Omnimatte: Associating Objects and Their Effects in Video
Supplementary Material

1. Formulation Details

1.1. Flow Reconstruction Weight (Section 3.3, Eq 6)

The confidence at each pixel p is defined as:

$$W_t(p) = W_t^{lr}(p) \cdot W_t^p(p) \cdot M_t(p), \tag{1}$$

where W_t^{lr} is computed using a standard left-right flow consistency error by: $W_t^{lr}(p) = \max(1 - e_t^{lr}(p), 0)$, and e_{lr} is the forward-backward flow error. W_t^p measures photometric error and is given by: $W_t^p = \mathbb{1}_{e_p < \beta}$, where $e_p = \|\mathit{Warp}(I_{t+1}; F_{t,t+1}) - I_t\|_1$ measures the photometric difference between I_t and I_{t+1} when backward warped using the flow, and $\beta = 20$. Finally, we mask the regions outside of the object mask M_t since semi-transparent effects such as shadows tend to have inaccurate flow.

1.2. Warp and Brightness Adjustment (Section 3.4)

To compensate for minor camera stabilization errors, we apply the learnable warp at each pixel $\mathbf{x}$ of the background layer:

$$\hat{C}_t^0(\mathbf{x}) = C_t^0(\mathbf{G_w}[t, x_v, x_u]) \tag{2}$$

where $\mathbf{G_w}$ is a $n/10 \times 4 \times 7 \times 2$ grid of offset vectors, n is the number of frames in the video, and $[\cdot,\cdot,\cdot]$ denotes trilinear filtering. We additionally apply a learnable coarse brightness scaling to the final composite:

$$\hat{\mathit{Comp}}(\mathcal{L}_t, o_t) = \mathit{Comp}(\mathcal{L}_t, o_t) \cdot \mathbf{G_b}[t, x_v, x_u] \tag{3}$$

where $\mathbf{G_b}$ is a $n/10 \times 4 \times 7$ grid of brightness coefficients. The values of $\mathbf{G_w}$ and $\mathbf{G_b}$ are optimized together with the network parameters.

The effect of this adjustment layer can be seen in Fig. 1. Adding the background adjustment significantly increases the sparsity of the alpha mattes for the same reconstruction quality.

1.3. RGBA Detail transfer (Section 3.5)

We adopt the same detail transfer step as in Lu, et al. [5] to produce high-resolution omnimattes by transferring detail from the original frame to the CNN outputs. We first compute the residual between the CNN output and the original frame, and determine the amount of the residual to transfer to each RGBA layer using the transmittance map τ_t^i for layer i at time t:

$$\tau_t^i = 1.0 - \mathit{Comp}_\alpha(\mathcal{L}_t \setminus \{L_t^j \mid j < i\}, o_t \setminus \{j \mid j < i\}) \tag{4}$$

where Comp_α denotes the alpha channel of the composite produced by the network. The final layer colors with the transferred detail are:

$$C_t^i = \mathit{Cnr}_t^i + \tau_t^i(I_t - \mathit{Comp}(\mathcal{L}_t, o_t)) \tag{5}$$

where Cnr is the color produced by the network.

2. Implementation Details

Network Architecture We adopt the same network architecture as in Lu, et al. [5], with the exception of replacing batchnorm with instance norm:

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

'IN' refers to instance normalization, 'convt' refers to convolutional transpose. All convolutions are 4×4. Additionally, we have a convolutional layer following layer 12 which outputs 2 channels for optical flow.

Training Details We implement our network in JAX [1] and Haiku [2]. We optimize our full objective (Eq. 2, Section 3.1), with relative weights: $\lambda_\text{r} = .005, \lambda_\text{m} = 50$ until $\mathbf{E}_\text{mask}$ falls below 0.05, and is 0 afterward, and $\lambda_\text{w} = .005$. We use the Adam optimizer [3] with an initial learning rate of 1e-3. We train each video with a batch size of 32 on

Google Cloud v3-8 TPU for 2000 epochs. We resize all videos to 256×448 spatial resolution. Training the 'Bear' sequence from the DAVIS dataset [6], which has 82 frames and 1 output omnimatte, takes 2 hours.

2.1. Video effects details (Section 4.6)

All the video effects were created in *postprocessing* in NUKE [4], a standard video compositing package. That is, we produce the effects by editing our output omnimattes (and using the camera homographies), whereas the omnimatte network model itself is not used.

Color Pop: to create the "color pop" effect, we create two versions of the original video, one with color desaturated and one with color amplified. We blend the amplified version over the desaturated version using the alpha matte from the foreground layer.

Background Replacement: replacing the background while preserving camera motion is accomplished by treating the new image as a new extended canvas. Thus, we create a new time-varying background frames C_t^0 by applying the original camera homographies.

