Real-Time High-Resolution Background Matting

Shanchuan Lin* Andrey Ryabtsev* Soumyadip Sengupta
Brian Curless Steve Seitz Ira Kemelmacher-Shlizerman
University of Washington
{linsh,ryabtsev,soumya91,curless,seitz,kemelmi}@cs.washington.edu

Zoom input and background shot | Zoom with new background | Our Zoom plugin with new background

Figure 1: Current video conferencing tools like Zoom can take an input feed (left) and replace the background, often introducing artifacts, as shown in the center result with close-ups of hair and glasses that still have the residual of the original background. Leveraging a frame of video without the subject (far left inset), our method produces real-time, high-resolution background matting without those common artifacts. The image on the right is our result with the corresponding close-ups, screenshot from our Zoom plugin implementation.

Abstract

We introduce a real-time, high-resolution background replacement technique which operates at 30fps in 4K resolution, and 60fps for HD on a modern GPU. Our technique is based on background matting, where an additional frame of the background is captured and used in recovering the alpha matte and the foreground layer. The main challenge is to compute a high-quality alpha matte, preserving strand-level hair details, while processing high-resolution images in real-time. To achieve this goal, we employ two neural networks; a base network computes a low-resolution result which is refined by a second network operating at high-resolution on selective patches. We introduce two large-scale video and image matting datasets: VideoMatte240K *and* PhotoMatte13K/85. *Our approach yields higher quality results compared to the previous state-of-the-art in background matting, while simultaneously yielding a dramatic boost in both speed and resolution. Our code and data is available at* https://grail.cs.washington.edu/projects/background-matting-v2/

1. Introduction

Background replacement, a mainstay in movie special effects, now enjoys wide-spread use in video conferencing tools like Zoom, Google Meet, and Microsoft Teams. In addition to adding entertainment value, background replacement can enhance privacy, particularly in situations where a user may not want to share details of their location and environment to others on the call. A key challenge of this video conferencing application is that users do not typically have access to a green screen or other physical props used to facilitate background replacement in movie special effects.

While many tools now provide background replacement functionality, they yield artifacts at boundaries, particularly in areas where there is fine detail like hair or glasses (Figure 1). In contrast, traditional image matting methods [6, 16, 17, 30, 9, 2, 7] provide much higher quality results, but do not run in real-time, at high resolution, and frequently require manual input. In this paper, we introduce the first fully-automated, real-time, high-resolution matting technique, producing state-of-the-art results at 4K (3840×2160) at 30fps and HD (1920×1080) at 60fps. Our method relies on capturing an extra background image to compute the alpha matte and the foreground layer, an approach known as background matting.

Designing a neural network that can achieve real-time matting on high-resolution videos of people is extremely challenging, especially when fine-grained details like strands of hair are important; in contrast, the previous state-of-the-art method [28] is limited to 512×512 at 8fps. Training a deep network on such a large resolution is extremely slow and memory intensive. It also requires large volumes of images with high-quality alpha mattes to generalize; the publicly available datasets [33, 25] are too limited.

*Equal contribution.

Since it is difficult to collect a high-quality dataset with manually curated alpha mattes in large quantities, we propose to train our network with a series of datasets, each with different characteristics. To this end, we introduce VideoMatte240K and PhotoMatte13K/85 with high-resolution alpha mattes and foreground layers extracted with chroma-key software. We first train our network on these larger databases of alpha mattes with significant diversity in human poses to learn robust priors. We then train on publicly available datasets [33, 25] that are manually curated to learn fine-grained details.

To design a network that can handle high-resolution images in real-time, we observe that relatively few regions in the image require fine-grained refinement. Therefore, we introduce a base network that predicts the alpha matte and foreground layer at lower resolution along with an error prediction map which specifies areas that may need high-resolution refinement. A refinement network then takes the low-resolution result and the original image to generate high-resolution output only at select regions.

We produce state-of-the-art background matting results in real-time on challenging real-world videos and images of people. We will release our VideoMatte240K and PhotoMatte85 datasets and our model implementation.

2. Related Work

Background replacement can be achieved with segmentation or matting. While binary segmentation is fast and efficient, the resulting composites have objectionable artifacts. Alpha matting can produce visually pleasing composites but often requires either manual annotations or a known background image. In this section, we discuss related works that perform background replacement with segmentation or matting.

Segmentation. The literature in both instance and semantic segmentation is vast and out of scope for this paper, so we will review the most relevant works. Mask RCNN [11] is still a top choice for instance segmentation while DeepLabV3+ [5] is a state-of-the-art semantic segmentation network. We incorporate the Atrous Spatial Pyramid Pooling (ASPP) module from DeepLabV3 [4] and DeepLabV3+ within our network. Since segmentation algorithms tend to produce coarse boundaries especially at higher resolutions, Kirillov *et al.* presented PointRend [15] which samples points near the boundary and iteratively refines the segmentation. This produces high-quality segmentation for large image resolutions with significantly cheaper memory and computation. Our method adopts this idea to the matting domain via learned refinement-region selection and a convolutional refinement architecture that improves the receptive field. Specific applications of human segmentation and parsing have also received considerable attention in recent works [34, 19].

Trimap-based matting. Traditional (non-learning based) matting algorithms [6, 16, 17, 30, 9, 2, 7] require manual annotation (a trimap) and solve for the alpha matte in the 'unknown' region of the trimap. Different matting techniques are reviewed in the survey by Wang and Cohen [32]. Xu *et al.* [33] introduced a matting dataset and used a deep network with a trimap input to predict the alpha matte. Many recent approaches rely on this dataset to learn matting, e.g., Context-Aware Matting [13], Index Matting [21], sampling-based matting [31] and opacity propagation-based matting [18]. Although the performance of these methods depends on the quality of the annotations, some recent methods consider coarse [20] or faulty human annotations [3] to predict the alpha matte.

Matting without any external input. Recent approaches have also focused on matting humans without any external input. Portrait matting without a trimap [36, 29] is one of the more successful applications due to less variability among portrait images compared to full body humans. Soft segmentation for natural images had also been explored in [1]. Recent approaches like Late Fusion Matting [35] and HAttMatting [25] aim to solve for the alpha matte directly from the image, but these approaches can often fail to generalize as shown in [28].

Matting with a known natural background. Matting with known natural background had been previously explored in [24], Bayesian matting [7] and Poisson matting [30, 10] which also requires a trimap. Recently Sengupta *et al.* [28] introduced Background Matting (BGM) where an additional background image is captured and it provides a significant cue to predict the alpha matte and the foreground layer. Although this method showed high-quality matting results, the architecture is limited to 512×512 resolution and runs only at 8fps. In contrast, we introduce a real-time unified matting architecture that operates on 4K videos at 30fps and HD videos at 60fps, and produces higher quality results than BGM.

3. Our Dataset

Since it is extremely difficult to obtain a large-scale, high-resolution, high-quality matting dataset where the alpha mattes are cleaned by human artists, we rely on multiple datasets including our own collections and publicly available datasets.

Publicly available datasets. The Adobe Image Matting (AIM) dataset [33] provides 269 human training samples and 11 test samples, averaging around 1000×1000 resolution. We also use a humans-only subset of Distinctions-646 [25] consisting of 362 training and 11 test samples, averaging around 1700×2000 resolution. The mattes were created manually and are thus high-quality. However 631 training images are not enough to learn large variations in human poses and finer details at high resolution, so we in-

(a) VideoMatte240K

(b) PhotoMatte13K/85

Figure 2: We introduce two large-scale matting datasets containing 240k unique frames and 13k unique photos.

troduce 2 additional datasets.

VideoMatte240K. We collect 484 high-resolution green screen videos and generate a total of 240,709 unique frames of alpha mattes and foregrounds with chroma-key software Adobe After Effects. The videos are purchased as stock footage or found as royalty-free materials online. 384 videos are at 4K resolution and 100 are in HD. We split the videos by 479 : 5 to form the train and validation sets. The dataset consists of a vast amount of human subjects, clothing, and poses that are helpful for training robust models. We are releasing the extracted alpha mattes and foregrounds as a dataset to the public. To our knowledge, our dataset is larger than all existing matting datasets publicly available by far, and it is the first public video matting dataset that contains continuous sequences of frames instead of still images, which can be used in future research to develop models that incorporate motion information.

