Background Matting: The World is Your Green Screen
Supplementary Material

Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman

University of Washington

1. Overview

We provide additional details and results in this supplementary material. In Sec. 2 we describe the details of our network architecture. In Sec. 3.1 we clarify our choice of automatic trimap generation, especially to show why background subtraction is not good enough for this problem. In Sec. 3.2, we show why algorithms that do not predict foreground F introduce additional artifacts in compositing. In Sec. 4, we provide additional qualitative examples for ablation studies. Specifically, we show the role of Content Switching Block in Sec. 4.1, motion cue in Sec. 4.2, self-supervised adversarial training on real data in Sec. 4.3 and robustness w.r.t. segmentation and background in Sec. 4.4.

Please see Supplementary Video for results.

2. Network Architectures

Our proposed matting network G_{Adobe}, shown again for reference in Figure 1 (same as Figure 2 in the main paper) consists of the Context Switching Block (CS Block) followed by residual blocks (ResBLKs) and decoders to predict alpha matte α and foreground layer F. Below we describe the network architecture in details. Most of our Generator architecture, especially residual blocks and decoders, and Discriminator architecture are based on that of [4].

Generator G_{Adobe}

'Image Encoder' and 'Prior Encoder' (CS Block): C64(k7) - C*128(k3) - C*256(k3)

'CN(kS)' denotes convolution layers with N $S \times S$ filters with stride 1, followed by Batch Normalization and ReLU. 'C*N(kS)' denotes convolution layers with N $S \times S$ filters with stride 2, followed by Batch Normalization and ReLU. The output of 'Image Encoder' layer produces a blob of spatial resolution $256 \times 128 \times 128$. All convolution layers do not have any bias term.

'Selector' (CS Block): C64(k1)

'CN(kS)' denotes convolution layers with N $S \times S$ filters with stride 1, followed by Batch Normalization and ReLU. The 'Selector' block takes as input the concatenation of image feature and a prior feature as a blob of spatial resolution $512 \times 128 \times 128$. The output of the 'Selector' network is a blob of spatial resolution $64 \times 128 \times 128$. The goal of the 'Selector' block is to generate prior features conditioned on the image. This will help the network to generalize from synthetic-composite dataset (on which it was trained) to real images by not over-relying on one kind of features (e.g. color difference with background).

'Combinator' (CS Block): C256(k1)

'CN(kS)' denotes convolution layers with N $S \times S$ filters with stride 1, followed by Batch Normalization and ReLU. The 'Combinator' block takes as input the concatenation of image features of spatial resolution $256 \times 128 \times 128$, along with 3 other prior features from the 'Selector' network of spatial resolution $64 \times 128 \times 128$ each. Thus the input to the 'Combinator' block is of spatial resolution $448 \times 128 \times 128$ and the output is of spatial resolution $256 \times 128 \times 128$. The 'Combinator' block learns to combine the individual priors features with the original image feature for the goal of improving matting.

'ResBLKs': K ResBLK

The output of the combinator is first passed through $K = 7$ 'ResBLK's and then provided as input to 2 separate 'ResBLK's of $K = 3$ for alpha matte and foreground layer separately. All 'ResBLK's operate at a spatial resolution of $256 \times 128 \times 128$. Each 'ResBLK' consists of Conv256(k3) - BN -ReLU - Conv256(k3) - BN, where 'BN' denote Batch Normalization.

'Decoder' for alpha matte α: CU*128(k3)-CU*64(k3)-Co1(k7)-Tanh

The input to the 'Decoder' is of resolution $256 \times 128 \times 128$.'CU*N(kS)' denotes bilinear upsampling by factor of 2, followed by convolution layer with N $S \times S$ filters with stride 1, Batch Normalization and ReLU. The last layer Co3k(7) consists of only convolution layers of 1 7×7 filters, followed by Tanh layer. This scales the output alpha between $(-1, 1)$.

