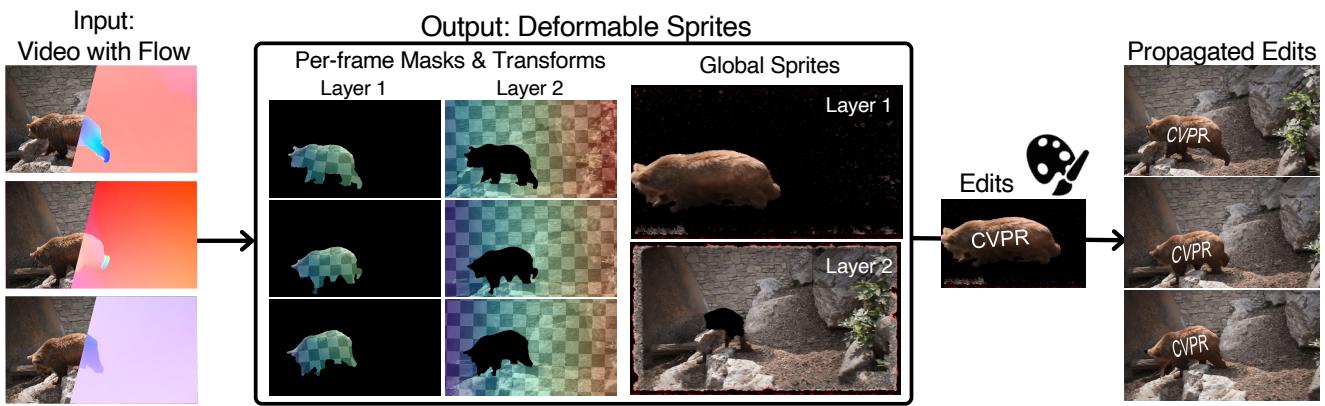


# Deformable Sprites for Unsupervised Video Decomposition

Vickie Ye<sup>1</sup>Zhengqi Li<sup>2</sup>Richard Tucker<sup>2</sup>Angjoo Kanazawa<sup>1</sup>Noah Snavely<sup>2</sup><sup>1</sup>University of California, Berkeley<sup>2</sup>Google

**Figure 1. Video Decomposition with Deformable Sprites.** Given an RGB video and its optical flow (left), we present a method that decomposes the video into layers of persistent motion groups. We represent each group as a *Deformable Sprite* (center), which consists of an RGB sprite image, and masks and non-rigid transforms mapping the sprite to each frame. We fit Deformable Sprites to each input video independently without any pre-training or user input. The resulting decomposition captures long-term correspondences of sprites over time, enabling effects such as propagating edits on the sprite across the entire video (right). We show full videos of our results at the [project site](#).

## Abstract

We describe a method to extract persistent elements of a dynamic scene from an input video. We represent each scene element as a Deformable Sprite consisting of three components: 1) a 2D texture image for the entire video, 2) per-frame masks for the element, and 3) non-rigid deformations that map the texture image into each video frame. The resulting decomposition allows for applications such as consistent video editing. Deformable Sprites are a type of video auto-encoder model that is optimized on individual videos, and does not require training on a large dataset, nor does it rely on pre-trained models. Moreover, our method does not require object masks or other user input, and discovers moving objects of a wider variety than previous work. We evaluate our approach on standard video datasets and show qualitative results on a diverse array of Internet videos.

## 1. Introduction

When we observe a video of a dynamic scene, such as the bear video in Fig. 1, we do not see a disjoint set of pixels over time, but rather a bear walking in a zoo. However, computer vision methods often represent videos as 3D raster

pixel grids. While this low-level representation is convenient for processing on hardware, it does not capture our intuitive notion of high-level objects moving through a 3D scene.

*Moving layers* are an alternative representation proposed in the seminal work of Wang & Adelson, in which scene elements are modeled as persistent image layers that transform over time to compose each video frame [35]. Such layered representations capture the idea that there are persistent motion groups that move smoothly in the scene, while still accounting for sharp edges that result from occlusions. However, these classic methods were limited by machinery of the time to relatively simple motions and scenes.

Inspired by these classic ideas [10, 27, 35], we present a new approach that decomposes videos of complex dynamic scenes into sets of persistent motion groups. We do so by introducing *Deformable Sprites* (Fig. 1, center), representation of motion groups across an entire video. A Deformable Sprite for a motion group consists of three key components: 1) a canonical texture image, or *sprite*, describing the group’s appearance over all input frames, 2) masks locating the group in each input frame, and 3) a non-rigid geometric transformation that maps each sprite into each frame. The resulting decomposition captures the correspondences of each motion group across the entire video, such that modifications of the

sprite can be propagated consistently throughout the video (Fig. 1, right).

We achieve this decomposition by fitting the Deformable Sprites representation to a video without any user input or even an *a priori* notion of what kind of objects will be present. Instead, the decomposition is derived solely from image and motion cues present in the video. Our approach optimizes the Deformable Sprites for each video independently, and does not require training on a dataset. This freedom from the need for training data allows our method to handle videos of novel objects and categories that are not labeled in standard segmentation benchmarks.

Our approach can handle videos with moving camera and articulated or deformable objects. To capture such non-rigid motion, we model the transformation as composition of a homography with 2D spatial splines that evolve smoothly over time. This explicit parameterization has the benefit of modeling non-rigid deformation with relatively few parameters while being continuous. The sprites and the masks are optimized through a convolutional neural network.

Our method has several advantages over recent approaches for video decomposition, which do not recover persistent layer appearances [21, 39], or require user inputs in the form of segmentation masks [20]. Although our method outputs a rich video decomposition, and not simply object masks, we evaluate it on standard video object segmentation benchmarks where we obtain competitive results. On DAVIS [26], we obtain decomposition results that are similar to recent approaches that require user mask initialization [11], while being faster to optimize (30 minutes vs. 10 hours) due to the low dimensionality of our deformation model. We further demonstrate our approach on a variety of Internet clips, where off-the-shelf segmentation methods do not generalize to discover meaningful groupings. To our knowledge, we present the first work that demonstrates video decomposition with a global texture model on in-the-wild videos without any user supervision.