PhotoMatte13K/85. We acquired a collection of 13,665 images shot with studio-quality lighting and cameras in front of a green-screen, along with mattes extracted via chroma-key algorithms with manual tuning and error repair. We split the images by 13,165 : 500 to form the train and validation sets. These mattes contain a narrow range of poses but are high resolution, averaging around 2000×2500, and include details such as individual strands of hair. We refer to this dataset as PhotoMatte13K. However privacy and licensing issues prevent us from sharing this set; thus, we also collected an additional set of 85 mattes of similar quality for use as a test set, which we are releasing to the public as PhotoMatte85. In Figure 2 we show examples from the VideoMatte240K and PhotoMatte13K/85 datasets.

We crawl 8861 high-resolution background images from Flickr and Google and split them by 8636 : 200 : 25 to use when constructing the train, validation, and test sets. We will release the test set in which all images have a CC license (see appendix for details).

4. Our Approach

Given an image I and the captured background B we predict the alpha matte α and the foreground F, which will allow us to composite over any new background by $I' = \alpha F + (1-\alpha)B'$, where B' is the new background. Instead of solving for the foreground directly, we solve for foreground residual $F^R = F - I$. Then, F can be recovered by adding F^R to the input image I with suitable clamping: $F = \max(\min(F^R + I, 1), 0)$. We find this formulation improves learning, and allows us to apply a low-resolution foreground residual onto a high-resolution input image through upsampling, aiding our architecture as described later.

Matting at high resolution is challenging, as applying a deep network directly incurs impractical computation and memory consumption. As Figure 4 shows, human mattes are usually very sparse, where large areas of pixels belong to either background ($\alpha = 0$) or foreground ($\alpha = 1$), and only a few areas involve finer details, e.g., around the hair, glasses, and person's outline. Thus instead of designing one network that operates on high-resolution images, we introduce two networks; one operates at lower-resolution and another only operates on selected patches at the original resolution based on the prediction of the previous network.

The architecture consists of a base network G_{base} and a refinement network G_{refine}. Given the original image I and the captured background B, we first downsample by a factor of c to I_c and B_c. The base network G_{base} takes I_c and B_c as input and predicts coarse-grained alpha matte α_c, foreground residual F_c^R, an error prediction map E_c, and hidden features H_c. Then, the refinement network G_{refine} employs H_c, I, and B to refine α_c and F_c^R only in regions where the predicted error E_c is large, and produces alpha α and foreground residual F^R at the original resolution. Our model is fully-convolutional and is trained to work on arbitrary sizes and aspect ratios.

4.1. Base Network

The base network is a fully-convolutional encoder-decoder network inspired by the DeepLabV3 [4] and DeepLabV3+ [5] architectures, which achieved state-of-the-art performance on semantic segmentation tasks in 2017 and 2018. Our base network consists of three modules: Backbone, ASPP, and Decoder.

We adopt ResNet-50 [12] for our encoder backbone, which can be replaced by ResNet-101 and MobileNetV2 [27] to trade-off between speed and quality. We adopt the

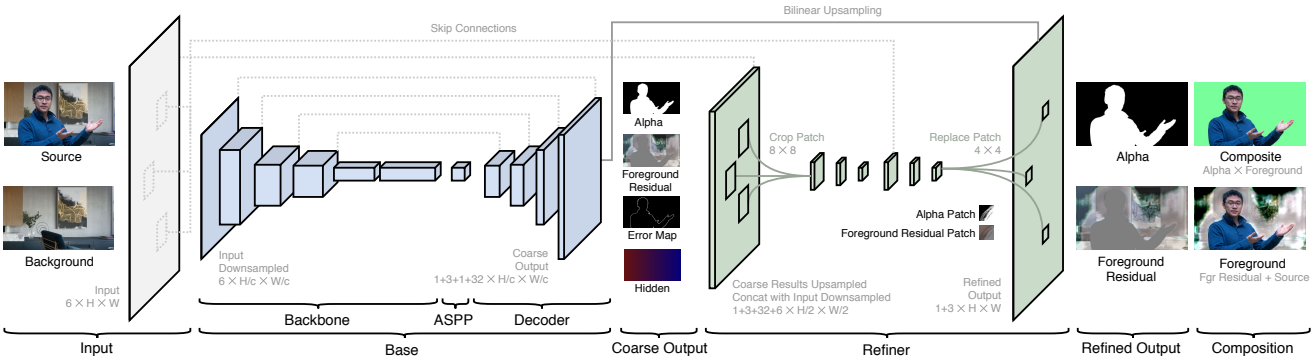

Figure 3: The base network G_{base} (blue) operates on the downsampled input to produce coarse-grained results and an error prediction map. The refinement network G_{refine} (green) selects error-prone patches and refines them to the full resolution.

(a) Coarse (b) Selection (c) Refined

Figure 4: We only refine on error-prone regions (b) and directly upsample the rest to save computation.

ASPP (Atrous Spatial Pyramid Pooling) module after the backbone following the DeepLabV3 approach. The ASPP module consists of multiple dilated convolution filters with different dilation rates of 3,6 and 9. Our decoder network applies bilinear upsampling at each step, concatenated with the skip connection from the backbone, and followed by a 3×3 convolution, Batch Normalization [14], and ReLU activation [22] (except the last layer). The decoder network outputs coarse-grained alpha matte α_c, foreground residual F_c^R, error prediction map E_c and a 32-channel hidden features H_c. The hidden features H_c contain global contexts that will be useful for the refinement network.

4.2. Refinement Network

The goal of the refinement network is to reduce redundant computation and recover high-resolution matting details. While the base network operates on the whole image, the refinement network operates only on patches selected based on the error prediction map E_c. We perform a two-stage refinement, first at $\frac{1}{2}$ of the original resolution and then at the full resolution. During inference, we refine k patches, with k either set in advance or set based on a threshold that trades off between quality and computation time.

Given the coarse error prediction map E_c at $\frac{1}{c}$ of the original resolution, we first resample it to $\frac{1}{4}$ of the original resolution E_4, s.t. each pixel on the map corresponds to a 4×4 patch on the original resolution. We select the top k pixels with the highest predicted error from E_4 to denote the k 4×4 patch locations that will be refined by our refinement module. The total number of refined pixels at the original resolution is $16k$.

We perform a two-stage refinement process. First, we bilinearly resample the coarse outputs, i.e., alpha matte α_c, foreground residual F_c^R and hidden features H_c, as well as the input image I and background B to $\frac{1}{2}$ of the original resolution and concatenate them as features. Then we crop out 8×8 patches around the error locations selected from E_4, and pass each through two layers of 3×3 convolution with valid padding, Batch Normalization, and ReLU, which reduce the patch dimension to 4×4. These intermediate features are then upsampled to 8×8 again and concatenated with the 8×8 patches extracted from the original-resolution input I and background B at the corresponding location. We then apply an additional two layers of 3×3 convolution with valid padding, Batch Normalization and ReLU (except the last layer) to obtain 4×4 alpha matte and foreground residuals results. Finally, we upsample the coarse alpha matte α_c and foreground residual F_c^R to the original resolution and swap in the respective 4×4 patches that have been refined to obtain the final alpha matte α and foreground residual F^R. The entire architecture is illustrated in Figure 3. See appendix for the details of implementation.

4.3. Training

All matting datasets provide an alpha matte and a foreground layer, which we compose onto multiple high-resolution backgrounds. We employ multiple data augmentation techniques to avoid overfitting and help the model generalize to challenging real-world situations. We apply affine transformation, horizontal flipping, brightness, hue, and saturation adjustment, blurring, sharpening, and random noise as data augmentation to both the foreground and background layer independently. We also slightly translate the background to simulate misalignment and create artificial shadows to simulate how the presence of a subject can cast shadows in real-life environments (see appendix for more details). We randomly crop the images in every minibatch so that the height and width are each uniformly distributed between 1024 and 2048 to support inference at any resolution and aspect ratio.

