Hashing Neural Video Decomposition with Multiplicative Residuals in Space-Time —Supplementary Material—

1. More Experimental Results

This section summarizes our supplementary results with links to the corresponding videos on our project website at https://lightbulb12294.github.io/hashing-nvd/.

1.1. Video Reconstruction

The supplementary material includes additional reconstruction results on the DAVIS dataset [5], specifically, bear, black swan, car-turn, cows, hike, kite-surf, libby, lucia, and parkour, as well as internet videos such as disco ball [6] and sunset [4]. These results are compiled into a single video and are showcased in the Reconstruction section of our project page. We also provide several full-resolution (1080p) video reconstructions, including bear, black swan, car-turn, disco ball, hike, and lucia, which can be found in Reconstruction (1080p) section, with a video 1080p.mp4 showing simple comparisons between 1080p and 480p reconstructions.

1.2. Video Editing

We showcase our video editing results, which can be found in Video Editing section of our project page.

1.3. Camera Motion Manipulation

We manipulate camera motion in different ways. The demonstrations can be found in Camera Motion Control section of our project page.

1.4. Comparison with Previous Work

We present a comparison of our reconstruction results with those obtained using Layered Neural Atlases [3] and Deformable Sprites [8]. The comparison results are included in Compare with Previous Works section of our project page. We also provide a comparison of the alpha masks for the bear video, as we find that Layered Neural Atlases tend to compensate for variations in lighting through the alpha mask, resulting in noisy and inaccurate object masks.

1.5. Multiple Foreground Objects

Our model can handle multiple foreground layers simultaneously. In Fig. 1, we present an example where the pipe and trunk are assigned to one foreground layer while the dog belongs to another. The mask for the pipe and trunk is manually crafted for a single frame, while the remaining frames are generated using pre-computed optical flow, which may result in some inaccuracies in the generated masks. Furthermore, the presence of occlusion between the dog and the pipe and trunk in certain frames poses a challenge for accurate decomposition. Our method successfully decomposes the video into two foreground layers and a background, even in the presence of inaccurate masks and occlusion.

2. Losses Inherited from Previous Work

2.1. Reconstruction Loss

We utilize the squared distance as our reconstruction loss, which is composed of two terms. The first term represents the distance between the ground truth and the predicted color, while the second term accounts for the image gradient.

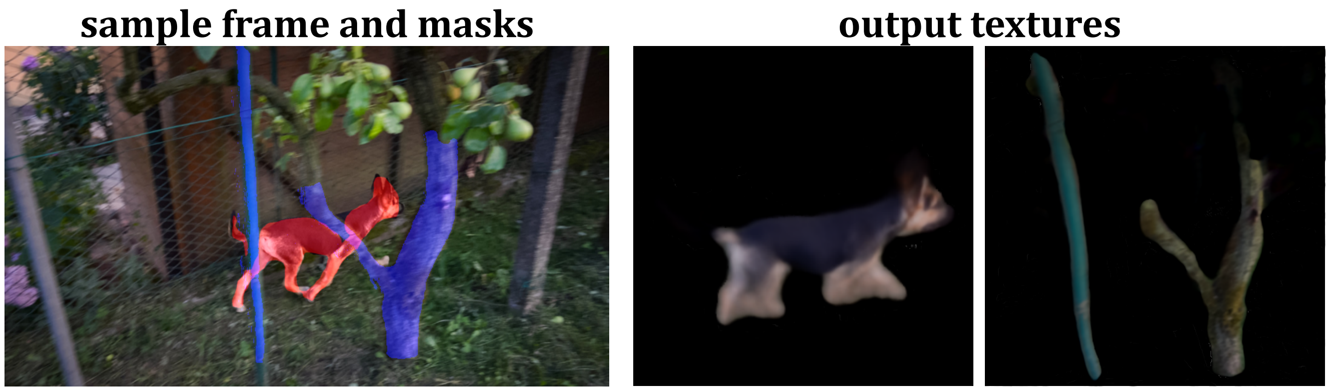

Figure 1: **Representation of multiple foreground objects.** We split two foreground objects for the video. Our method can handle multiple foreground objects, even when one object occludes another.

$$
\begin{aligned}
\mathcal{L}_{\text{RGB}} &= \lambda_{\text{RGB}} \left\| \hat{c}^{(p)} - \bar{c}^{(p)} \right\|_2^2 , \\
\mathcal{L}_{\text{Grad}} &= \lambda_{\text{Grad}} \left(\left\| \hat{\nabla}_x^{(p)} - \bar{\nabla}_x^{(p)} \right\|_2^2 + \left\| \hat{\nabla}_y^{(p)} - \bar{\nabla}_y^{(p)} \right\|_2^2 \right) ,
\end{aligned} \tag{1}
$$

where $\bar{c}^{(p)}$ and $\left(\bar{\nabla}_x^{(p)}, \bar{\nabla}_y^{(p)}\right)$ are the ground truth color and spatial derivatives of pixel p, and $\hat{c}^{(p)}$ and $\left(\hat{\nabla}_x^{(p)}, \hat{\nabla}_y^{(p)}\right)$ are our predictions, respectively.

