Robust High-Resolution Video Matting with Temporal Guidance

Shanchuan Lin[1*] Linjie Yang[2] Imran Saleemi[2] Soumyadip Sengupta[1]
[1]University of Washington [2]ByteDance Inc.
{linsh,soumya91}@cs.washington.edu {linjie.yang,imran.saleemi}@bytedance.com

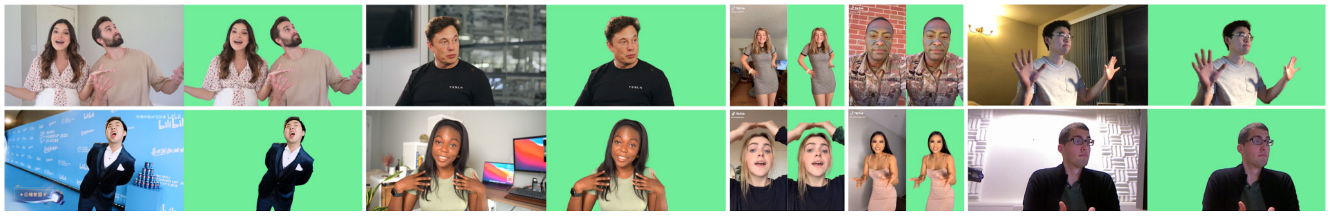


Figure 1: Our method produces robust and coherent results on all videos without requirements for trimaps or pre-captured backgrounds.

Abstract

We introduce a robust, real-time, high-resolution human video matting method that achieves new state-of-the-art performance. Our method is much lighter than previous approaches and can process 4K at 76 FPS and HD at 104 FPS on an Nvidia GTX 1080Ti GPU. Unlike most existing methods that perform video matting frame-by-frame as independent images, our method uses a recurrent architecture to exploit temporal information in videos and achieves significant improvements in temporal coherence and matting quality. Furthermore, we propose a novel training strategy that enforces our network on both matting and segmentation objectives. This significantly improves our model's robustness. Our method does not require any auxiliary inputs such as a trimap or a pre-captured background image, so it can be widely applied to existing human matting applications. Our code is available at `https://peterl1n.github.io/RobustVideoMatting/`

1. Introduction

Matting is the process of predicting the alpha matte and foreground color from an input frame. Formally, a frame I can be viewed as the linear combination of a foreground F and a background B through an α coefficient:

$$I = \alpha F + (1 - \alpha)B \tag{1}$$

By extracting α and F, we can composite the foreground object to a new background, achieving the background replacement effect.

Background replacement has many practical applications. Many rising use cases, *e.g.* video conferencing and entertainment video creation, need real-time background replacement on human subjects without green-screen props. Neural models are used for this challenging problem but the current solutions are not always robust and often generate artifacts. Our research focuses on improving the matting quality and robustness for such applications.

Most existing methods [18, 22, 34], despite being designed for video applications, process individual frames as independent images. Those approaches neglect the most widely available feature in videos: temporal information. Temporal information can improve video matting performance for many reasons. First, it allows the prediction of more coherent results, as the model can see multiple frames and its own predictions. This significantly reduces flicker and improves perceptual quality. Second, temporal information can improve matting robustness. In the cases where an individual frame might be ambiguous, *e.g.* the foreground color becomes similar to a passing object in the background, the model can better guess the boundary by referring to the previous frames. Third, temporal information allows the model to learn more about the background over time. When the camera moves, the background behind the subjects is revealed due to the perspective change. Even if the camera is fixed, the occluded background still often reveals due to the subject's movements. Having a better understanding of the background simplifies the matting task. Therefore, we propose a recurrent architecture to exploit the temporal information. Our method significantly improves the matting quality and temporal coherence. It can be applied to all videos without any requirements for auxiliary inputs, such as a manually annotated trimap or a pre-captured background image.

Furthermore, we propose a new training strategy to enforce our model on both matting and semantic segmentation objectives simultaneously. Most existing methods

*Work performed during an internship at ByteDance.

[18, 22, 34] are trained on synthetic matting datasets. The samples often look fake and prevent the network to generalize to real images. Previous works [18, 22] have attempted to initialize the model with weights trained on segmentation tasks, but the model still overfits to the synthetic distribution during the matting training. Others have attempted adversarial training [34] or semi-supervised learning [18] on unlabeled real images as an additional adaptation step. We argue that human matting tasks are closely related to human segmentation tasks. Simultaneously training with a segmentation objective can effectively regulate our model without additional adaptation steps.