To learn α w.r.t. ground-truth α^*, we use an L1 loss over the whole alpha matte and its (Sobel) gradient:

$$\mathcal{L}_\alpha = ||\alpha - \alpha^*||_1 + ||\nabla\alpha - \nabla\alpha^*||_1. \quad (1)$$

We obtain the foreground layer from predicted foreground residual F^R, using $F = \max(\min(F^R + I, 1), 0)$. We compute L1 loss only on the pixels where $\alpha^* > 0$:

$$\mathcal{L}_F = ||(\alpha^* > 0) * (F - F^*))||_1. \quad (2)$$

where that $(\alpha^* > 0)$ is a Boolean expression.

For refinement region selection, we define the ground truth error map as $E^* = |\alpha - \alpha^*|$. We then compute mean squared error between the predicted error map and the ground truth error map as the loss:

$$\mathcal{L}_E = ||E - E^*||_2. \quad (3)$$

This loss encourages the predicted error map to have larger values where the difference between the predicted alpha and the ground-truth alpha is large. The ground-truth error map changes over iterations during training as the predicted alpha improves. Over time, the error map converges and predicts high error in complex regions, e.g. hair, that would lead to poor composites if simply upsampled.

The base network $(\alpha_c, F_c^R, E_c, H_c) = G_{\text{base}}(I_c, B_c)$ operates at $\frac{1}{c}$ of the original image resolution, and is trained with the following loss function:

$$\mathcal{L}_{\text{base}} = \mathcal{L}_{\alpha_c} + \mathcal{L}_{F_c} + \mathcal{L}_{E_c}. \quad (4)$$

The refinement network $(\alpha, F^R) = G_{\text{refine}}(\alpha_c, F_c^R, E_c, H_c, I, B)$ is trained using:

$$\mathcal{L}_{\text{refine}} = \mathcal{L}_\alpha + \mathcal{L}_F. \quad (5)$$

We initialize our model's backbone and ASPP module with DeepLabV3 weights pre-trained for semantic segmentation on ImageNet and Pascal VOC datasets. We first train the base network till convergence and then add the refinement module and train it jointly. We use Adam optimizer and $c = 4$, $k = 5,000$ during all the training. For training only the base network, we use batch size 8 and learning rate [1e-4, 5e-4, 5e-4] for backbone, ASPP, and decoder. When training jointly, we use batch size 4 and learning rate [5e-5, 5e-5, 1e-4, 3e-4] for backbone, ASPP, decoder, and refinement module respectively.

We train our model on multiple datasets in the following order. First, we train only the base network G_{base} and then the entire model G_{base} and G_{refine} jointly on VideoMatte240K, which makes the model robust to a variety of subjects and poses. Next, we train our model jointly on PhotoMatte13K to improve the high-resolution details. Finally, we train our model jointly on Distinctions-646. The dataset has only 362 unique training samples, but it is of the highest quality and contains human-annotated foregrounds that are very helpful for improving the foreground quality produced by our model. We omit training on the AIM dataset as a possible 4th stage and only use it for testing because it causes a degradation in quality as shown in our ablation study in Section 6.

Dataset	Method	Alpha SAD	Alpha MSE	Alpha Grad	Alpha Conn	FG MSE
AIM	DIM†	37.94	80.67	32935	37861	-
	FBA†	**9.68**	**6.38**	**4265**	**7521**	**1.94**
	BGM	16.07	21.00	15371	14123	47.98
	BGM$_a$	19.28	29.31	19877	18083	42.84
	Ours	**12.86**	**12.01**	**8426**	**11116**	**5.31**
Distinctions	DIM†	43.70	86.22	49739	43914	-
	FBA†	**11.03**	**8.32**	**6894**	**9892**	**12.51**
	BGM	19.21	25.89	30443	18191	36.13
	BGM$_a$	16.02	20.18	24845	14900	43.00
	Ours	**9.19**	**7.08**	**6345**	**7216**	**6.10**
PhotoMatte85	DIM†	32.26	45.40	44658	30876	-
	FBA†	**7.37**	**4.79**	**7323**	**5206**	**7.03**
	BGM	17.32	21.21	27454	15397	14.25
	BGM$_a$	14.45	19.24	23314	13091	16.80
	Ours	**8.65**	**9.57**	**8736**	**6637**	**13.82**

Table 1: Quantitative evaluation on different datasets. † indicates methods that require a manual trimap.

5. Experimental Evaluation

We compare our approach to two trimap-based methods, Deep Image Matting (DIM) [33] and FBA Matting (FBA) [8], and one background-based method, Background Matting (BGM) [28]. The input resolution to DIM was fixed at 320×320 by the implementation, while we set the FBA input resolution to approximately HD due to memory limits. We additionally train the BGM model on our datasets and denote it as BGM$_a$ (BGM adapted).

Our evaluation uses $c = 4$, $k = 20,000$ for photos, $c = 4$, $k = 5,000$ for HD videos, and $c = 8$, $k = 20,000$ for 4K videos, where c is the downsampling factor for the base network and k is the number of patches that get refined.

5.1. Evaluation on composition datasets

We construct test benchmarks by separately compositing test samples from AIM, Distinctions, and PhotoMatte85 datasets onto 5 background images per sample. We apply minor background misalignment, color adjustment, and noise to simulate flawed background capture. We generate trimaps from ground-truth alpha using thresholding and morphological operations. We evaluate matte outputs using metrics from [26]: MSE (mean squared error) for alpha and foreground, SAD (sum of absolute difference), Grad (spatial-gradient metric), and Conn (connectivity) for alpha only. All MSE values are scaled by 10^3 and all metrics are only computed over the unknown region of trimaps generated as described above. Foreground MSE is additionally only measured where the grouth-truth alpha $\alpha^* > 0$.

Table 1 shows that our approach outperforms the existing background-based BGM method across all datasets.