## 2. Related work

**Layer-based video decomposition.** Our work is inspired by the rich history of work that factorizes appearance and motion in videos by representing video frames as compositions of moving layers or sprites. Wang & Adelson introduced this concept to computer vision in the 1990’s [35], building on earlier work on estimation of multiple motions in image sequences [2, 5]. Wang & Adelson solve for a set of ordered RGB $\alpha$  layers with associated affine transformations using optical flow as a motion cue, and a number of other layer-based decomposition methods followed [3, 10, 15, 36]. These methods generally factor image sequences into appearance and motion according to different motion models (e.g. rigid, affine). Closely related is unwrap mosaics [27], which model objects in a video with a single global texture plus per-image

warp fields and occlusion masks.

More recently, layered video decomposition methods extend these formulations to handle more complex, realistic videos. For instance, Lao et al. [16] extends the layered model to explicitly handle 3D motion. More modern methods utilize neural networks in various ways, such as augmenting object masks [21] or computing texture atlases and uv-coordinates for multiple layers using coordinate-based neural networks [11]. One shared aspect of these works is that they require reasonable input masks of an object of interest, usually provided by the user or by segmentation networks trained on pre-defined object categories. In contrast, we explore this problem in a fully unsupervised setting.

Our work is also closely related to MarioNette [32], which decomposes a video into set of static sprites, and like us operates in an unsupervised manner. However, their learned sprites are static, not deformable, and so they mainly demonstrate their work on video game imagery, where even animated sprites are just repetitions of a few discrete frames. They must learn multiple representations of a single object to account for such motion. In contrast, we can learn a single global sprite per object, and model complex object motion using non-rigid transformations. Other recent work explores 3D object discovery in various settings [25, 41], but can require manual annotation or simplified appearance models.

**Motion segmentation.** As one aspect of computing our video representation, we also solve a motion grouping problem. Approaches to motion segmentation include treating the problem as one of a spatio-temporal image clustering [31], or alternatively starting from motion estimates in the form of optical flow or point trajectories, and solving a grouping problem to associate pixels into a number of motion clusters [4, 12, 13, 22, 24]. Recent approaches have also explored optimizing neural networks to map image and/or flow inputs to segmentation masks [6, 37]. Yang et al., recognizing that the motion of a foreground object should not be predictable from background motion (and vice versa), propose an adversarial approach to compute segmentations that are mutually uninformative [40]. While these works combine motion and appearance cues for segmentation, they are designed for frame-by-frame prediction. In contrast, our work computes explicit global texture images for each scene element, shared over all frames of a video. Our resulting representation thus gives us long-term correspondences across entire videos.

**Unsupervised video object segmentation.** While motion segmentation may or may not result in groups corresponding to semantic objects; *video object segmentation* (VOS) methods seek to detect objects in videos. Such methods are often trained on videos in advance—either in a fully supervised manner on datasets like DAVIS [26], or via self-supervision—to learn what constitutes objects in videos. Relevant to our work are approaches that do not consider full supervision.

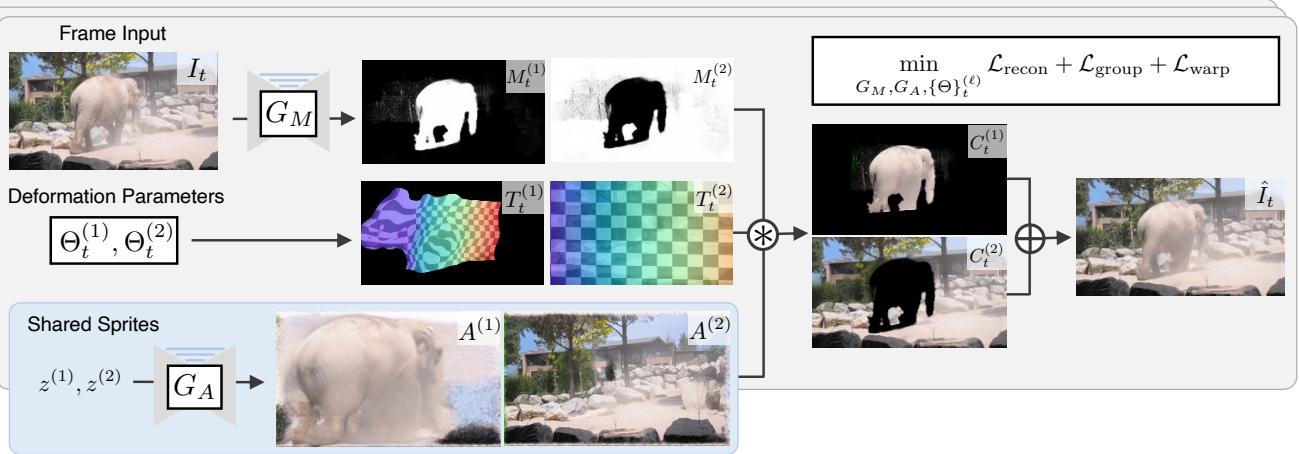


Figure 2. **Deformable Sprites**, a representation for persistent motion groups in a video of a dynamic scene. Here we represent  $L = 2$  motion groups: the elephant and background. Deformable Sprites disentangle video into groups of three components: 1) a single global appearance model, in the form of canonical per-layer RGB texture (*sprites*) used across the entire video (denoted as  $A^{(\ell)}$ ), 2) per-frame masks indicating each group (denoted as  $M_t^{(\ell)}$ ), and 3) spatio-temporal non-rigid transformations that capture the deformation and camera motion in each frame (denoted as  $T_t^{(\ell)}$ ). Transforming then masking the sprites results in a set of layers that are composed to reconstruct the input frame  $t$  while preserving occlusion effects. Deformable Sprites are optimized by minimizing per-frame losses on the reconstructions, masks and transformations described in Sec 4.