Our method outperforms the previous state-of-the-art method while being much lighter and faster. Our model uses only 58% parameters and can process real-time high-resolution videos at 4K 76 FPS and HD 104 FPS on an Nvidia GTX 1080Ti GPU.

2. Related Works

Trimap-based matting. Classical (non-learning) algorithms [1, 5, 7, 10, 20, 21, 38] require a manual trimap annotation to solve for the unknown regions of the trimap. Such methods are reviewed in the survey by Wang and Cohen [43]. Xu *et al*. [45] first used a deep network for trimap-based matting and many recent research continued this approach. FBA [9] is one of the latest. Trimap-based methods are often object agnostic (not limited to human). They are suitable for interactive photo-editing applications where the user can select target objects and provide manual guidance. To extend it to video, Sun *et al*. proposed DVM [39], which only requires a trimap on the first frame and can propagate it to the rest of the video.

Background-based matting. Soumyadip *et al*. proposed background matting (BGM) [34], which requires an additional pre-captured background image as input. This information acts as an implicit way for foreground selection and increases matting accuracy. Lin and Ryabtsev *et al*. further proposed BGMv2 [22] with improved performance and a focus on real-time high-resolution. However, background matting cannot handle dynamic backgrounds and large camera movements.

Segmentation. Semantic segmentation is to predict a class label for every pixel, often without auxiliary inputs. Its binary segmentation mask can be used to locate the human subjects, but using it directly for background replacement will result in strong artifacts. Nonetheless, segmentation tasks are similar to matting tasks in an auxiliary-free setting, and the research in segmentation inspires our network design. DeepLabV3 [3] proposed ASPP (Atrous Spatial Pyramid Pooling) module and used dilated convolution in its encoder to improve performance. This design has been adopted by many following works, including MobileNetV3 [15], which simplified ASPP to LR-ASPP.

Auxiliary-free matting. Fully automatic matting without any auxiliary inputs has also been studied. Methods like [29, 46] work on any foreground objects but not as robust, while others like [18, 35, 47] are trained specifically for human portraits. MODNet [18] is the latest portrait matting method. In contrast, our method is trained to work well on the full human body.

Video matting. Very few neural matting methods are designed for videos natively. MODNet [18] proposed a post-processing trick that compares the prediction of neighboring frames to suppress flicker, but it cannot handle fast-moving body-parts and the model itself still operates on frames as independent images. BGM [34] explored taking a few neighboring frames as additional input channels, but this only provides short-term temporal cues and its effects were not the focus of the study. DVM [45] is video-native but focused on utilizing temporal information to propagate trimap annotations. On contrary, our method focuses on using temporal information to improve matting quality in an auxiliary-free setting.

Recurrent architecture. Recurrent neural network has been widely used for sequence tasks. Two of the most popular architectures are LSTM (Long Short-Term Memory) [13] and GRU (Gated Recurrent Unit) [6], which have also being adopted to vision tasks as ConvLSTM [36] and ConvGRU [2]. Previous works have explored using recurrent architectures for various video vision tasks and showed improved performance compared to the image-based counterparts [42, 28, 41]. Our work adopts recurrent architectures to the matting task.

High-resolution matting. Patch-based refinement has been explored by PointRend [19] for segmentation and BGMv2 [22] for matting. It only performs convolution on selective patches. Another approach is using Guided Filter [11], a post-processing filter that jointly upsamples the low-resolution prediction given the high-resolution frame as guidance. Deep Guided Filter (DGF) [44] was proposed as a learnable module that can be trained with the network end-to-end without manual hyperparameters. Despite filter-based upsampling being less powerful, we choose it because it is faster and well supported by all inference frameworks.

3. Model Architecture

Our architecture consists of an encoder that extracts individual frame's features, a recurrent decoder that aggregates temporal information, and a Deep Guided Filter module for high-resolution upsampling. Figure 2 shows our model architecture.