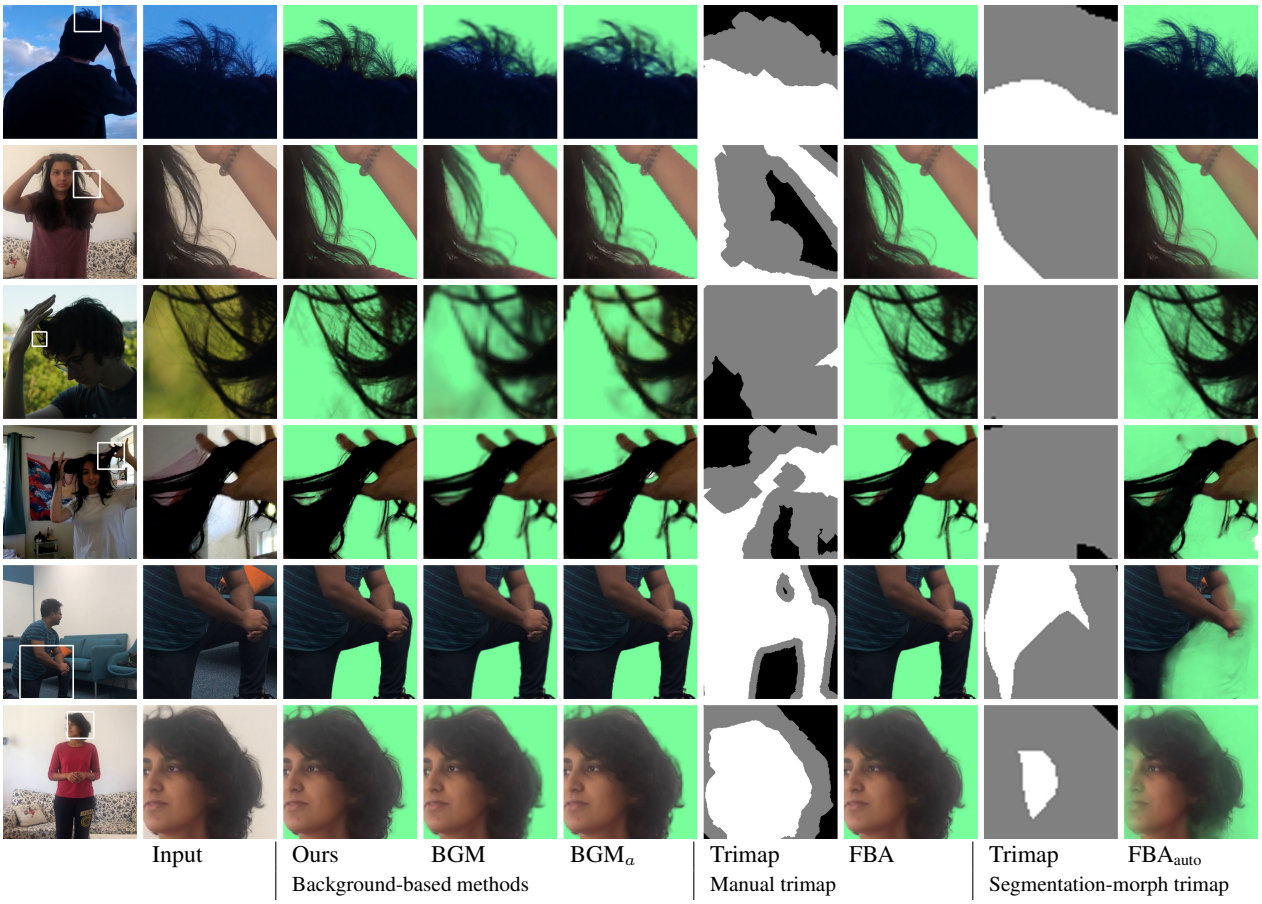

Figure 5: Qualitative comparison on real images. We produce superior results at high-resolution with minimal user input. While FBA is competitive, it fails in a fully automatic application where the segmentation-based trimap is faulty.

Our approach is slightly worse than the state-of-the-art trimap-based FBA method, which requires carefully annotated manual trimaps and is much slower than our approach, as shown later in the performance comparison.

5.2. Evaluation on captured data

Although quantitative evaluation on the above-mentioned datasets serves the purpose of quantifying the performance of different algorithms, it is important to evaluate how these methods perform on unconstrained real data. To evaluate on real data, we capture a number of photos and videos containing subjects in varying poses and surroundings. The videos are captured on a tripod with consumer smartphones (Samsung S10+ and iPhone X) and a professional camera (Sony α7s II), in both HD and 4K resolution. The photos are captured in 4000×6000 resolution. We also use some HD videos presented in the BGM paper that are made public to compare with our method.

For fair comparison in the real-time scenario, where manual trimaps cannot be crafted, we construct trimaps by morphing segmentation result from DeepLabV3+, as suggested in [28]. We show results on both trimaps, denoting FBA using this fully automatic trimap as FBA$_{\text{auto}}$.

Figure 5 shows our method produces much sharper and more detailed results around hair and edges compared to other methods. Since our refinement operates at the native resolution, the quality is far superior relative to BGM, which resizes the images and only processes them at 512×512 resolution. FBA, with manual trimap, produces excellent results around hair details, however cannot be evaluated at resolutions above around HD on standard GPUs. When FBA is applied on automatic trimaps generated with segmentation, it often shows large artifacts, mainly due to faulty segmentation.

We extract 34 frames from both the test videos shared by the BGM paper and our captured videos and photos to create a user study. 40 participants were presented with an interactive interface showing each input image as well as the mattes produced by BGM and our approach, in random order. They were encouraged to zoom in on details and asked to rate one of the mattes as ”much better”, ”slightly better”, or ”similar”. The results, shown in Table 2, demonstrate significant qualitative improvement over BGM. 59% of the time participants perceive our algorithm to be better, com-

pared to 23% for BGM. For sharp samples in 4K and larger, our method is preferred 75% of the time to BGM's 15%.

	Much worse	Worse	Similar	Better	Much better
All	6%	17%	18%	32%	27%
4K+	5%	10%	10%	34%	41%

Table 2: User study results: Ours vs BGM

5.3. Performance comparison

Table 3 and 4 show that our method is smaller and much faster than BGM. Our method contains only 55.7% of the parameters compared to BGM. Our method can achieve HD 60fps and 4K 30fps at batch size 1 on an Nvidia RTX 2080 TI GPU, considered to be real-time for many applications. It is a significant speed-up compared to BGM which can only handle 512×512 resolution at 7.8fps. The performance can be further improved by switching to MobileNetV2 backbone, which achieves 4K 45fps and HD 100fps. More performance results, such as adjusting the refinement selection parameter k and using a larger batch size, are included in the ablation studies and in the appendix.

Method	Backbone	Resolution	FPS	GMac
FBA		HD	3.3	54.3
FBA_{auto}		HD	2.9	137.6
BGM		512^2	7.8	473.8
Ours	ResNet-50*	HD	60.0	34.3
	ResNet-101	HD	42.5	44.0
	MobileNetV2	HD	100.6	9.9
Ours	ResNet-50*	4K	33.2	41.5
	ResNet-101	4K	29.8	51.2
	MobileNetV2	4K	45.4	17.0

Table 3: Speed measured on Nvidia RTX 2080 TI as PyTorch model pass-through without data transferring at FP32 precision and with batch size 1. GMac does not account for interpolation and cropping operations. For the ease of measurement, BGM and FBA_{auto} use adapted PyTorch DeepLabV3+ implementation with ResNet101 backbone as segmentation.

Method	Backbone	Parameters	Size
FBA		34,693,031	138.80 MB
FBA_{auto}		89,398,348	347.48 MB
BGM		72,231,209	275.53 MB
Ours	ResNet-50*	40,247,703	153.53 MB
	ResNet-101	59,239,831	225.98 MB
	MobileNetV2	5,006,839	19.10 MB

Table 4: Model size comparison. BGM and FBA_{auto} use DeepLabV3+ with Xception backbone for segmentation.

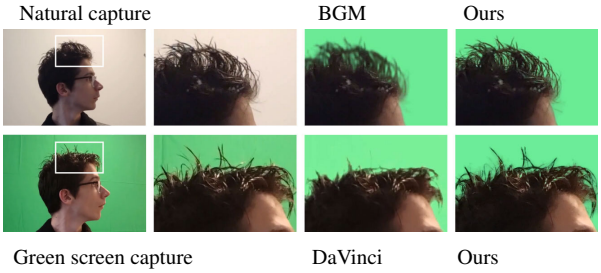

Figure 6: We produce better results than a chroma-keying software, when an amateur green-screen setup is used.

5.4. Practical use

Zoom implementation We build a Zoom plugin that intercepts the webcam input, collects one no-person (background) shot, then performs real-time video matting and compositing, streaming the result back into the Zoom call. We test with a 720p webcam in Linux. The upgrade elicits praise in real meetings, demonstrating its practicality in a real-world setting.

Comparison to green-screen Chroma keying with a green screen is the most popular method for creating high-quality mattes. However, it requires even lighting across the screen and background-subject separation to avoid cast shadows. In Figure 6, we compare our method against chroma-keying under the same lighting with an amateur green-screen setup. We find that in the unevenly lit setting, our method outperforms approaches designed for the green screen.

6. Ablation Studies

Role of our datasets We train on multiple datasets, each of which brings unique characteristics that help our network produce high-quality results at high-resolution. Table 5 shows the metrics of our method by adding or removing a dataset from our training pipeline. We find adding the AIM dataset as a possible 4th stage worsens the metrics even on the AIM test set itself. We believe it is because samples in the AIM dataset are lower in resolution and quality compared to Distinctions and the small number of samples may have caused overfitting. Removal of VideoMatte240K, PhotoMatte13K, and Distinctions datasets from the training pipeline all result in worse metrics, proving that those datasets are helpful in improving the model's quality.

Role of the base network We experiment with replacing ResNet-50 with ResNet-101 and MobileNetV2 as our encoder backbone in the base network. The metrics in Table 6 show that ResNet-101 has slight improvements over ResNet-50 on some metrics while doing worse on others. This indicates that ResNet-50 is often sufficient for obtaining the best quality. MobileNetV2 on the other hand is worse than ResNet-50 on all metrics, but it is significantly faster and smaller than ResNet-50 as shown in Tables 3 and 4, and still obtains better metrics than BGM.