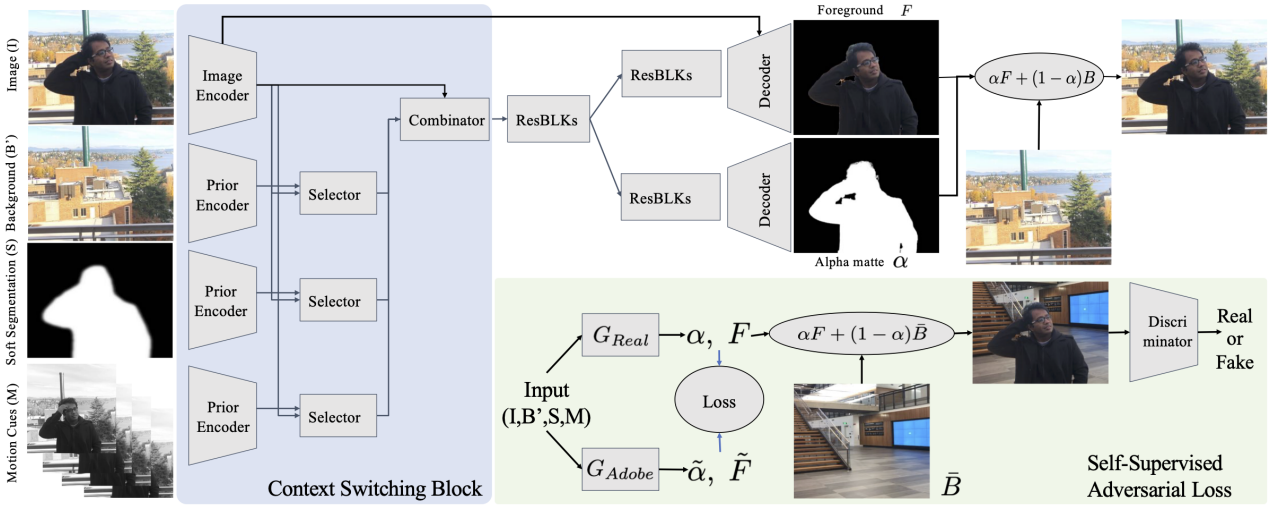

Figure 1. **Overview of our approach.** Given an input image I and background image B', we jointly estimate the alpha matte α and the foreground F using soft segmentation S and motion prior M (for video only). We propose a Context Switching Block that efficiently combines all different cues. We also introduce self-supervised training on unlabelled real data by compositing into novel backgrounds.

'Decoder' for foreground F: CU*128(k3)-CU*64(k3)-Co1(k7)
The input to the 'Decoder' is of resolution $256 \times 128 \times 128$.'CU*N(kS)' denotes bilinear upsampling by factor of 2, followed by convolution layer with N $S \times S$ filters with stride 1, Batch Normalization and ReLU. There is also a skip connection from the image input features of resolution $128 \times 256 \times 256$ which is combined with the output of CU*128(k3) and passed on to CU*64(k3). The last layer Co3k(7) consists of only convolution layers of 1 7×7 filters.

Discriminator D: C*64(k4) - C*I128(k4) - C*I256(k4) - C*I512(k4)
We use 70×70 PatchGAN [2]. 'C*N(kS)' denotes convolution layers with N $S \times S$ filters with stride 2, followed by leaky ReLUs of slope 0.2. 'I' denotes the presense of Instance Norm before leaky ReLU, in all layers except the first one. After the last layer a convolution filter of kernel 4×4 is applied to produce a 1 dimensional output. The PatchGAN is applied over the whole composite image by convolving with every 70×70 patch to determine if it is real or fake.

3. Experimental Evaluation

3.1. Automatic Trimap Generation

We compare our method with algorithms that require user defined trimaps (CAM, IM, BM). It is extremely time consuming to annotate trimaps for every frame of a video, or even for a bunch of keyframes whose trimaps then need to be propagated to the remaining frames and then touched up. As described in the paper, we instead created trimaps automatically by applying segmentation [1], and labelling each pixel with person-class probability > 0.95 as foreground, < 0.05 as background, and the rest as unknown. We tried, and rejected, alternative methods, including background subtraction and erosion-dilateion of the segmentation mask, which we now describe and illustrate here for completeness.