3.1. Feature-Extraction Encoder

Our encoder module follows the design of state-of-the-art semantic segmentation networks [3, 4, 15] because the ability to accurately locate human subjects is fundamental

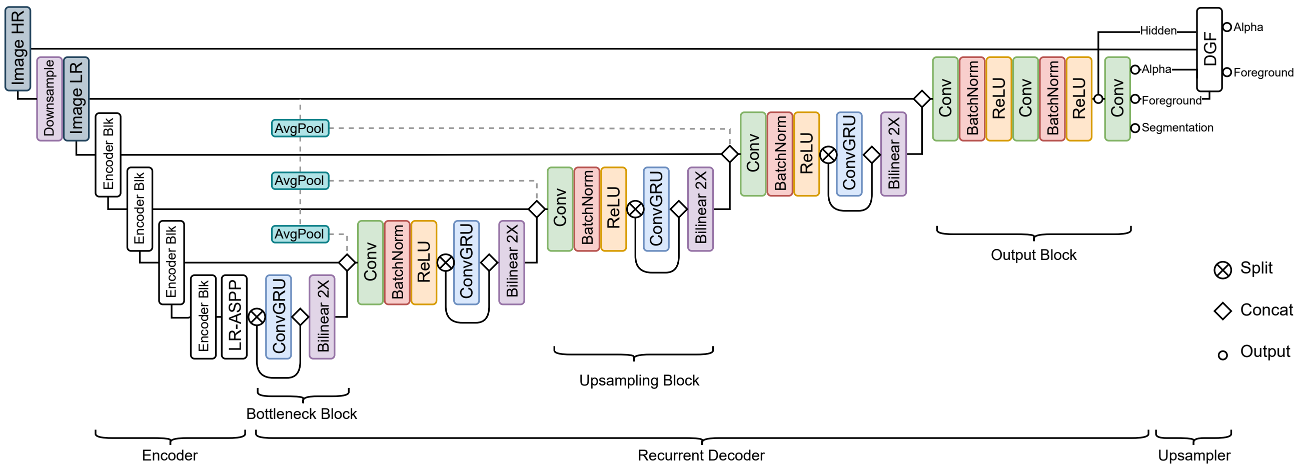


Figure 2: Our network consists of a feature-extraction encoder, a recurrent decoder, and Deep Guided Filter (DGF) module. To process high-resolution videos, the input is first downsampled for the encoder-decoder network, then DGF is used to upsample the result.

to the matting task. We adopt MobileNetV3-Large [15] as our efficient backbone followed by the LR-ASPP module as proposed by MobileNetV3 for semantic segmentation tasks. Noticeably, the last block of MobileNetV3 uses dilated convolutions without downsampling stride. The encoder module operates on individual frames and extracts features at $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, and $\frac{1}{16}$ scales for the recurrent decoder.

3.2. Recurrent Decoder

We decide to use a recurrent architecture instead of attention or simply feedforwarding multiple frames as additional input channels for several reasons. Recurrent mechanisms can learn what information to keep and forget by itself on a continuous stream of video, while the other two methods must rely on a fixed rule to remove old and insert new information to the limited memory pool on every set interval. The ability to adaptively keep both long-term and short-term temporal information makes recurrent mechanisms more suitable for our task.

Our decoder adopts ConvGRU at multiple scales to aggregate temporal information. We choose ConvGRU because it is more parameter efficient than ConvLSTM by having fewer gates. Formally, ConvGRU is defined as:

$$\begin{aligned} z_t &= \sigma(w_{zx} * x_t + w_{zh} * h_{t-1} + b_z) \\ r_t &= \sigma(w_{rx} * x_t + w_{rh} * h_{t-1} + b_r) \\ o_t &= tanh(w_{ox} * x_t + w_{oh} * (r_t \odot h_{t-1}) + b_o) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot o_t \end{aligned} \tag{2}$$

where operators $*$ and $\odot$ represent convolution and element-wise product respectively; $tanh$ and σ represent hyperbolic tangent and sigmoid function respectively. w and b are the convolution kernel and the bias term. The hidden state h_t is used as both the output and the recurrent state to the next time step as h_{t-1}. The initial recurrent state h_0 is an all zero tensor.

As shown in Figure 2, our decoder consists of a bottleneck block, upsampling blocks, and an output block.

Bottleneck block operates at the $\frac{1}{16}$ feature scale after the LR-ASPP module. A ConvGRU layer is operated on only half of the channels by split and concatenation. This significantly reduces parameters and computation since ConvGRU is computationally expansive.