Method	Alpha SAD	MSE	Grad	Conn	FG MSE
Ours*	**12.86**	**12.01**	**8426**	**11116**	**5.31**
+ AIM	14.19	14.70	9629	12648	6.34
- PhotoMatte13K	14.05	14.10	10102	12749	6.53
- VideoMatte240K	15.17	17.31	11907	13827	7.04
- Distinctions	15.95	19.51	11911	14909	14.36
BGM	16.07	21.00	15371	14123	42.84

Table 5: Effect of removing or appending a dataset in the training pipeline, evaluated on the AIM test set.

Base Backbone	Refine Kernel	Alpha SAD	MSE	Grad	Conn	FG MSE
BGM_a		16.02	20.18	24845	14900	43.00
MobileNetV2	3×3	10.53	9.62	7904	8808	8.19
ResNet-50*	3×3	**9.19**	7.08	6345	7216	**6.10**
ResNet-101	3×3	9.30	**6.82**	**6191**	**7128**	7.68
ResNet-50	1×1	9.36	8.06	7319	7395	6.92

Table 6: Comparison of backbones and refinement kernels on the Distinctions test set

Role of the refinement network Our refinement network improves detail sharpness over the coarse results in Figure 7, and is effective even in 4K resolution. Figure 8 shows the effects of increasing and decreasing the refinement area. Most improvement can be achieved by refining over only 5% to 10% of the image resolution. Table 7 shows that refining only the selected patches provides significant speedup compared to refining the full image.

Figure 7: Effect of refinement, from coarse to HD and 4K.

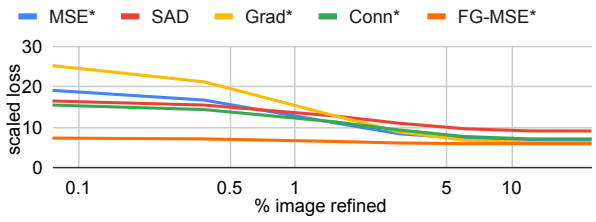

Figure 8: Metrics on the Distinctions test set over percentage of image area refined. Grad and Conn are scaled by 10^{-3}.

k	2,500	5,000*	7,500	Full
FPS	62.9	60.0	55.7	42.8

Table 7: Performance with different k values. Measured on our method with ResNet-50 backbone at HD.

Figure 9: Failure cases. Our method fails when the subject casts a substantial shadow on, or matches color with, the background (top) and when the background is highly textured (bottom).

Patch-based refinement vs. Point-based refinement Our refinement module uses a stack of 3×3 convolution kernels, creating a 13×13 receptive field for every output pixel. An alternative is to refine only on points using 1×1 convolution kernels, which would result in a 2×2 receptive field with our method. Table 6 shows that the 3 × 3 kernel can achieve better metrics than point-based kernels, due to a larger receptive field.

Limitations Our method can be used on handheld input by applying homography alignment to the background on every frame, but it is limited to small motion. Other common limitations are indicated in Figure 9. We recommend using our method with a simple-textured background, fixed exposure/focus/WB setting, and a tripod for the best result.

7. Conclusion

We have proposed a real-time, high-resolution background replacement technique that operates at 4K 30fps and HD 60fps. Our method only requires an input image and an pre-captured background image, which is easy to obtain in many applications. Our proposed architecture efficiently refines only the error-prone regions at high-resolution, which reduces redundant computation and makes real-time high-resolution matting possible. We introduce two new large-scale matting datasets that help our method generalize to real-life scenarios. Our experiment shows our method sets new state-of-the-art performance on background matting. We demonstrate the practicality of our method by streaming our results to Zoom and achieve a much more realistic virtual conference call.

Ethics Our primary goal is to enable creative applications and give users more privacy options through background replacement in video calls. However, we recognize that image editing can also be used for negative purposes, which can be mitigated through watermarking and other security techniques in commercial applications of this work.

References

[1] Yağiz Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. Semantic soft segmentation. *ACM Transactions on Graphics (TOG)*, 37(4):72, 2018. 2

[2] Yagiz Aksoy, Tunc Ozan Aydin, and Marc Pollefeys. Designing effective inter-pixel information flow for natural image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 29–37, 2017. 1, 2

[3] Shaofan Cai, Xiaoshuai Zhang, Haoqiang Fan, Haibin Huang, Jiangyu Liu, Jiaming Liu, Jiaying Liu, Jue Wang, and Jian Sun. Disentangled image matting. *International Conference on Computer Vision (ICCV)*, 2019. 2

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 3

[5] 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, 3

[6] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2175–2188, 2013. 1, 2

[7] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *CVPR (2)*, pages 264–271, 2001. 1, 2

[8] Marco Forte and François Pitié. F,b, alpha matting. *arXiv preprint arXiv:2003.07711*, 2020. 5

[9] Eduardo SL Gastal and Manuel M Oliveira. Shared sampling for real-time alpha matting. In *Computer Graphics Forum*, volume 29, pages 575–584. Wiley Online Library, 2010. 1, 2

[10] Minglun Gong and Yee-Hong Yang. Near-real-time image matting with known background. In *2009 Canadian Conference on Computer and Robot Vision*, pages 81–87. IEEE, 2009. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[13] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. *International Conference on Computer Vision (ICCV)*, 2019. 2

[14] S. Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015. 4

[15] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9808, 2020. 2

[16] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):228–242, 2007. 1, 2

[17] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1699–1712, 2008. 1, 2

[18] Yaoyi Li, Qingyao Xu, and Hongtao Lu. Hierarchical opacity propagation for image matting. *arXiv preprint arXiv:2004.03249*, 2020. 2

[19] Xiaodan Liang, Ke Gong, Xiaohui Shen, and Liang Lin. Look into person: Joint body parsing & pose estimation network and a new benchmark. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):871–885, 2018. 2

[20] Jinlin Liu, Yuan Yao, Wendi Hou, Miaomiao Cui, Xuansong Xie, Changshui Zhang, and Xian-sheng Hua. Boosting semantic human matting with coarse annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8563–8572, 2020. 2

[21] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Indices matter: Learning to index for deep image matting. *International Conference on Computer Vision (ICCV)*, 2019. 2

[22] V. Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 4

[23] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, B. Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019. 11

[24] 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. 2

[25] Yu Qiao, Yuhao Liu, Xin Yang, Dongsheng Zhou, Mingliang Xu, Qiang Zhang, and Xiaopeng Wei. Attention-guided hierarchical structure aggregation for image matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13676–13685, 2020. 1, 2

[26] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1826–1833. IEEE, 2009. 5, 12

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3

[28] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2291–2300, 2020. 1, 2, 5, 6

[29] Xiaoyong Shen, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia. Deep automatic portrait matting. In *European Conference on Computer Vision*, pages 92–107. Springer, 2016. 2

[30] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 315–321. ACM, 2004. 1, 2

[31] Jingwei Tang, Yagiz Aksoy, Cengiz Oztireli, Markus Gross, and Tunc Ozan Aydin. Learning-based sampling for natural image matting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[32] Jue Wang, Michael F Cohen, et al. Image and video matting: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(2):97–175, 2008. 2

[33] 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. 1, 2, 5

[34] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul Rosin, Zixi Cai, Xi Han, Dingcheng Yang, Haozhi Huang, and Shi-Min Hu. Pose2seg: Detection free human instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 889–898, 2019. 2

[35] Yunke Zhang, Lixue Gong, Lubin Fan, Peiran Ren, Qixing Huang, Hujun Bao, and Weiwei Xu. A late fusion cnn for digital matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7469–7478, 2019. 2

[36] Bingke Zhu, Yingying Chen, Jinqiao Wang, Si Liu, Bo Zhang, and Ming Tang. Fast deep matting for portrait animation on mobile phone. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 297–305. ACM, 2017. 2

A. Overview

We provide additional details in this appendix. In Sec. B, we describe the details of our network architecture and implementation. In Sec. C, we clarify our use of keywords for crawling background images. In Sec. D, we explain how we train our model and show details of our data augmentations. In Sec. E, we show additional metrics about our method's performance. In Sec. F, we show all the qualitative results used in our user study along with the average score per sample.