Such methods include that of Koh *et al.* [14] and Hu *et al.* [9], which leverage low- and mid-level cues like flow and edges to compute foreground objects (where the latter requires training an edge detector), and Yang *et al.* [38], which uses slot attention [19] and self-supervision to learn to detect objects from optical flow. DyStaB learns object saliency models using object motion in video, which can then be applied at test time to segment objects, even in static images [39]. In contrast to these approaches, we do not seek to detect semantic object categories, but to find the decomposition that best explains the scene motion and its resulting appearance. As we show below, this allows us to represent moving elements that do not fall into traditional object categories.

### 3. Deformable Sprites

The inputs to our method are video frames of a dynamic scene  $\{I_t \in \mathbb{R}^{H \times W}\}_{t=1}^N$ , and optical flow computed between consecutive frames,  $\{F_{t \rightarrow t+1}\}_{t=1}^{N-1}$ . From these inputs, we aim to recover a set of  $L$  distinct *motion groups* in the dynamic scene. We want these groups to capture the underlying elements in the video, moving smoothly through the scene. To do this, we introduce *Deformable Sprites* to consistently represent motion groups as they evolve throughout a video. A Deformable Sprite consists of a canonical appearance model in the form of a texture image, or *sprite*, for each group, used to describe the motion group’s appearance across all video frames, and geometric transformation and mask models, used to relate the motion group’s position

and geometry in each input frame to the sprite.

We illustrate our representation, and our approach to fitting it to an input video, in Fig. 2. For each frame  $I_t$ , we predict a mask for each of the  $L$  motion groups, denoted  $\{M_t^{(\ell)}\}_{\ell=1}^L$ . In the figure, there are two masks corresponding to two motion groups, elephant and background. Each motion group has its own RGB texture sprite describing the group’s appearance across the entire video. We denote these  $L$  texture images as  $\{A^{(\ell)}\}_{\ell=1}^L$ . Finally, we need to know where to place and how to deform each sprite in each frame, *e.g.* to match the position and pose of the elephant in the figure. To that end, we estimate a spatio-temporal spline-based transformation  $T_t^{(\ell)}$  between the canonical texture coordinates and the frame coordinates of  $I_t$ , for every motion group  $\ell$  and every frame  $t$ . Given a set of Deformable Sprites, we can reconstruct any video frame  $I_t$  by using the associated transformations to warp the global textures into the given frame, masking each layer according to the computed masks, then composing the layers into a single RGB image. This yields a reconstructed image  $\hat{I}_t$ .

**Design motivations.** A natural way to fit any kind of layered model to images is to simply optimize the layers and transforms to minimize a reconstruction loss. However, this video decomposition problem is severely ill-posed—a large variety of layer decompositions might perfectly reproduce the video but yield nonsensical group separations. We seek a natural decomposition where the discovered layers correspond to objects that are coherent within a frame and consistent across time. This desire for a coherent decomposition suggests that

the decomposition should be compact and low-dimensional, which is a key motivation for representing the appearance of each motion layer with a fixed sprite texture. As a consequence, we must also represent the geometry of each sprite per frame—where to place it, and how to pose it. We employ a spatio-temporal spline to achieve this non-rigid transformation, which can be parameterized with a small number of explicit parameters.

### 3.1. Grouping

A key question in computing our layered representation from a video is: for each frame, which pixels belong to which layer? Many prior works use optical flow cues to compute explicit motion groups, either with robust parametric model fitting (e.g. [35, 36]), or with robust clustering methods (e.g. [12, 18, 22–24]). However, explicit techniques struggle to handle objects with multiple motions and complex trajectories. Moreover, we are not solely interested in computing motion groups per-frame, but rather in representing the both the appearance and the motion of a group as it evolves through the entire video. For these reasons, we opt to learn a single CNN grouping model,  $G_m$ , for each video sequence (and shared across all frames of that sequence), to predict masks that are consistent with appearance cues.  $G_m$  takes in an RGB image  $I_t$  and predicts  $L$  soft masks  $\{B_t^{(\ell)}\}_{\ell=1}^L \in [0, 1]^{H \times W}$  for that frame, one per group. We then use back-to-front compositing to convert these to probability masks that sum to 1 (per pixel) across the groups:

$$M_t^{(\ell)} = B_t^{(\ell)} \cdot \prod_{i=\ell+1}^L (1 - B_t^{(i)}). \quad (1)$$

Our final grouping is the per-pixel mask,  $M_t^{(\ell)}$ , for each layer  $\ell$  and video frame  $t$ . We use optical flow in computing motion groups; however, rather than explicitly computing groups directly from flow, we use it in our losses as supervision (Sec. 4), and let the model learn which appearance cues in the input frames to associate with each motion group.

### 3.2. Global Sprite Images

The central component of Deformable Sprites is a global sprite image that represents each layer’s appearance across the entire video as a fixed RGB texture. In the case of a dynamic foreground object, the canonical representation helps group together parts of the object moving in complex ways. For example, consider the walking bear in Fig. 1. As it walks, different parts of the body undergo distinct motions. These parts are difficult to group together if one looks at motion alone. However, a global sprite image can group these distinct trajectories together because the collective body has a consistent appearance over time.

For each layer  $\ell$ , we represent its appearance across the entire video as a fixed texture image  $A^{(\ell)} \in \mathbb{R}^{H' \times W'}$ . We

leverage the natural image prior of CNNs [34] and optimize each sprite image  $A^{(\ell)}$  through a shared CNN,  $G_A$ , with a fixed code  $z \in \mathbb{R}^{H' \times W' \times d}$  uniformly sampled from  $[0, 1]$ .

While the global sprites provide cues to group parts that move differently in a single frame, a single, fixed sprite is not sufficient if the camera or the object moves in the scene. Therefore we estimate a set of transformations that can non-rigidly deform the sprites to match the appearance of the motion groups in each frame.