Upsampling block is repeated at $\frac{1}{8}$, $\frac{1}{4}$, and $\frac{1}{2}$ scale. First, it concatenates the bilinearly upsampled output from the previous block, the feature map of the corresponding scale from the encoder, and the input image downsampled by repeated 2×2 average pooling. Then, a convolution followed by Batch Normalization [16] and ReLU [26] activation is applied to perform feature merging and channel reduction. Finally, a ConvGRU is applied to half of the channels by split and concatenation.

Output block does not use ConvGRU because we find it expansive and not impactful at this scale. The block only uses regular convolutions to refine the results. It first concatenates the input image and the bilinearly upsampled output from the previous block. Then it employs 2 repeated convolution, Batch Normalization, and ReLU stacks to produce the final hidden features. Finally, the features are projected to outputs, including 1-channel alpha prediction, 3-channel foreground prediction, and 1-channel segmentation prediction. The segmentation output is used for the segmentation training objective as later described in Section 4.

We find applying ConvGRU on half of the channels by split and concatenation effective and efficient. This design helps ConvGRU to focus on aggregating temporal information, while the other split branch forwards the spatial features specific to the current frame. All convolutions use 3×3 kernels, except the last projection uses a 1×1 kernel.

We modify our network such that it can be given T frames at once as input and each layer processes all T frames before passing to the next layer. During training, this allows Batch Normalization to compute statistics across both batch and time to ensure the normalization is consistent. During inference, $T = 1$ can be used to process live videos and $T > 1$ can be used to exploit more GPU parallelism from the non-recurrent layers as a form of batching if the frames are allowed to be buffered. Our recurrent decoder is uni-directional so it can be used for both live streaming and post-processing.

3.3. Deep Guided Filter Module

We adopt Deep Guided Filter (DGF) as proposed in [44] for high-resolution prediction. When processing high-resolution videos such as 4K and HD, we downsample the input frame by a factor s before passing through the encoder-decoder network. Then the low-resolution alpha, foreground, final hidden features, as well as the high-resolution input frame are provided to the DGF module to produce high-resolution alpha and foreground. The entire network is trained end-to-end as described in Section 4. Note that the DGF module is optional and the encoder-decoder network can operate standalone if the video to be processed is low in resolution.

Our entire network does not use any special operators and can be deployed to most existing inference frameworks. More architectural details are in the supplementary.

4. Training

We propose to train our network with both matting and semantic segmentation objectives simultaneously for several reasons:

First, human matting tasks are closely related to human segmentation tasks. Unlike trimap-based and background-based matting methods that are given additional cues as inputs, our network must learn to semantically understand the scene and be robust in locating the human subjects.

Second, most existing matting datasets only provide ground-truth alpha and foreground that must be synthetically composited to background images. The compositions sometimes look fake due to the foreground and the background having different lighting. On the other hand, semantic segmentation datasets feature real images where the human subjects are included in all types of complex scenes. Training with semantic segmentation datasets prevents our model from overfitting to the synthetic distribution.

Third, there is a lot more training data available for semantic segmentation tasks. We harvest a variety of publically available datasets, both video-based and image-based, to train a robust model.

4.1. Matting Datasets

Our model is trained on VideoMatte240K (VM) [22], Distinctions-646 (D646) [30], and Adobe Image Matting (AIM) [45] datasets. VM provides 484 4K/HD video clips. We divide the dataset into 475/4/5 clips for train/val/test splits. D646 and AIM are image matting datasets. We use only images of humans and combine them to form 420/15 train/val splits for training. For evaluation, D646 and AIM each provide 11 and 10 test images respectively.

For backgrounds, the dataset by [39] provides HD background videos that are suitable for matting composition. The videos include a variety of motions, such as cars passing, leaves shaking, and camera movements. We select 3118 clips that does not contain humans and extract the first 100 frames from every clip. We also crawl 8000 image backgrounds following the approach of [22]. The images have more indoor scenes such as offices and living rooms.

We apply motion and temporal augmentations on both foreground and background to increase data variety. Motion augmentations include affine translation, scale, rotation, sheer, brightness, saturation, contrast, hue, noise and blur that change continuously over time. The motion is applied with different easing functions such that the changes are not always linear. The augmentation also adds artificial motions to the image datasets. Additionally, we apply temporal augmentation on videos, including clip reversal, speed changes, random pausing, and frame skipping. Other discrete augmentations *i.e.* horizontal flip, grayscale, and sharpening, are applied consistently to all frames.