B. Network

B.1. Architecture

Backbone Both ResNet and MobileNetV2 are adopted from the original implementation with minor modifications. We change the first convolution layer to accept 6 channels for both the input and the background images. We follow DeepLabV3's approach and change the last downsampling block with dilated convolutions to maintain an output stride of 16. We do not use the multi-grid dilation technique proposed in DeepLabV3 for simplicity.

ASPP We follow the original implementation of ASPP module proposed in DeepLabV3. Our experiment suggests that setting dilation rates to (3, 6, 9) produces the better results.

Decoder

CBR128 - CBR64 - CBR48 - C37

"CBRk" denotes k 3×3 convolution filters with same padding without bias followed by Batch Normalization and ReLU. "Ck" denotes k 3×3 convolution filters with same padding and bias. Before every convolution, decoder uses bilinear upsampling with a scale factor of 2 and concatenates with the corresponding skip connection from the backbone. The 37-channel output consists of 1 channel of alpha α_c, 3 channels of foreground residual F_c^R, 1 channel of error map E_c, and 32 channels of hidden features H_c. We clamp α_c and E_c to 0 and 1. We apply ReLU on H_c.

Refiner

First stage:	C*BR24 - C*BR16
Second stage:	C*BR12 - C*4

"C*BRk" and "C*k" follow the same definition above except that the convolution does not use padding.

Refiner first resamples coarse outputs α_c, F_c^R, H_c, and input images I, B to $\frac{1}{2}$ resolution and concatenates them as $[n \times 42 \times \frac{h}{2} \times \frac{w}{2}]$ features. Based on the error prediction E_c, we crop out top k most error-prone patches $[nk \times 42 \times 8 \times 8]$. After applying the first stage, the patch dimension becomes $[nk \times 16 \times 4 \times 4]$. We upsample the patches with nearest upsampling and concatenate them with patches at the corresponding location from I and B to form $[nk \times 22 \times 8 \times 8]$ features. After the second stage, the patch dimension becomes $[nk \times 4 \times 4 \times 4]$. The 4 channels are alpha and foreground residual. Finally, we bilinearly upsample the coarse α_c and F_c^R to full resolution and replace the refined patches to their corresponding location to form the final output α and F^R.

B.2. Implementation

We implement our network in PyTorch [23]. The patch extraction and replacement can be achieved via the native vectorized operations for maximum performance. We find that PyTorch's nearest upsampling operation is much faster on small-resolution patches than bilinear upsampling, so we use it when upsampling the patches.

C. Dataset

VideoMatte240K The dataset contains 484 video clips, which consists a total of 240,709 frames. The average frames per clip is 497.3 and the median is 458.5. The longest clip has 1500 frames while the shortest clip has 124 frames. Figure 10 shows more examples from VideoMatte240K dataset.

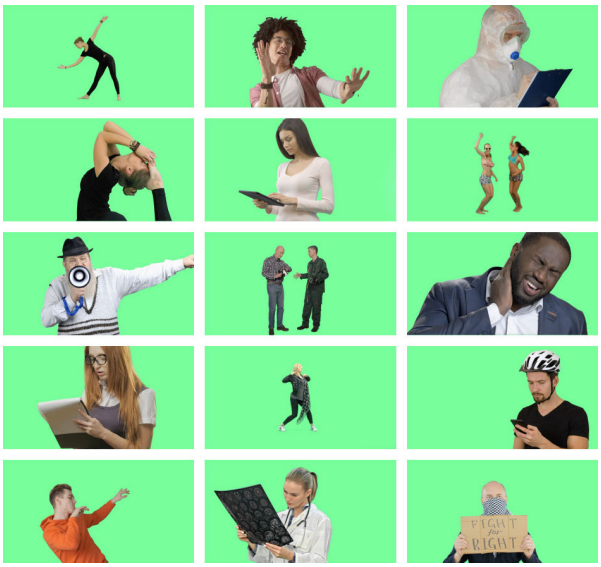

Figure 10: More examples from VideoMatte240K dataset.

Background The keywords we use for crawling background images are:

airport interior, attic, bar interior, bathroom, beach, city, church interior, classroom interior, empty city, forest, garage interior, gym interior, house outdoor, interior, kitchen, lab interior, landscape, lecture hall, mall interior, night club interior, office, rainy woods, rooftop, stadium interior, theater interior, train station, warehouse interior, workplace interior

D. Training

Table 8 records the training order, epochs, and hours of our final model on different datasets. We use 1×RTX 2080 TI when training only the base network and 2×RTX 2080 TI when training the network jointly.

Dataset	Network	Epochs	Hours
VideoMatte240K	G_{base}	8	24
VideoMatte240K	$G_{\text{base}} + G_{\text{refine}}$	1	12
PhotoMatte13K	$G_{\text{base}} + G_{\text{refine}}$	25	8
Distinctions	$G_{\text{base}} + G_{\text{refine}}$	30	8

Table 8: Training epochs and hours on different datasets. Time measured on model with ResNet-50 backbone.

Additionally, we use mixed precision training for faster computation and less memory consumption. When using multiple GPUs, we apply data parallelism to split the mini-batch across multiple GPUs and switch to use PyTorch's Synchronized Batch Normalization to track batch statistics across GPUs.

D.1. Training augmentation

For every alpha and foreground training sample, we rotate to composite with backgrounds in a "zip" fashion to form a single epoch. For example, if there are 60 training samples and 100 background images, a single epoch is 100 images, where the 60 samples first pair with the first 60 background images, then the first 40 samples pair with the rest of the 40 background images again. The rotation stops when one set of images runs out. Because the datasets we use are very different in sizes, this strategy is used to generalize the concept of an epoch.

We apply random rotation (±5deg), scale (0.3∼1), translation (±10%), shearing (±5deg), brightness (0.85∼1.15), contrast (0.85∼1.15), saturation (0.85∼1.15), hue (±0.05), gaussian noise (σ^2 ≤0.03), box blurring, and sharpening independently to foreground and background on every sample. We then composite the input image using $I = \alpha F + (1 - \alpha)B$.

We additionally apply random rotation (±1deg), translation (±1%), brightness (0.82∼1.18), contrast (0.82∼1.18), saturation (0.82∼1.18), and hue (±0.1) only on the background 30% of the time. This small misalignment between input I and background B increases model's robustness on real-life captures.

We also find creating artificial shadows increases model's robustness because subjects in real-life often cast shadows on the environment. Shadows are created on I by darkening some areas of the image behind the subject following the subject's contour 30% of the time. Examples of composited images are shown in Figure 11. The bottom row shows examples of shadow augmentation.

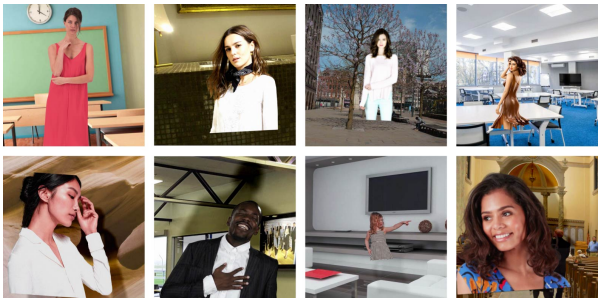

Figure 11: Training samples with augmentations. Bottom row are samples with shadow augmentation. Actual samples have different resolutions and aspect ratios.

D.2. Testing augmentation

For AIM and Distinctions, which have 11 human test samples each, we pair every sample with 5 random backgrounds from the background test set. For PhotoMatte85, which has 85 test samples, we pair every sample with only 1 background. We use the method and metrics described in [26] to evaluate the resulting sets of 55, 55, and 85 images.

We apply a random subpixel translation (±0.3 pixels), random gamma (0.85∼1.15), and gaussian noise ($\mu = \pm 0.02$, $0.08 \leq \sigma^2 \leq 0.15$) to background B only, to simulate misalignment.

The trimaps used as input for trimap-based methods and for defining the error metric regions are obtained by thresholding the grouth-truth alpha between 0.06 and 0.96, then applying 10 iterations of dilation followed by 10 iterations of erosion using a 3×3 circular kernel.