### 3.3. Spatio-Temporal Deformation with Splines

In general, objects move about the world in a smooth fashion. However, as noted by Wang & Adelson [35], modeling the 2D motion field as a smooth function fails to capture sharp edges resulting from occlusion boundaries. The benefit of a layered representation like ours, then, is that the masks capture the visibility and occlusion boundaries, allowing the transformations to be modeled as smooth functions.

Specifically, we model the motion of each sprite with a transformation that maps the pixel coordinates of frame  $t$  to the texture coordinates of layer  $\ell$ . We represent this transformation as a *continuous* function  $T^{(\ell)}(t)$  that is smooth over space and time via two splines, one for space and one for time. This approach has the benefit of modeling non-rigid deformation with relatively few motion parameters. For simplicity, we omit layer superscripts in the notation below.

**Splines in space.** The spatial transformation evaluated at time  $t$  is modeled as a combination of rigid and non-rigid 2D transformations  $\Theta(t) = [\eta(t), \mathbf{V}(t)]$ . The rigid component is modeled as a homography  $\mathbf{H}(t)$ , parameterized by 8 parameters  $\eta(t) \in \mathbb{R}^8$ . This component captures scaling effects and simple object and camera motions. We assume that the background layer, assigned to layer index  $L$ , is largely static (and hence only moving due to camera motion), and so we only model its motion with a homography, without a non-rigid component.

The non-rigid component of the motion is modeled by a 2D B-spline, which defines a smoothly varying deformation field  $D$ . The B-spline is parameterized by a grid of fixed control points  $X \in [-1, 1]^{N_i \times N_j \times 2}$ , with uniform spacing  $\Delta_x$  in the input (view) frame coordinates. We then define a grid of parameters,  $\mathbf{V}(t) \in \mathbb{R}^{N_i \times N_j \times 2}$ , at these control points. These parameters describe the position of the frame control point when it is transformed into texture (canonical) coordinates. Then at a point  $\mathbf{x}$  in the input frame, the deformation vector  $\mathbf{v} = D(\mathbf{x}; \mathbf{V})$  can be interpolated as

$$D(\mathbf{x}; \mathbf{V}) = \sum_{i,j} \mathbf{V}_{i,j} \cdot B_d^{2d} \left( \frac{\mathbf{x} - X_{i,j}}{\Delta_x} \right), \quad (2)$$

where  $B_d^{2d}$  is a degree  $d$  piecewise polynomial defined over the unit square  $[0, 1]^2$ ; we use degree 2. More details about B-spline interpolation are provided in the supplemental.

**Keyframes over time.** We expect corresponding points within each layer to deform smoothly over time. To enforce this smoothness, we represent our transformations over time as a 1D function parameterized by temporal control points or *keyframes*. This allows us to represent temporally varying transformations with a smaller number of parameters compared to estimating a dense transformation separately for every time  $t$  while ensuring smoothness. We store a set of  $N_T$  keyframes of transformation parameters  $\{\Theta_i\}_{i=1}^{N_T}$  at times  $\{t_i\}_{i=1}^{N_T}$ , for every layer. The keyframes are spaced uniformly at time intervals  $\Delta_T := \frac{N}{N_T}$ . At time  $t$ , we interpolate between these keyframes with a 1D B-spline [1]:

$$\Theta(t) = \sum_i \Theta_i \cdot B_d^{\text{1d}}\left(\frac{t - t_i}{\Delta_T}\right), \quad (3)$$

where  $B_d^{\text{1d}}$  is a degree  $d$  piecewise polynomial defined over the unit interval  $[0, 1]$ ; we use degree 2.

The final transformation from an image coordinate  $\mathbf{x}$  to texture coordinate  $\mathbf{p}$  of layer  $l$  at time  $t$  is then

$$\mathbf{p} = T_t^{(\ell)}(\mathbf{x}) = H(\mathbf{x}; \eta^{(\ell)}(t)) + D(\mathbf{x}; \mathbf{V}^{(\ell)}(t)), \quad (4)$$

where  $\eta^{(\ell)}(t)$  and  $\mathbf{V}^{(\ell)}(t)$  are interpolated values at time  $t$ . The parameters to be estimated are the spatio-temporal control points  $\{\Theta_i\}_{i=1}^{N_T}$ , where  $\Theta_i := [\eta_i, \mathbf{V}_i]$ .

**Compositing.** Given a set of Deformable Sprites representing a video, we can reconstruct a frame  $\hat{I}_t$  at time  $t$  as follows. For each layer  $\ell$ , we resample the canonical texture  $A^{(\ell)}$  with the transformation  $T_t^{(\ell)}$  into texture coordinates to get frame appearance  $C_t^{(\ell)}$ . Then, each appearance can be composited with the masks  $\{M_t^{(\ell)}\}_{t=1}^N$  to get our final reconstruction.

## 4. Training Deformable Sprites

We fit our Deformable Sprites model using gradient descent to optimize model parameters, namely the per-frame, per-layer masks  $M_t^\ell$ , per-layer spatio-temporal transformation keyframes  $\{\Theta_i\}_{i=1}^{N_T}$ , and per-layer global appearance sprites  $A^\ell$ . Our primary loss for optimizing the representation is a video reconstruction loss. However, because our recovery problem is under-determined, we add regularizing losses to encourage our model toward solutions with sprites that (1) move coherently, and (2) are consistent over time.

**Motion grouping loss.** We determine how to group together pixels using losses on the optical flow vectors between consecutive frames. In particular, we use the optical flow field to estimate the static background content, as well as to group together dynamic pixels that move similarly to each other:

$$\mathcal{L}_{\text{group}} = \mathcal{L}_{\text{static}} + \mathcal{L}_{\text{dynamic}}. \quad (5)$$

We denote the optical flow between frames  $t$  and  $t+1$  as  $F_{t \rightarrow t+1}$ ; we refer to flow correspondences as  $(x, x')$ , where  $x' = F_{t \rightarrow t+1}(x)$ .