4.2. Segmentation Datasets

We use video segmentation dataset YouTubeVIS and select 2985 clips containing humans. We also use image segmentation datasets COCO [23] and SPD [40]. COCO provides 64,111 images containing humans while SPD provides additional 5711 samples. We apply similar augmentations but without motion, since YouTubeVIS already contains large camera movements and the image segmentation datasets do not require motion augmentation.

4.3. Procedures

Our matting training is pipelined into four stages. They are designed to let our network progressively see longer sequences and higher resolutions to save training time. We use Adam optimizer for training. All stages use batch size $B = 4$ split across 4 Nvidia V100 32G GPUs.

Stage 1: We first train on VM at low-resolution without the DGF module for 15 epochs. We set a short sequence length $T = 15$ frames so that the network can get updated quicker. The MobileNetV3 backbone is initialized with pretrained ImageNet [32] weights and uses $1e^{-4}$ learning rate, while the rest of the network uses $2e^{-4}$. We sample the height and width of the input resolution h, w independently between 256 and 512 pixels. This makes our network robust to different resolutions and aspect ratios.

Stage 2: We increase T to 50 frames, reduce the learning rate by half, and keep other settings from stage 1 to train our model for 2 more epochs. This allows our network to see longer sequences and learn long-term dependencies. $T = 50$ is the longest we can fit on our GPUs.

Stage 3: We attach the DGF module and train on VM with high-resolution samples for 1 epoch. Since high resolution consumes more GPU memory, the sequence length must be set to very short. To avoid our recurrent network overfitting to very short sequences, we train our network on both low-resolution long sequences and high-resolution short sequences. Specifically, the low-resolution pass does not employ DGF and has $T = 40$ and $h, w \sim (256, 512)$. The high-resolution pass entails the low-resolution pass and employs DGF with downsample factor $s = 0.25$, $\hat{T} = 6$ and $\hat{h}, \hat{w} \sim (1024, 2048)$. We set the learning rate of DGF to $2e^{-4}$ and the rest of the network to $1e^{-5}$.

Stage 4: We train on the combined dataset of D646 and AIM for 5 epochs. We increase the decoder learning rate to $5e^{-5}$ to let our network adapt and keep other settings from stage 3.

Segmentation: Our segmentation training is interleaved between every matting training iteration. We train the network on image segmentation data after every odd iteration, and on video segmentation data after every even ones. Segmentation training is applied to all stages. For video segmentation data, we use the same B, T, h, w settings following every matting stage. For image segmentation data, we treat them as video sequences of only 1 frame, so $T' = 1$. This gives us room to apply a larger batch size $B' = B \times T$. Since the images are feedforwarded as the first frame, it forces the segmentation to be robust even in the absence of recurrent information.

4.4. Losses

We apply losses on all $t \in [1, T]$ frames. To learn alpha α_t w.r.t. ground-truth α_t^*, we use L1 loss $\mathcal{L}_{l1}^{\alpha}$ and pyramid Laplacian loss $\mathcal{L}_{lap}^{\alpha}$, as reported by [9, 14] to produce the best result. We also apply a temporal coherence loss $\mathcal{L}_{tc}^{\alpha}$, as used by [39], to reduce flicker:

$$\mathcal{L}_{l1}^{\alpha} = ||\alpha_t - \alpha_t^*||_1 \tag{3}$$

$$\mathcal{L}_{lap}^{\alpha} = \sum_{s=1}^{5} \frac{2^{s-1}}{5} ||L_{pyr}^{s}(\alpha_t) - L_{pyr}^{s}(\alpha_t^*)||_1 \tag{4}$$

$$\mathcal{L}_{tc}^{\alpha} = ||\frac{\mathrm{d}\alpha_t}{\mathrm{d}t} - \frac{\mathrm{d}\alpha_t^*}{\mathrm{d}t}||_2 \tag{5}$$

To learn foreground F_t w.r.t. ground-truth F_t^*, We compute L1 loss $\mathcal{L}_{l1}^{F}$ and temporal coherence loss $\mathcal{L}_{tc}^{F}$ on pixels where $\alpha_t^* > 0$ following the approach of [22]:

$$\mathcal{L}_{l1}^{F} = ||(a_t^* > 0) * (F_t - F_t^*)||_1 \tag{6}$$

$$\mathcal{L}_{tc}^{F} = ||(a_t^* > 0) * (\frac{\mathrm{d}F_t}{\mathrm{d}t} - \frac{\mathrm{d}F_t^*}{\mathrm{d}t})||_2 \tag{7}$$