E. Performance

Table 9 shows the performance of our method on two Nvidia RTX 2000 series GPUs: the flagship RTX 2080 TI and the entry-level RTX 2060 Super. The entry-level GPU yields lower FPS but is still within an acceptable range for many real-time applications. Additionally, Table 10 shows that switching to a larger batch size and a lower precision can increase the FPS significantly.

F. Additional Results

In Figures 13, 14, 15, we show all 34 examples in the user study, along with their average rating and results by different methods. Figure 12 shows the web UI for our user-study.

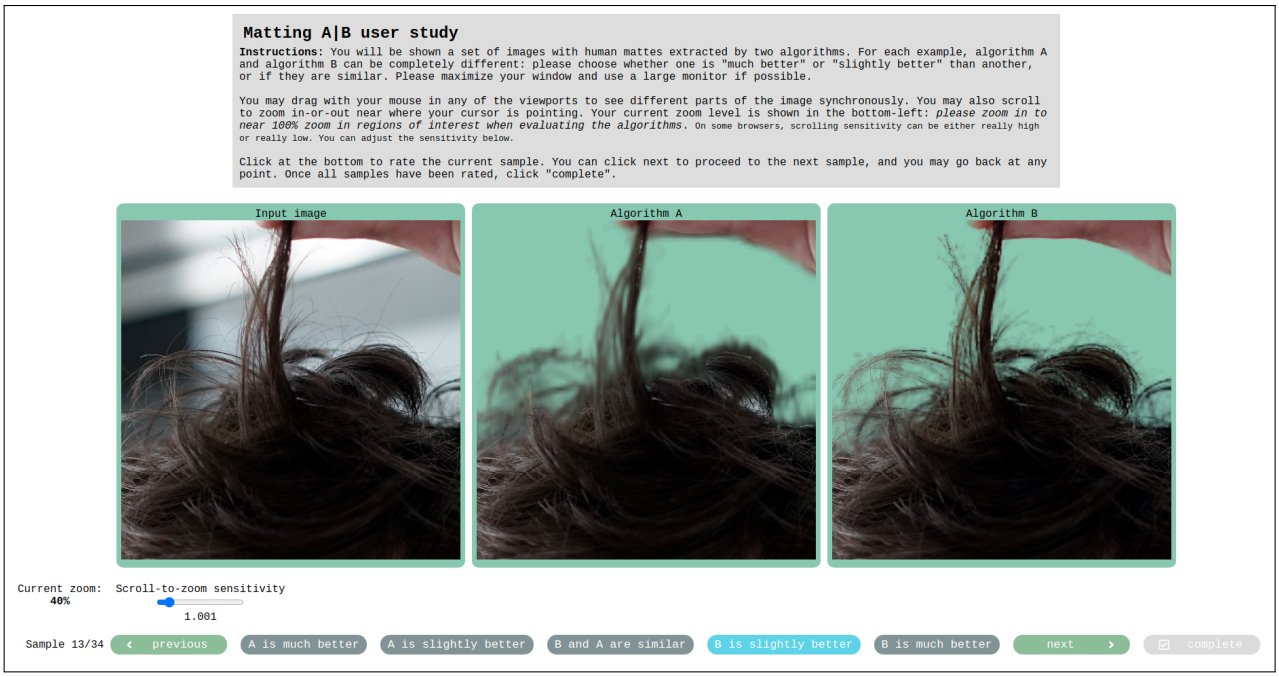

Figure 12: The web UI for our user study. Users are shown the original image and two result images from Ours and BGM methods. Users are given the instruction to rate whether one algorithm is ”much better”, ”slightly better”, or both as ”similar”.

GPU	Backbone	Reso	FPS
RTX 2080 TI	ResNet-50	HD	60.0
		4K	33.2
	MobileNetV2	HD	100.6
		4K	45.4
RTX 2060 Super	ResNet-50	HD	42.8
		4K	23.3
	MobileNetV2	HD	75.6
		4K	31.3

Table 9: Performance on different GPUs. Measured with batch size 1 and FP32 precision.

Backbone	Reso	Batch	Precision	FPS
MobileNetV2	HD	1	FP32	100.6
		8	FP32	138.4
		8	FP16	200.0
	4K	8	FP16	64.2

Table 10: Performance using different batch sizes and precisions. Measured on RTX 2080 TI.

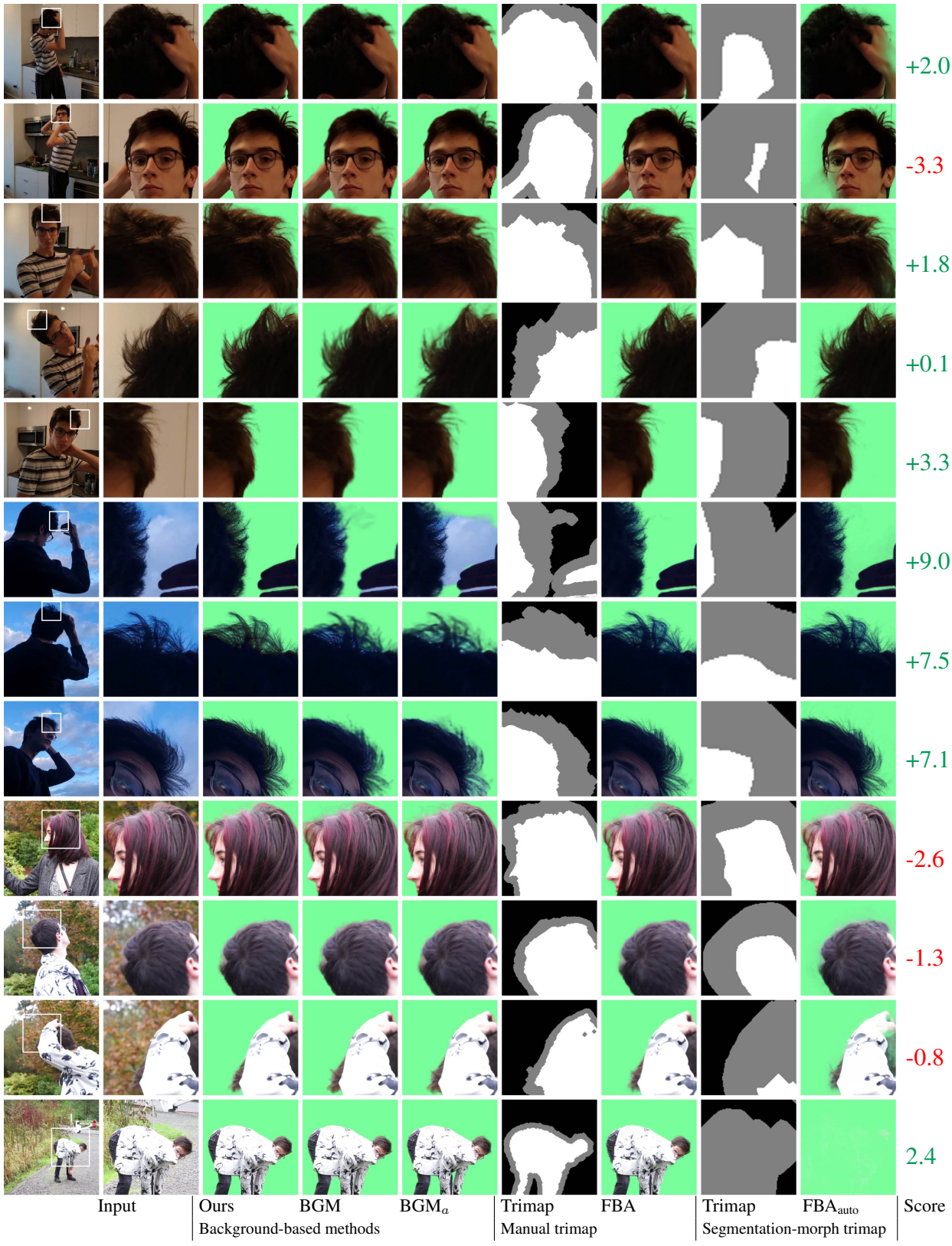

Figure 13: Additional qualitative comparison (1/3). Average user ratings between Ours and BGM are included. A score of -10 denotes BGM is ”much better”, -5 denotes BGM is ”slightly better”, 0 denotes ”similar”, +5 denotes Ours is ”slightly better”, +10 denotes Ours is ”much better”. Our method receives an average 3.1 score.