$\mathcal{L}_{\text{static}}$  uses a robust estimate of the two-view geometry to separate static from dynamic points. We estimate the fundamental matrix between frames  $t$  and  $t+1$ ,  $\hat{\mathbf{F}}_{t \rightarrow t+1} \in \mathbb{R}^{3 \times 3}$ , with least median of squares (LMedS) regression [30] on the dense correspondences from flow. Note that we fit  $\hat{\mathbf{F}}_{t \rightarrow t+1}$  to all pixels in the frame, and must detect when the LMedS estimate fails capture the static geometry, when non-static pixels outnumber static pixels (see the supplement).

We then compute the Sampson distance  $\varepsilon_t(x, x')$  using the estimated fundamental matrix [7]. The Sampson distance approximately captures how well the correspondence  $(x, x')$  adheres to the epipolar geometry of  $\hat{\mathbf{F}}_{t \rightarrow t+1}$ ; points with high Sampson distance are likely moving, and therefore do not obey the geometric constraints. We thus penalize the background layer with

$$\mathcal{L}_{\text{static}} = \sum_x \varepsilon_t(x) \cdot M_t^{(L)}(x) + \beta \cdot (1 - \varepsilon_t(x)) \cdot (1 - M_t^{(L)}(x)), \quad (6)$$

where  $M_t^{(L)}$  is the background mask, and  $\beta = 0.002$ .

We further wish to group dynamic points that move similarly to each other. We encourage this with  $\mathcal{L}_{\text{dynamic}}$ , which minimizes the distortion of the optical flow in each moving group, much like in k-means clustering. For each group  $\ell$ , we determine a group exemplar flow vector  $\mu_t(\ell) \in \mathbb{R}^2$ . We then penalize the group assignments with

$$\mathcal{L}_{\text{dynamic}} = \sum_{\ell,x} M_t^\ell(x) \cdot \|F_{t \rightarrow t+1}(x) - \mu_t^\ell\|^2. \quad (7)$$

We let the exemplar for each layer  $\ell$  be the weighted mean of the flow vectors assigned to that layer:

$$\mu_t^\ell \leftarrow (\sum_x M_t^\ell(x) \cdot F_{t \rightarrow t+1}(x)) / (\sum_x M_t^\ell(x)). \quad (8)$$

**Optical flow consistency losses.** Ideally, even as points in the scene move with respect to the camera, they should map to the same coordinates in the texture map. To encourage this behavior, we penalize inconsistencies between the mappings, masks, and optical flow for each layer:

$$\mathcal{L}_{\text{warp}} = \mathcal{L}_{\text{transform}} + \mathcal{L}_{\text{mask}}. \quad (9)$$

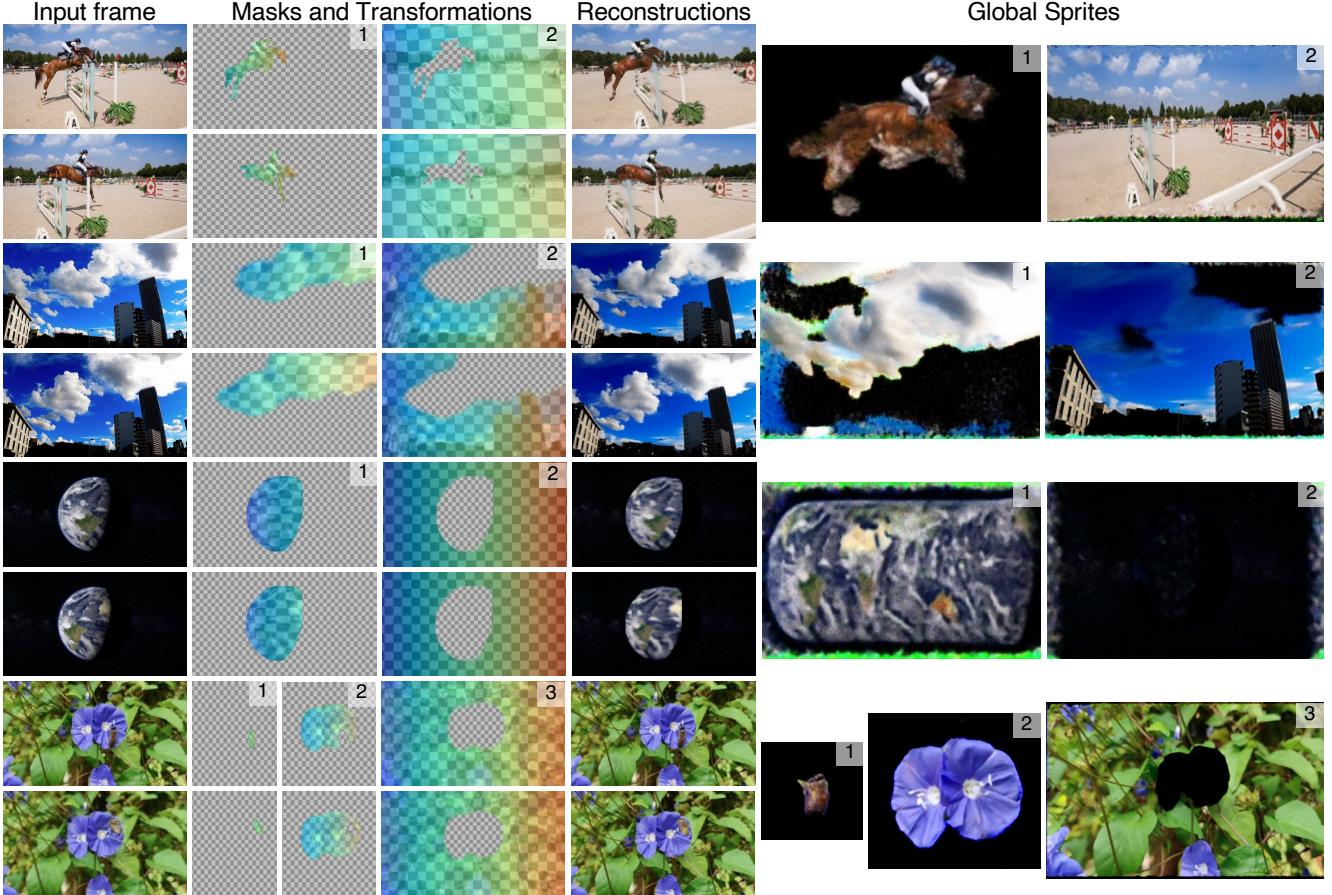
We encourage the transformations to be consistent with the flow between consecutive frames  $t$  and  $t+1$ . We also make the loss scale-invariant to stabilize training and prevent degenerate transforms. Hence we formulate our loss as