The total matting loss $\mathcal{L}^M$ is:

$$\mathcal{L}^M = \mathcal{L}_{l1}^{\alpha} + \mathcal{L}_{lap}^{\alpha} + 5\mathcal{L}_{tc}^{\alpha} + \mathcal{L}_{l1}^{F} + 5\mathcal{L}_{tc}^{F} \tag{8}$$

For semantic segmentation, our network is only trained on the human category. To learn the segmentation probability S_t w.r.t. the ground-truth binary label S_t^*, we compute binary cross entropy loss:

$$\mathcal{L}^S = S_t^*(-\log(S_t)) + (1 - S_t^*)(-\log(1 - S_t)) \tag{9}$$

5. Experimental Evaluation

5.1. Evaluation on Composition Datasets

We construct our benchmark by compositing each test sample from VM, D646, and AIM datasets onto 5 video and 5 image backgrounds. Every test clip has 100 frames. Image samples are applied with motion augmentation.

We compare our approach against state-of-the-art trimap-based method (FBA [9]), background-based method (BGMv2 [22] with MobileNetV2 [33] backbone), and auxiliary-free method (MODNet [18]). To fairly compare them for fully automatic matting, FBA uses synthetic trimaps generated by dilation and erosion of semantic segmentation method DeepLabV3 [3] with ResNet101 [12] backbone; BGMv2 only sees the first frame's ground-truth background; MODNet applies its neighbor frame smoothing trick. We attempt to re-train MODNet on our data but get worse results, potentially due to issues during the training, so MODNet uses its official weights; BGMv2 is already trained on all three datasets; FBA has not released the training code at the time of writing.

We evaluate α w.r.t. ground-truth α^* using MAD (mean absolute difference), MSE (mean squared error), Grad (spatial gradient) [31], and Conn (connectivity) [31] for quality, and adopt dtSSD [8] for temporal coherence. For F, we only measure pixels where $\alpha^* > 0$ by MSE. MAD and MSE are scaled by $1e^3$ and dtSSD is scaled by $1e^2$ for better readability. F is not measured on VM since it contains noisy ground-truth. MODNet does not predict F, so we evaluate on the input frame as its foreground prediction. This simulates directly applying the alpha matte on the input.

Table 1 compares methods using low-resolution input. Our method does not use DGF in this scenario. Ours predicts more accurate and consistent alpha across all datasets. In particular, FBA is limited by the inaccurate synthetic trimap. BGMv2 performs poorly for dynamic backgrounds. MODNet produces less accurate and coherent results than ours. For foreground prediction, ours is behind BGMv2 but outperforms FBA and MODNet.

Table 2 further compares our method with MODNet on high-resolution. Since DGF must be trained end-to-end with the network, we modify MODNet to use non-learned Fast Guided Filter (FGF) to upsample the prediction. Both methods use downsample scale $s = 0.25$ for the encoder-decoder network. We remove Conn metric because it is too

Dataset	Method	Alpha MAD	Alpha MSE	Alpha Grad	Alpha Conn	Alpha dtSSD	FG MSE
VM 512×288	DeepLabV3	14.47	9.67	8.55	1.69	5.18	
	FBA	8.36	3.37	2.09	0.75	2.09	
	BGMv2	25.19	19.63	2.28	3.26	2.74	
	MODNet	9.41	4.30	1.89	0.81	2.23	
	Ours	**6.08**	**1.47**	**0.88**	**0.41**	**1.36**	
D646 512×512	DeepLabV3	24.50	20.1	20.30	6.41	4.51	
	FBA	17.98	13.40	7.74	4.65	2.36	5.84
	BGMv2	43.62	38.84	5.41	11.32	3.08	**2.60**
	MODNet	10.62	5.71	3.35	2.45	1.57	6.31
	Ours	**7.28**	**3.01**	**2.81**	**1.83**	**1.01**	2.93
AIM 512×512	DeepLabV3	29.64	23.78	20.17	7.71	4.32	
	FBA	23.45	17.66	9.05	6.05	2.29	6.32
	BGMv2	44.61	39.08	5.54	11.60	2.69	**3.31**
	MODNet	21.66	14.27	5.37	5.23	1.76	9.51
	Ours	**14.84**	**8.93**	**4.35**	**3.83**	**1.01**	5.01

Table 1: Low-resolution comparison. Our alpha prediction is better than all others. Our foreground prediction is behind BGMv2 but outperforms FBA and MODNet. Note that FBA uses synthetic trimap from DeepLabV3; BGMv2 only sees ground-truth background from the first frame; MODNet does not predict foreground so it is evaluated on the input image.