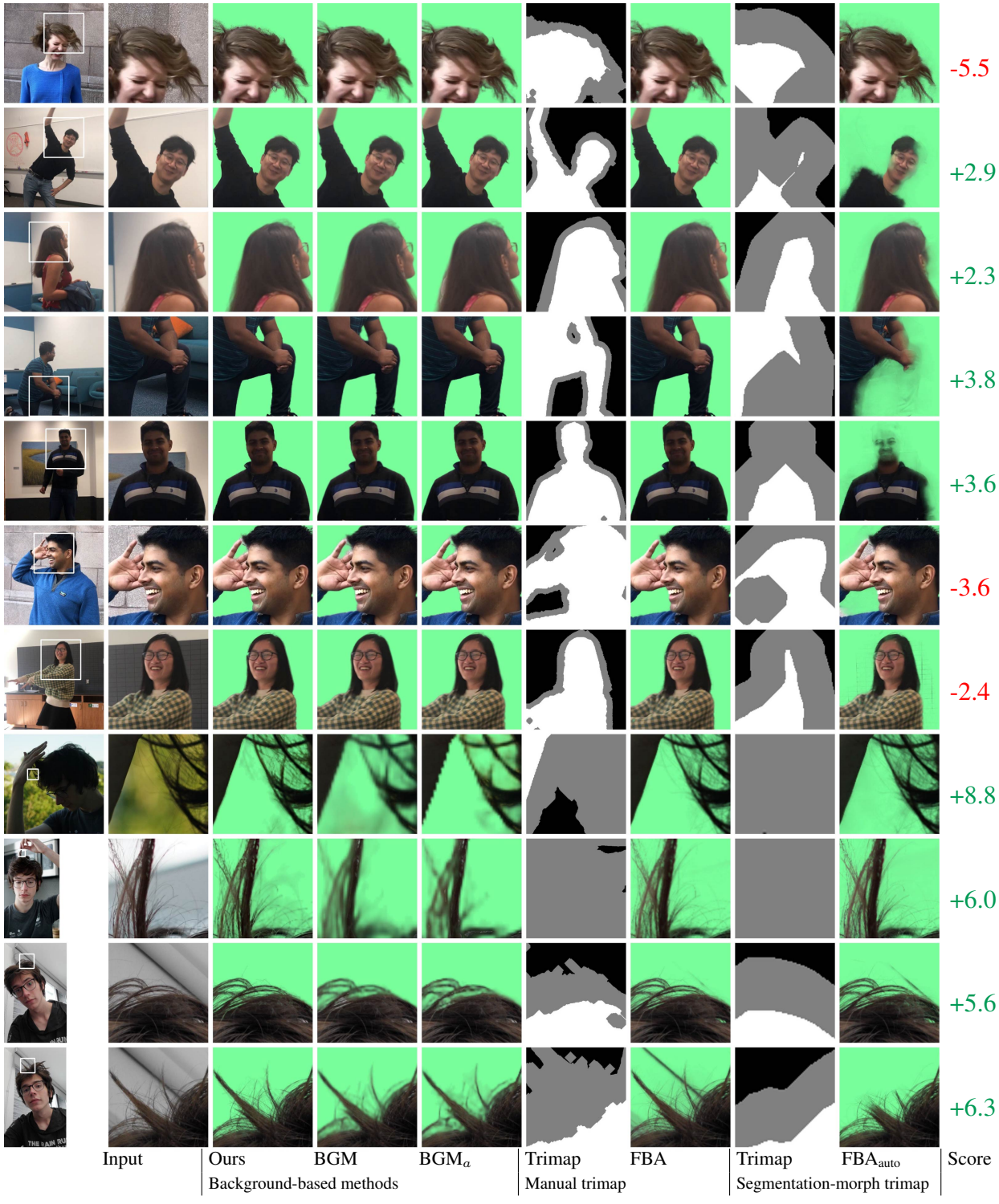

Figure 14: Additional qualitative comparisons (2/3)

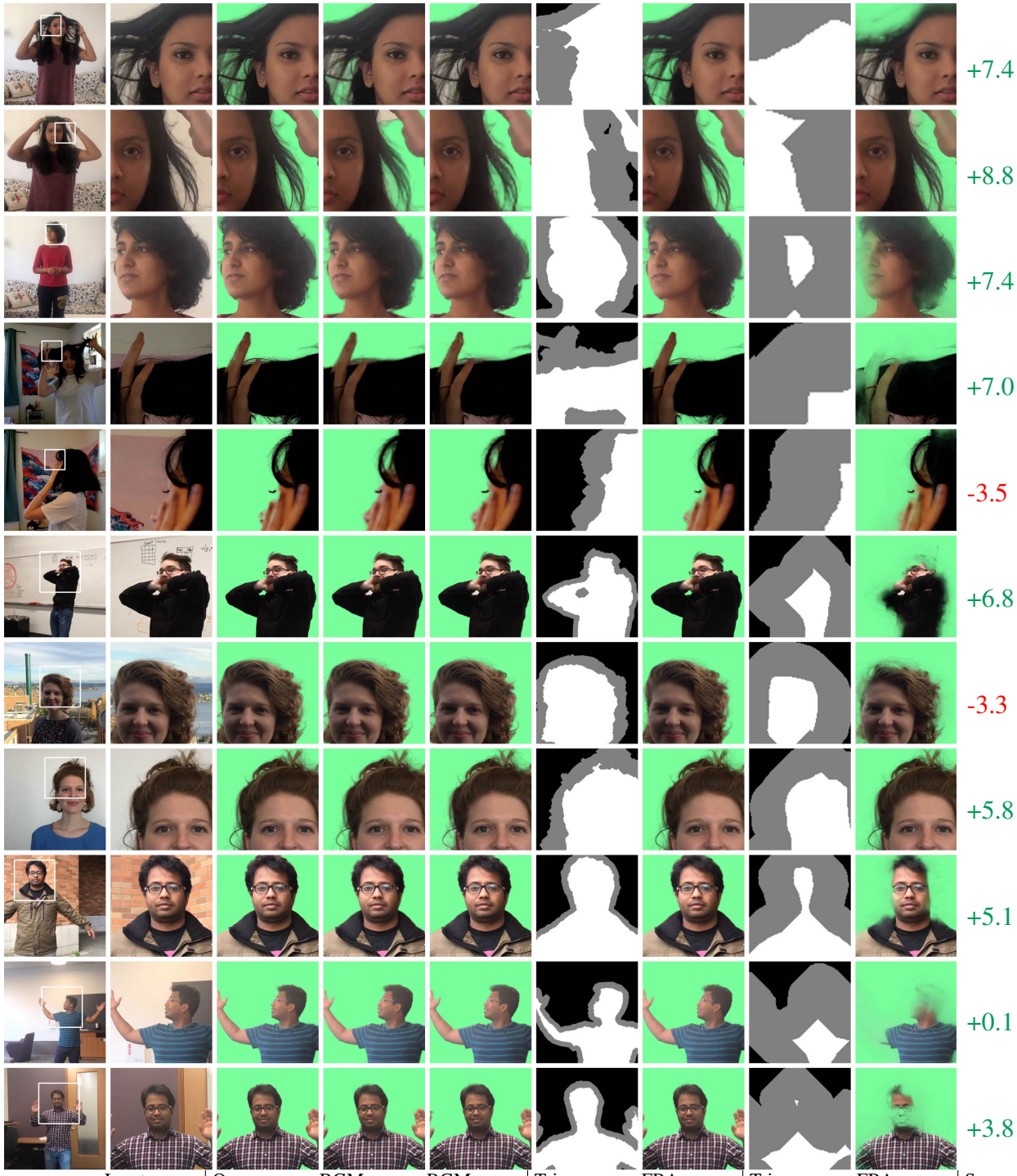

Figure 15: Additional qualitative comparisons (3/3)