$$\mathcal{L}_{\text{transform}} = \sum_{\ell,x} M_t^\ell(x) \cdot \frac{\|T_t^\ell(x) - T_{t+1}^\ell(x')\|}{s_t^{(\ell)} + s_{t+1}^{(\ell)}} \quad (10)$$

where  $x' = F_{t \rightarrow t+1}(x)$ , and  $s_t^{(\ell)}$  is the scale of  $T_t^{(\ell)}$  (see the supplement for details).

Likewise  $\mathcal{L}_{\text{mask}}$  encourages masks to be consistent with flow:

$$\mathcal{L}_{\text{mask}} = \sum_{\ell,x} \|M_t^\ell(x) - M_{t+1}^\ell(x')\|. \quad (11)$$



**Figure 3. Qualitative results on diverse videos.** We show examples with two and three (second row) motion groups and examples with non-traditional foreground objects (bottom two rows). We show two input frames for each example. For each frame, we show the masks and transformations for each motion group (where the number in the corner of the image indicates the group index), as well as the composited reconstruction. Finally, we show the global sprites for each motion group, shared between all frames (with index similarly indicated). We overlay a rainbow checkerboard on the masks to visualize the corresponding sprite texture coordinates. Note for instance the decomposition into bee, flower, and background in the second video, the layer that captures non-object-like cloud regions in the third video, and the aggregation of the globe texture across the full video in the last video. We show full video results and more examples in the supplemental.

**Reconstruction loss.** We use a combination of L1 distance and Laplacian pyramid similarity as our reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \|\hat{I}_t - I_t\|_1 + \mathcal{L}_{\text{edges}}, \quad (12)$$

with  $\mathcal{L}_{\text{edges}} = \sum_m 4^m \|\text{Lap}_m(\hat{I}_t) - \text{Lap}_m(I_t)\|_1$ , where  $\text{Lap}_m$  is the  $m$ -th level of the Laplacian pyramid.

The final loss function is then:

$$\mathcal{L} = \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} + \lambda_{\text{group}} \mathcal{L}_{\text{group}} + \lambda_{\text{warp}} \mathcal{L}_{\text{warp}}. \quad (13)$$

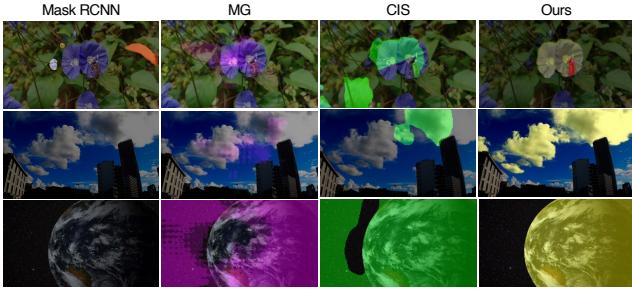
**Implementation details.** We compute optical flow with RAFT [33]. We use UNets [28] for the mask prediction and texture generation models. We use  $\Delta_T = 4$  and  $N_i = 16$ ,  $N_j = \lfloor \frac{16*W}{H} \rfloor$  for the spline knots. We introduce losses to the optimization problem in a schedule: we first warmstart the grouping network with  $\mathcal{L}_{\text{group}}$ . We use these initial rough

masks to initialize the scales and translations of the frame-to-texture transforms for each layer, then add  $\mathcal{L}_{\text{warp}}$  and  $\mathcal{L}_{\text{recon}}$ . Please refer to the supplemental for additional details. Code can be found at the [project website](#).

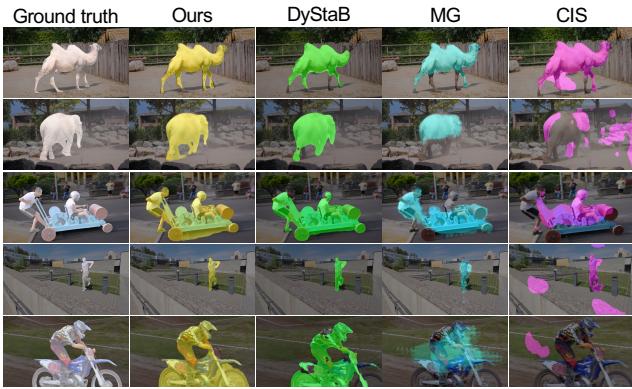
## 5. Results

### 5.1. Qualitative results on real videos

In Figure 3, we show our estimated Deformable Sprites for a variety of real videos. The top row is from the DAVIS dataset [26]; the bottom three are Internet videos. We show results for a video with three sprites (second row), in which both the flower and bee are moving foreground objects, as well as for videos with non-traditional foreground objects, such as clouds (third row) and Earth (bottom row). We show result videos and additional examples in the supplemental.



**Figure 4. Segmentation models on non-traditional objects.** We show predicted segmentations on frames of the bottom three video sequences in Figure 3. We show the masks from MaskRCNN [8], trained on COCO categories, and two recent motion segmentation methods, MG [38] and CIS [40], trained on DAVIS [26]. These methods are unable to handle the out-of-distribution images.