Background subtraction is popularly used for change detection, but is extremely sensitive to color differences and produces *only* a binary matte. Thus it is not in itself a suitable candidate for matting. However background subtraction could in principle be used to generate a trimap. In our experiments, we observed that the best thresholds varied from image to image and even then produced mediocre results (see hand-tuned example in Figure 3). We also tried erosion-dilation of the segmentation mask (erode 5 steps, dilate 15 steps) to try to produce a fixed width 'unknown' band but we often ended up with mattes that pulled in the background as a part of the foreground. Figure 3 shows examples of trimaps and resulting alpha mattes using erosion-dilation, hand-tuned thresholding of background subtraction, and our probability-thresholded 'Automatic Trimap' method.

3.2. Predicting Foreground layer F

To produce composites – the primary reason for extracting a matte in the first place – we require both α and F. Since IM and LFM do not estimate F, we are still left with the task of estimating it ourselves. Why is this non-trivial? From the matting equation ($I = \alpha F + (1-\alpha)B$) after estimating only α, we can say that observed pixel color I must be α of the way along a line segment from B to F. Clearly,

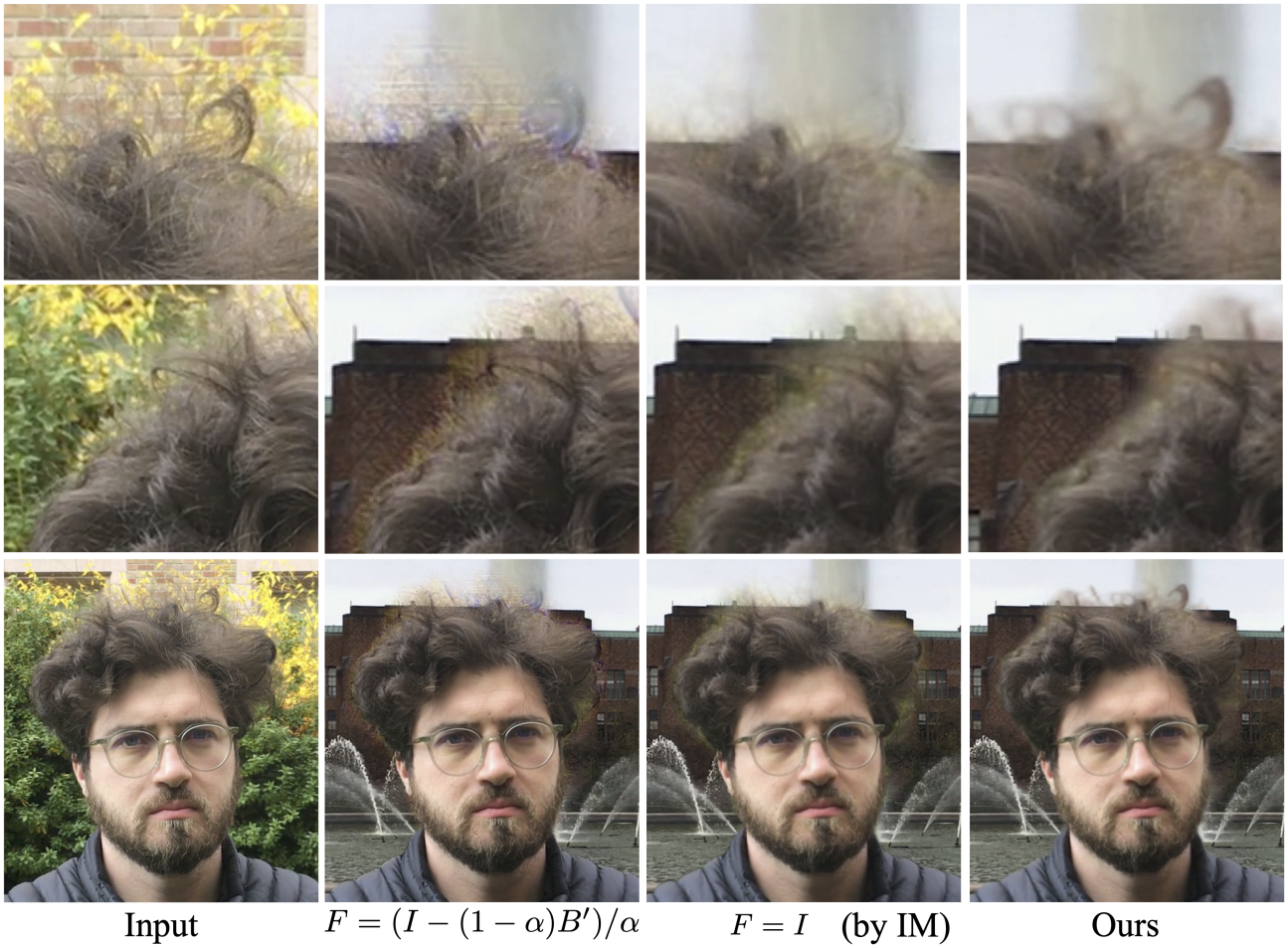

Figure 2. **Choice of Foreground layer.** For baseline algorithms, IM and LFM, that do not predict the foreground layer F, we observe that $F = I$ produces less visible artifacts compared to predicting F from the matting equation using the captured background B'. Notice how some of the brick texture creeps into the foreground when solving for F with the matting equation. We also show that our approach, which jointly estimates F and α, produces less artifacts in compositing.

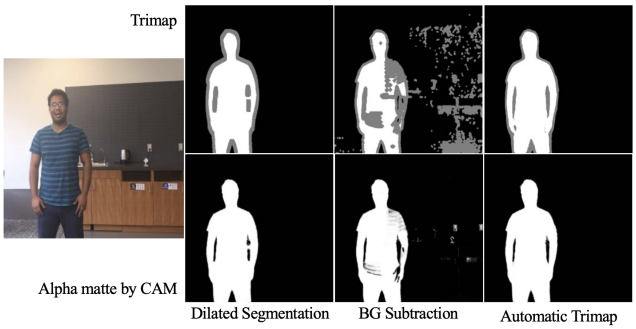

Figure 3. **Automatic Trimap generation.** Our choice of automatic trimap generation from the probability estimates of the segmentation network performs better than background subtraction or erosion-dilation of the segmentation mask.

there is an infinite family of B's and F's that satisfy this constraint; thus, given just I and α, we cannot readily infer F. Our seemingly naive solution is to set $F = I$ for these methods. We also tried estimating F directly from the matting equation given B' – i.e., $F = (I - (1-\alpha)B')/\alpha$ when $\alpha \neq 0$, with F clamped so that each color channel is in $[0, 1]$ – but the results were worse than $F = I$, largely due to