha matte (b).

We ablate the flow component of our model by removing both flow inputs and flow losses (Sec. 3.3), and show results in (g). In the top row, part of the bear is missing from the omnimatte, and in the bottom row, the person's omnimatte incorrectly contains parts of other people. In contrast, our full model (f) has a complete bear and a clean person omnimatte. These examples show that providing the model with motion information (flow) allows it to better associate scene elements with the correct objects, and prevents holes appearing in the foreground object's alpha matte.

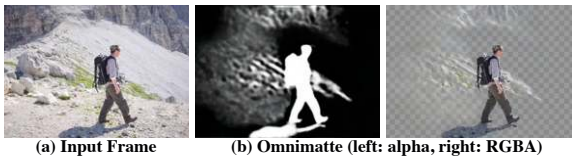


Figure 11. **Failure case due to incorrect camera registration.** When the background motion cannot be accurately represented by a homography (in this case due to a significant depth variation in the scene), the predicted omnimatte may contain regions of the background to compensate for the registration inaccuracies.

4.8. Limitations

While our method allows for small deviations from a static background via smooth, coarse geometric and photometric offsets, when the homographies do not accurately represent the background, the omnimattes must correct for these errors by including background elements (e.g. rocks and grass, Fig. 11). Conversely, we cannot separate objects or effects that remain entirely stationary relative to the background throughout the video. These issues could be addressed by building a background representation that explicitly models the 3D structure of the scene (e.g. [4]).

Finally, we observed that different random initializations of the network's weights may occasionally lead to different, sometimes undesirable, solutions (see supplemental material for visualization). We speculate that more reliable convergence could be obtained by further optimizing the order in which frames are introduced to the model.

5. Conclusion

We have posed a new problem: from an input video with one or more segmented moving subjects, we produce an *omnimatte* for each subject – an opacity map and color image that includes the subject itself along with the visual effects related to it. These effects can be reflections of the subject, shadows they cast, or attached objects. We have proposed a network and training framework for solving this new problem, and have demonstrated omnimattes produced automatically for real-world videos with a variety of objects and associated effects. We have also shown how omnimattes can support a variety of video editing applications.

Acknowledgements. This work was supported in part by an Oxford-Google DeepMind Graduate Scholarship and a Royal Society Research Professorship. We thank Weidi Xie for assisting with object removal baselines.

References

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *ACM SIGGRAPH 2004 Papers*, pages 294–302. 2004.

[2] Jean-Baptiste Alayrac, João Carreira, and Andrew Zisserman. The visual centrifuge: Model-free layered video representations. In *CVPR*, 2019.

[3] Jean-Baptiste Alayrac, Joao Carreira, Relja Arandjelovic, and Andrew Zisserman. Controllable attention for structured layered video decomposition. In *ICCV*, 2019.

[4] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020.

[5] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. Video snapcut: robust video object cutout using localized classifiers. *TOG*, 2009.

[6] Gabriel J Brostow and Irfan A Essa. Motion based decompositing of video. In *ICCV*, 1999.

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

[8] Yung-Yu Chuang, Dan B Goldman, Brian Curless, David H Salesin, and Richard Szeliski. Shadow matting and compositing. In *SIGGRAPH*. 2003.

[9] Ahmed Elgammal, David Harwood, and Larry Davis. Nonparametric model for background subtraction. In *ECCV*, 2000.

[10] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. In *ECCV*, 2020.

[11] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR*, 2011.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[13] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. In *ICCV*, 2019.

[14] Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J Brostow, and Neill DF Campbell. Roto++ accelerating professional rotoscoping using shape manifolds. *ACM Transactions on Graphics (TOG)*, 35(4):1–15, 2016.

[15] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. In *SIGGRAPH*, 2005.

[16] Long Ang Lim and Hacer Yalim Keles. Learning multi-scale features for foreground segmentation. *arXiv preprint arXiv:1808.01477*, 2018.

[17] 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. In *SIGGRAPH Asia*, 2020.

[18] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019.

[19] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[20] Massimo Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, 2004.

[21] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[22] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, Jan. 1984.

[23] Richard J Qian and M Ibrahim Sezan. Video background replacement without a blue screen. In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 4, pages 143–146. IEEE, 1999.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI*, 2015.

[25] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *CVPR*, 2020.

[26] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998.

[27] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019.

[28] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[29] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *ECCV*, 2018.

[30] John Wang and Edward Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 1994.

[31] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *TOG*, 2005.

[32] Tianyu Wang, Xiaowei Hu, Qiong Wang, Pheng-Ann Heng, and Chi-Wing Fu. Instance shadow detection. In *CVPR*, 2020.

[33] Yifan Wang, Brian L. Curless, and Steven M. Seitz. People as scene probes. In *ECCV*, 2020.

[34] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. CDnet 2014: An expanded change detection benchmark dataset. In *CVPR Workshop*, 2014.

[35] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *PAMI*, 2007.

[36] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[37] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, 2017.

[38] Wenbo Zheng, Kunfeng Wang, and Fei-Yue Wang. A novel background subtraction algorithm based on parallel vision and bayesian GANs. *Neurocomputing*, 2020.

[39] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.