**Figure 5. Qualitative mask comparison with top baselines on DAVIS.** We compare the masks from Deformable Sprites with baselines on DAVIS [26]. We compare with DyStaB [39], MG [38], and CIS [40]. Please see video results in the supplemental.

	DAVIS	FBMS	SegTV2
ARP [14]	76.2	59.8	57.2
ELM [16]	61.8	61.6	-
MG [38]	68.3	53.1	58.2
CIS [40]	71.5	63.5	62.0
DyStaB* [39]	80.0	73.2	74.2
<b>Ours</b>	<b>79.1</b>	<b>71.8</b>	<b>72.1</b>

**Table 1. Quantitative mask evaluation on VOS benchmarks.** We compare masks from Deformable Sprites with top-performing baselines on the DAVIS [26] and SegTrackV2 [17] benchmarks. We achieve IOU ( $\mathcal{J}$ ) scores competitive with current SOTA. \*(Note that DyStaB [39] is trained on a video dataset; all other methods use only the information present in the input video.)

For each example, we show two frames from the input video and our Deformable Sprites representation. Each frame has masks and transforms corresponding to each sprite; we visualize the masks and transforms for each layer together. We overlay a rainbow checkerboard on top of the masks to

	k-means	rigid only	+ splines (full)
DAVIS	64.6	71.5	79.1

**Table 2. Ablations** We compare our full model with the variants *k-means* and *rigid only* on the DAVIS dataset. *k-means* comprises of only a mask prediction network trained with  $\mathcal{L}_{\text{group}}$  and  $\mathcal{L}_{\text{mask}}$ . *rigid only* adds back the sprites, using only homographies as transforms.

show corresponding sprite texture coordinates; we see that the same texture coordinates follow the same points of the object throughout the video. The sprite images are also completed as regions in each layer become visible over the course of the video. The globe example in Figure 3 demonstrates this effect: as the globe rotates, the foreground sprite is completed into an unwrapped world map.

We recover our Deformable Sprites based solely on the motion and appearance cues present in the input, and do not rely on training data. As such, we can easily recover Deformable Sprites for objects that do not fall into common categories. This is in contrast to layer decomposition methods such as [21], [27] or [11], which rely on input masks, either from off-the-shelf pre-trained segmentation models or from user interaction. In Figure 4, we show the masks we obtain from our representation compared to off-the-shelf Mask RCNN, and two recent motion segmentation methods [40], [38], trained on DAVIS. Layer decomposition methods [11], [21] that require input masks would require further user interaction, e.g., using methods such as GrabCut [29] to decompose these scenes.

## 5.2. Motion segmentation results on benchmarks

A key advantage of Deformable Sprites is the ability to discover reasonable motion groups during optimization. In many cases, moving objects in a video are commonly segmented objects, such as animals, peoples, or cars. We compare the quality of our motion groups to other unsupervised motion segmentation baselines on two standard video-object segmentation benchmarks, DAVIS2016 [26] and SegTrackV2 [17]. We report quantitative performance in Table 1, in which our method is the second best in both benchmarks. We show qualitative comparisons with recent motion grouping methods DyStaB [39], CIS [40] and MG [38] in Figure 5. Our representation can better capture the limbs of articulated animals and people, thanks to our spline representation. Finally, we note that while motion group boundaries generally correspond with the semantic object masks, they also often include effects such as reflections and shadows. Sometimes this behavior is desired [21], although these effects are not included in the segmentation task we evaluate on.

## 5.3. Ablations

We ablate the effect of the global appearance model and the deformable transforms on the quality of the output masks



Figure 6. **Ablation.** We ablate our model without global sprites or transformations and trained only with the motion grouping loss (**K-means**), without only a rigid transformation model (**Rigid Only**), and compare with the full model (**w/Spatio Temporal Splines**). Modeling non-rigid motion enables us to capture the twisting dog and the camel’s legs.

on the DAVIS dataset; we report the quantitative differences in Table 2. We remove sprite textures and transformations and train the mask prediction network with the motion grouping loss  $\mathcal{L}_{\text{group}}$  and mask consistency loss  $\mathcal{L}_{\text{mask}}$  (K-means); the coherence of the masks for this variant drops dramatically. We then add back the sprite textures, but only use homographies as transformations (rigid only). We see in Figure 6 that the persistent sprites help in recognizing and segmenting objects when they are stationary. However, homographies cannot model long-term correspondences of deforming objects. We see in Figure 6 that the stationary camel’s legs and dog’s body are still grouped with the foreground, and the corresponding texture is more complete.

#### 5.4. Consistent video editing

Our method enables downstream applications such as creating video effects. In particular, we demonstrate consistent video editing by directly editing our learned foreground and background texture atlases. In Figure 7, we modify the sprite images with style transfer [42] or by adding decals, and consistently propagate edits with our learned texture mappings. Our method can also create 2D motion sculpture effects, as shown in Figure 8, by transforming foreground texture and learned alpha masks for several frames onto the background texture and overlaying them together. We refer readers to our [project website](#) for full video results.