discrepancies between B' and B, particularly in the hand-held camera case, where small misalignments can arise. We show a comparison of these two options for IM matting, plus a comparison to our result, in Figure 2. The figure shows that matting-equation based estimation of F pulls some of the background texture into the matte. The $F = I$ solution is better, but picks up some of the background colors (a bit of green and yellow in this case), since it is just copying foreground-background mixed pixels and blending them over another background. Our method, which estimates F directly, shows fewer artifacts. Our matte is a bit softer, but it captures structure like the curls on the top of the

Figure 4. **Role of CS Block.** When foreground color coincides with the background color, Context Switching Block utilizes soft segmentation to predict the correct matte. 'No Context Switching' produces holes when foreground color matches strongly with the background.

head; further, some of the apparent sharpness of the other composites comes from copying over too much of the original image rather (which has detail) rather than fully separating F from the background.

3.3. Results on Real Data

After conducting the user study, we realized that we had given a slight advantage to our method by retaining only the largest $\alpha > 0$ connected component for our background matting approach but not for the competing approaches. We noted that the CAM and IM methods had small "floaters" after matting, so applied the connected component removal to these videos and re-ran the study. We did not observe that BM had floaters in our examples; any that appeared to remain, were actually connected to the largest component by "bridges" of small α. LFM was more problematic. We found that LFM would at times pull in pieces of the background that were larger than the foreground person; the result of retaining the largest connected component would then mean losing the foreground subject altogether, an extremely objectionable artifact. Rather than continuing to refine the post-process for LFM, we simply did not apply a post-process for its results. As seen in the videos, LFM, in any case, had quite a few other artifacts that made it not competitive with the others.

Table 1 and 2 shows the result of the updated user study. We observe that the results are similar to the ones reported in the main paper (i.e., Tables 2 and 3 in the main paper); the excess connected components for CAM and IM did not have a significant impact relative to other errors in matte estimation.