Stroboscopic Photograph: to create the background of the stroboscopic photograph, we apply the inverse camera homographies to the background color images C_t^0 and accumulate them into the canvas space with over compositing. This step creates a clean background canvas without the foreground objects. We then apply the same inverse transformations to the foreground layers and composite them on top. For the "horsejump-low" photo, we picked 15 frame intervals.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. 1

[2] Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. 1

[3] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014. 1

[4] The Foundry Visionmongers Ltd. Nuke, 2018. 2

[5] 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. 1

[6] 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. 2

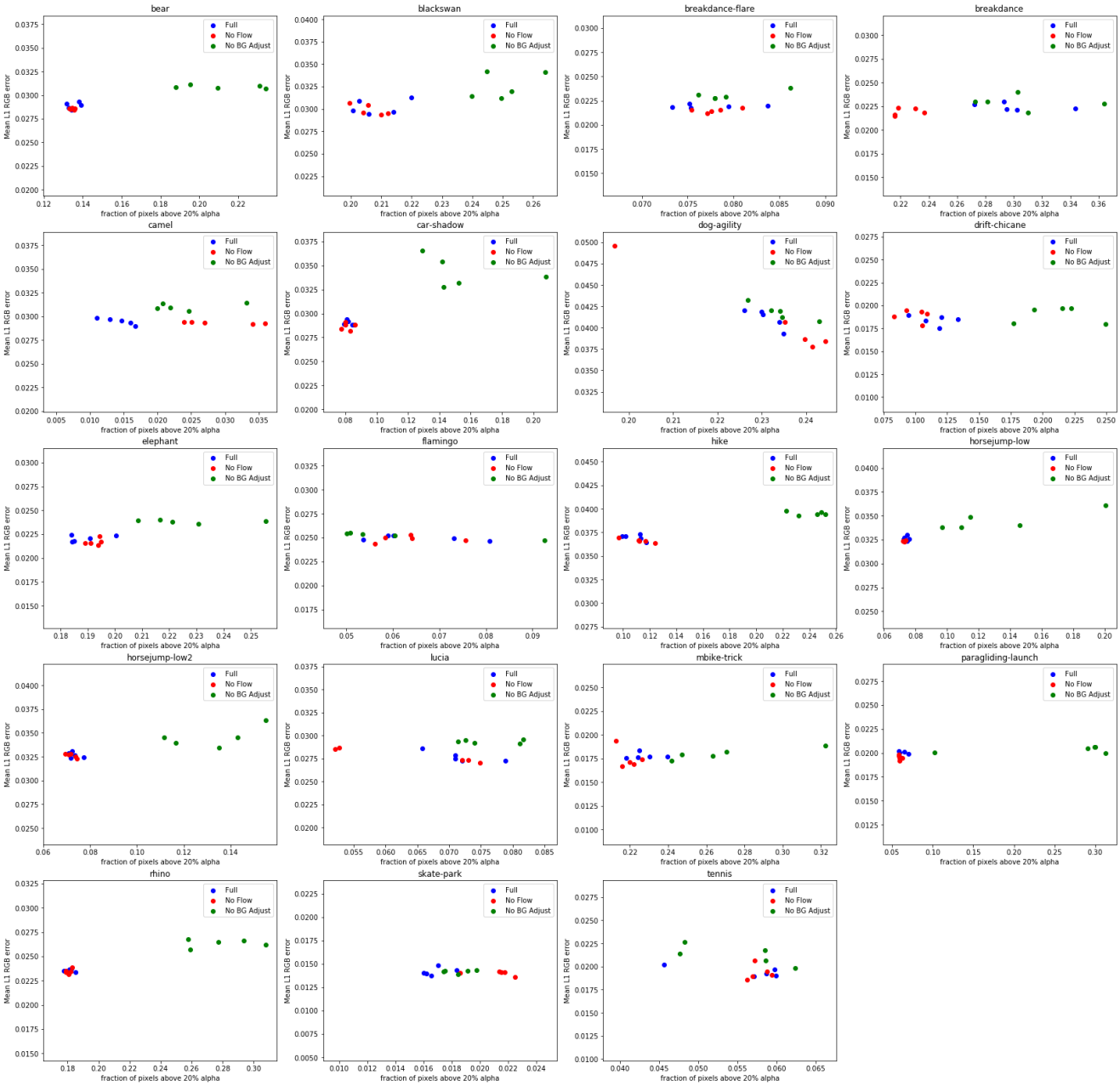

Figure 1. **Reconstruction error vs. matte sparsity for multiple runs on DAVIS.** Each graph measures the L1 reconstruction error against the fraction of pixels above 20% alpha. Lower is better for both axes. A pixel above 20% alpha is considered "visible," so this value gives a measurement of amount of clutter in the matte. Each video is run with 5 random seeds for each of 3 ablation conditions: full method, no flow input or flow loss ("No Flow"), and no background offset or brightness adjustment ("No BG Adjust"). Note that "No BG Adjust" tends to produce more matte clutter, especially for videos with considerable camera motion (e.g. "horsejump-low", "hike").