## 6. Discussion and Conclusion

Our approach has a few limitations. One is that our fixed appearance model does not explicitly handle changes in appearance over time, for instance due to lighting (e.g., if a person walks through a region in shadow). Such appearance changes can be modeled in unintended ways, e.g., through clever uses of unused part of the texture or use of blending via the soft masks; adding explicit modeling of appearance changes would be an interesting extension of our method. Because the layer ordering is weakly constrained for non-background elements, when multiple foreground layers are present they may not end up in a natural order. Finally, our method is limited to modeling elements with 2D textures

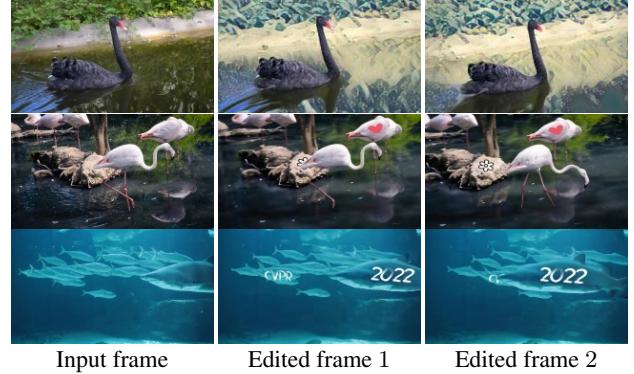


Figure 7. **Consistent video editing.** We can directly edit the recovered textures; the edits are then automatically propagated to the full output video. In the first row, the background is stylized; in the second two rows, decals are added to objects. The last row has two moving foreground groups.

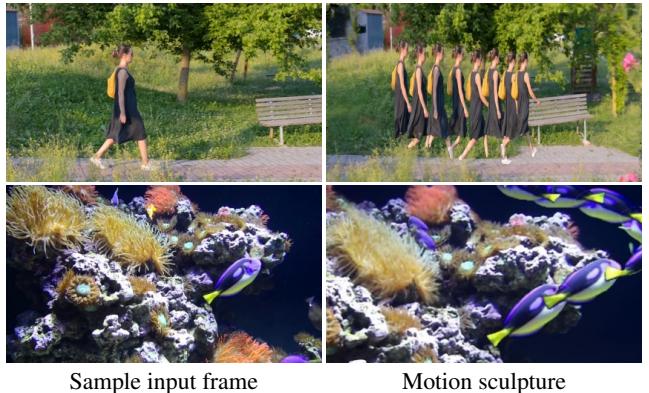


Figure 8. **Motion sculpture.** Our method can produce motion sculptures by compositing the foreground texture at several different times onto the background via the recovered transformations.

and transformations; it would be interesting to extend our approach to true 3D decompositions of scenes.

In summary, we presented a new method for decomposing videos into layers that combines neural representations with classic ideas in video representation, and show the effectiveness of our method across a range of challenging videos, and for applications in video editing.

## References

- [1] Richard H Bartels, John C Beatty, and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995. 5
- [2] Michael J. Black and P. Anandan. Robust dynamic motion estimation over time. In *CVPR*, 1991. 2
- [3] Gabriel J. Brostow and Irfan A. Essa. Motion based decomposing of video. In *ICCV*, 1999. 2
- [4] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 2
- [5] Trevor Darrell and Alex Pentland. Robust estimation of a multi-layered motion representation. In *Proceedings of the IEEE Workshop on Visual Motion*, 1991. 2
- [6] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In *ICCV Workshops*, 2019. 2
- [7] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 5
- [8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 7
- [9] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G. Schwing. Unsupervised video object segmentation using motion saliency-guided spatio-temporal propagation. In *ECCV*, 2018. 3
- [10] Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001. 1, 2
- [11] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. In *SIGGRAPH Asia*, 2021. 2, 7
- [12] Margret Keuper. Higher-order minimum cost lifted multicut for motion segmentation. In *ICCV*, 2017. 2, 4
- [13] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicut. In *ICCV*, 2015. 2
- [14] Yeong Jun Koh and Chang-Su Kim. Primary object segmentation in videos based on region augmentation and reduction. In *CVPR*, 2017. 3, 7
- [15] M. Pawan Kumar, Philip H. S. Torr, and Andrew Zisserman. Learning layered motion segmentations of video. *IJCV*, 76(3), 2008. 2
- [16] Dong Lao and Ganesh Sundaramoorthy. Extending layered models to 3d motion. In *Proceedings of the European conference on computer vision (ECCV)*, pages 435–451, 2018. 2, 7
- [17] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 7
- [18] Ce Liu, Antonio Torralba, William T Freeman, Frédéric Durand, and Edward H Adelson. Motion magnification. *ACM transactions on graphics (TOG)*, 24(3):519–526, 2005. 4
- [19] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020. 3
- [20] 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. 2
- [21] Erika Lu, Forrester Cole, Tali Dekel, Andrew Zisserman, William T. Freeman, and Michael Rubinstein. Omnimatte: Associating objects and their effects in video. In *CVPR*, 2021. 2, 7
- [22] Peter Ochs and Thomas Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, 2011. 2, 4
- [23] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 614–621. IEEE, 2012. 4
- [24] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *TPAMI*, 36(6), 2014. 2, 4
- [25] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 2
- [26] 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. 2, 6, 7
- [27] Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew Fitzgibbon. Unwrap Mosaics: A new representation for video editing . In *SIGGRAPH*, 2008. 1, 2, 7
- [28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 6
- [29] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 7
- [30] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984. 5
- [31] Jianbo Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, 1998. 2
- [32] Dmitriy Smirnov, Michael Gharbi, Matthew Fisher, Vitor Guizilini, Alexei A. Efros, and Justin Solomon. MarioNette: Self-supervised sprite learning. In *NeurIPS*, 2021. 2
- [33] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020. 6
- [34] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *IJCV*, 128(7), 2020. 4
- [35] John Y.A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5), 1994. 1, 2, 4
- [36] Josh Wills, Sameer Agarwal, and Serge Belongie. What went where. In *CVPR*, 2003. 2, 4
- [37] Christopher Xie, Yu Xiang, Zaid Harchaoui, and Dieter Fox. Object discovery in videos as foreground motion clustering. In *CVPR*, 2019. 2
- [38] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *ICCV*, 2021. 3, 7

- [39] Yanchao Yang, Brian Lai, and Stefano Soatto. DyStaB: Unsupervised object segmentation via dynamic-static bootstrapping. In *CVPR*, 2021. [2](#), [3](#), [7](#)
- [40] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *CVPR*, 2019. [2](#), [7](#)
- [41] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *ICLR*, 2022. [2](#)
- [42] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017. [8](#)