Ours vs.	much better	better	similar	worse	much worse
BM	52.9%	41.4%	5.7%	0%	0%
CAM	40.8%	36.7%	19.2%	3.3%	0%
IM	25.8%	52.5%	18.4%	2.5%	0.8%
LFM	72.0%	20.0%	4.0%	3.0%	1%

Table 1. User study on 10 real world videos (fixed camera).

Ours vs.	much better	better	similar	worse	much worse
BM	61.0%	31.0%	3.0%	4.0%	1.0%
CAM	45.0%	35.0%	5.0%	5.0%	10.0%
IM	34.2%	46.6%	6.7%	2.5%	10.0%
LFM	65.7%	27.1%	4.3%	0%	2.9%

Table 2. User study on 10 real world videos (handheld).

4. Ablation Study

In this section, we provide additional details and more results for the ablation studies already presented in the main

paper. Specifically, we analyze (i) Role of Context Switching Block (ii) Role of motion cues and (iii) Compare 'Ours-Real' to 'Ours-Adobe'.

4.1. Role of Context Switching Block

Here we go in more depth on the paper's CS block ablation study, again showing that the CS Block network is largely effective in utilizing all cues and in generalizing better from the synthetic-composite Adobe dataset [3] to real world images. To this end we train G_{Adobe} with CS Block on the Adobe dataset which we term as 'Ours-Adobe' (with Context Switching). Additionally, we construct another network G_{concat} (No Context Switching), where we remove the CS block. The input to this network is the concatenation of the input image, background, soft segmentation, and motion cues $\{I, B', S, M\}$, which is passes through the 'Image Encoder' architecture to produce a $256 \times 128 \times 128$ dimensional feature. Since there is no CS Block, we directly pass this feature to the ResBLKs and then continue through the same architecture presented in Figure 1. We also train this network on the Adobe dataset, following the exact protocol of training 'Ours-Adobe'.

We then test both G_{Adobe} and G_{concat} on our real video dataset. Note that for this experiment we use the motion cue $M = \{I_{-2T}, I_{-T}, I_{+T}, I_{+2T}\}$ for both G_{Adobe} and G_{concat}. We captured videos at 60fps and set $T = 20$ frames.

In Figure 4 of this document and in Figure 4 of the main paper, we show multiple examples from different videos where a part of the foreground person matches the background color. G_{concat} ('No Context Switching') fails in these situations, this is because while training on the synthetic-composite dataset it learns to focus too much on the color differences to perform matting and fails when colors coincide. On the other hand G_{Adobe} ('With Context Switching') handles color coincidences better by utilizing the soft segmentation cues provided in the input. Thus the CS Block learns to properly utilize the cues at hand, when compared to 'No Context Switching', which focuses more on the color differences to produce a matte. Note that, the holes produced by 'No Context Switching' as shown in Figure 4 appears in multiple frames of that video where the color coincides significantly; we show only 1 sample from all of these frames.

4.2. Role of Motion Cues

Here we provide more detail and examples on the paper's ablation study for motion cues. When the input is video, we have the option of setting the motion cue to $M = \{I_{-2T}, I_{-T}, I_{+T}, I_{+2T}\}$, where T=20 for a 60fps video. We train another network G_{still}, by removing the motion cue input and its related 'Prior Encoder' and 'Selector' from the architecture presented in Figure 1. Thus G_{still} is the same as G_{Adobe}, but without the motion cue block. We train both G_{Adobe} and G_{still} on the synthetic-composite Adobe dataset, following the same training protocol as described in the paper, for both networks. We then test both G_{Adobe} and G_{concat} on our real video dataset.