2.2. Sparsity Loss

To avoid the presence of duplicate foreground objects in the texture area, we incorporate a sparsity loss inspired by Layered Neural Atlases [3]. For a given pixel p in a video frame, if p is mapped to the background, it is invisible for any other foreground objects. Consequently, the color value of p in the invisible area should be purely black, devoid of any foreground information. Hence, we incentivize the color of p to be black through the sparsity loss:

$$
\mathcal{L}_{\text{sparsity}} = \lambda_{\text{sparsity}} \sum_{i=0}^{N-1} \left\| (1-\alpha_i) c_i \right\|^2 . \tag{2}
$$

2.3. Optical Flow Loss

To ensure consistent mapping between points in the scene and corresponding points on the texture, we utilize a pre-trained optical flow model [7] and incorporate the optical flow loss from the previous approach [3]. This loss encourages the predicted optical flow to align with the ground truth optical flow, enabling accurate mapping between the scene and texture coordinates. In particular, for consecutive frames at time i and $i+1$, we aim to achieve consistency in both the alpha and color values using optical flow as follows:

$$
\begin{aligned}
\mathcal{L}_{\text{of-c}} &= \lambda_{\text{of-c}} \sum_{i=0}^{N} \alpha_i^{(p)} \left\| \mathcal{M}_i(p) - \mathcal{M}_i(p') \right\| , \\
\mathcal{L}_{\text{of-}\alpha} &= \lambda_{\text{of-}\alpha} \sum_{i=0}^{N} \left| \alpha_i^{(p)} - \alpha_i^{(p')} \right| ,
\end{aligned} \tag{3}
$$

where $\mathcal{M}_i$ represents the mapping network of layer i, and p' is the corresponding point of p in either backward or forward direction.

2.4. Alpha Bootstrapping Loss

Given that our model lacks prior knowledge about the objects present in the scene, we leverage the use of coarse masks obtained from Mask-RCNN [2] as initial guidance for our model.

$$\mathcal{L}_{\text{bootstrap}} = \lambda_{\text{bootstrap}} \text{BCE}\left(m, \alpha\right) \,, \tag{4}$$

where m is the coarse mask. The loss would be deactivated after a period of training.

3. Implementation Details

Our standard experimental setup involves videos comprising 50 to 100 frames with a resolution of 768×432. We sample 10000 points for each iteration to train our model and train it for a total of 50000 iterations. We use RAFT [7] to compute the optical flow and Mask R-CNN [1] to generate an initial coarse object mask for every frame. The entire training process takes approximately 20–40 minutes which requires 3GB of GPU RAM and can process 71 frames per second during inference for both edited and unedited videos using an NVIDIA RTX 3090 Ti GPU.

To speed up the inference process, we cache the UV coordinates and alpha masks for all frames, as well as the textures that are independent of time. This allows users to modify the textures and view the edited results in real-time without any delay.

We follow Layered Neural Atlases [3] to set the loss hyperparameters but divide the values by 1000 to stabilize the training procedure. Additionally, we adopt alpha bootstrapping and pre-training stages to further improve our results. We set $\lambda_{\mathcal{R}\text{con}} = 0.1$, $\lambda_{\mathcal{R}\text{reg}} = 0.5$, and $\lambda_{\alpha\text{reg}} = 0.1$ for the hyperparameters of loss terms. As the video may include a large range of backgrounds, one might need to set the UV mapping scale to a lower value for different videos. In our experiments, we set the scale to 0.6. In our design, we prevent the gradient of the multiplicative residual from backpropagating to the mapping network; otherwise, it would lower the quality in the bootstrapping stage and is unstable, which might degrade the reconstruction quality.

We sample another batch of patches only from the edges in the video for computing the residual consistency loss. Once an edge patch centered at (x, y, t_1) is sampled, we randomly choose 15 more frames as t_2 to calculate the residual consistency loss. We then mask out all (u, v, t_2) that are visible at time t_2. Note that the gradient is not backpropagated to t_1, since t_1 serves as the supervision for all t_2. We may consider normalizing the raw data before computing the correlation, as in some cases, the computed standard deviation and covariance may be too small.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 3

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, 2020. 3

[3] 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] lukasbieri. Dubrovnik sunset sea city timelapse time lapse. source: https://pixabay.com/videos/dubrovnik-sunset-sea-city-12866/. 1

[5] 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. 1

[6] I see you so. Lamp shines on disco ball. beautiful light in room. interior details. rotating mechanism with backlight. lamp is spinning. source: https://www.vecteezy.com/video/11651553. 1

[7] 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. 2, 3

[8] 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