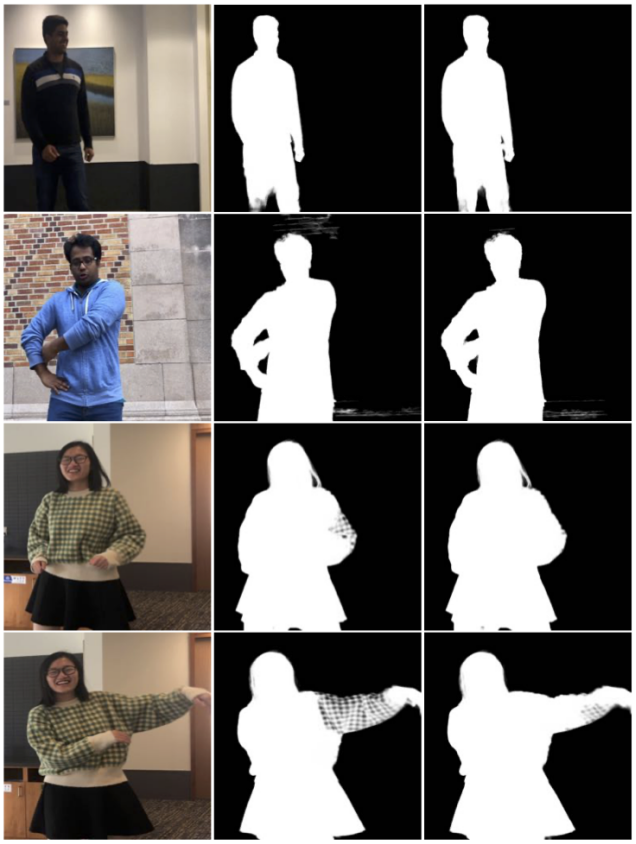

Figure 5. **Role of Motion Cues.** Motion cue helps in predicting better matte when foreground color coincides with the background and foreground moves in front of the background.

In Figure 5 of this document and in Figure 5 of the main paper, we show the role motion cue plays in improving the alpha matte estimation. Specifically, the motion cue helps when foreground color matches the background, and when the foreground is moving significantly. Due to the foreground motion, the network can utilize additional frames (4 in this case) to determine that the regions which move are more likely to be foreground than the background, even though the color matches with the background. Note that, this may not be always true, e.g. a shadow cast on the background also moves with the foreground. Small camera motion with a handheld camera can also effectively cause motion in the background due to misregistration. Additionally, since we consider only a small time window of 1.33 secs for the motion cue, often there is lack of information as the foreground appears to be almost static during that time.

To reiterate: whenever comparing to competing methods, we set the motion cue to $M = \{I, I, I, I\}$ and treat all images (including video frames) independently.

4.3. 'Ours-Real' vs 'Ours-Adobe'

Here we show more comparisons between using just the $G\mathrm{Adobe}$ network for matting ('Ours-Adobe') and using the full network G_{Real} guided by G_{Adobe} and discriminator D ('Ours-Real'). In the main paper, we present an ablation study comparing 'Ours-Real' with 'Ours-Adobe' as both a user study in Table 3 and with qualitatively comparisons in Figure 6. Additional visual comparison on our test videos are presented in our project webpage. In Figure 6 of this document, we provide additional qualitative comparison between 'Ours-Real' and 'Ours-Adobe'. We find that 'Ours-Real' is generally better though on occasion it is not; (k) and (l) are instances where 'Ours-Real' produces an inferior matte compared to 'Ours-Adobe'.

4.4. Role of background and segmentation

The captured background image without the subject and the estimated soft segmentation map are two key additional inputs that helps in estimating the foreground and alpha matte. We found that omitting the background image from the baseline G_{Adobe} model degrades results substantially: SAD error of 8.33 without background vs. 1.73 with background on synthetic-composite Adobe dataset. For backgrounds that are relatively distant or roughly planar, homography-based alignment is accurate, and the network learns to handle remaining mis-registrations by training on hand-held videos. Alignment fails when the background, e.g., has two planes (Fig 4f, with two orthogonal walls).

Soft segmentation is obtained by eroding and dilating the segmentation predicted by Deeplabv3+. We observe that eroding and dilating the segmentation by 20 steps only increase the SAD error from 1.73 to 1.76 on synthetic-composite Adobe dataset. Hence our method is quite robust to errors in segmentation, and soft segmentation only indicates which is the foreground subject in the image. In contrast, the captured background plays more crucial role in the performance of the method.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[3] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2970–2979, 2017.

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Figure 6. **Ours-Real vs Ours-Adobe.** 'Ours-Real' is trained on real data guided by 'Ours-Adobe' (trained on synthetic-composite dataset) along with an adversarial loss. (k) and (l) are instances where 'Ours-Real' produces worse result compared to 'Ours-Adobe'.