Dataset	Method	SAD	MSE	Grad	dtSSD
VM 1920×1080	MODNet + FGF	11.13	5,54	15.30	3.08
	Ours	**6.57**	**1.93**	**10.55**	**1.90**
D646 2048×2048	MODNet + FGF	11.27	6.13	30.78	2.19
	Ours	**8.67**	**4.28**	**30.06**	**1.64**
AIM 2048×2048	MODNet + FGF	17.29	10.10	35.52	2.60
	Ours	**14.89**	**9.01**	**34.97**	**1.71**

Table 2: High-resolution alpha comparison. Ours is better than MODNet with Fast Guided Filter (FGF).

expansive to compute at high-resolution. Our method outperforms MODNet on all metrics.

5.2. Evaluation on Real Videos

Figure 3 shows qualitative comparisons on real-videos. In Figure 3a, we compare alpha predictions across all methods and find ours predicts fine-grained details like hair strands more accurately. In Figure 3b, we experiment on random YouTube videos. We remove BGMv2 from the comparison since these videos do not have pre-captured backgrounds. We find our method is much more robust to semantic errors. In Figures 3c and 3d, we further compare real-time matting against MODNet on cellphone and webcam videos. Our method can handle fast-moving body parts better than MODNet.

5.3. Size and Speed Comparison

Tables 3 and 4 show that our method is significantly lighter, with only 58% parameters compared to MODNet. Ours is the fastest on HD (1920 × 1080), but a little slower than BGMv2 on 512 × 288 and MODNet with FGF on 4K (3840 × 2160). Our inspection finds that DGF and FGF incur very minor differences in performance. Our method is slower than MODNet in 4K because ours predicts foreground in addition to alpha, so it is slower to process 3 extra channels in high resolution. We use [37] to measure GMACs (multiply–accumulate operations), but it only measures convolutions and misses out resize and many tensor operations which are used most in DGF and FGF, so GMACs is only a rough approximation. Our method achieves HD 104 FPS and 4K 76 FPS, which is considered real-time for many applications.

Method	Parameters (Million)	Size (MB)
DeepLabV3	60.996	233.3
FBA	34.693	138.8
BGMv2	5.007	19.4
MODNet	6.487	25.0
Ours	**3.749**	**14.5**

Table 3: Ours is lighter than all compared methods. Size is measured on FP32 weights.

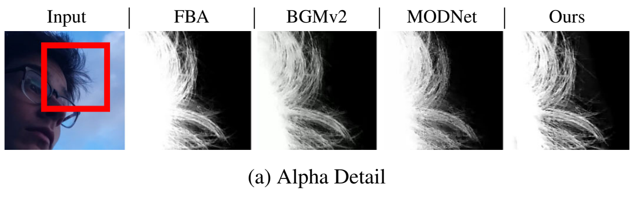
Resolution	s	Method	FPS	GMACs*
512×288	1	DeepLabV3 + FBA	12.3	205.77
		BGMv2	**152.5**	8.46
		MODNet	104.9	8.80
		Ours	131.9	**4.57**
1920×1080	0.25	BGMv2	70.6	9.86
		MODNet + FGF	100.3	7.78
		Ours	**104.2**	**4.15**
3840×2160	0.125	BGMv2	26.5	17.04
		MODNet + FGF	**88.6**	7.78
		Ours	76.5	**4.15**

Table 4: Model performance comparison. s denotes the downsample scale. Models are converted to TorchScript and optimized before testing (BatchNorm fusion *etc.*). FPS is measured as FP32 tensor throughput on an Nvidia GTX 1080Ti GPU. GMACs is a rough approximation.

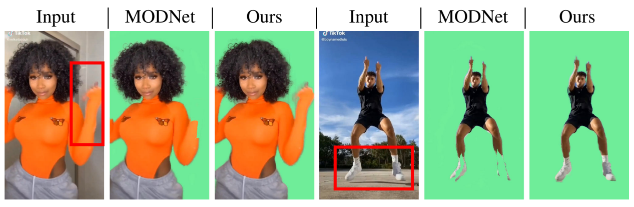
6. Ablation Studies

6.1. Role of Temporal Information

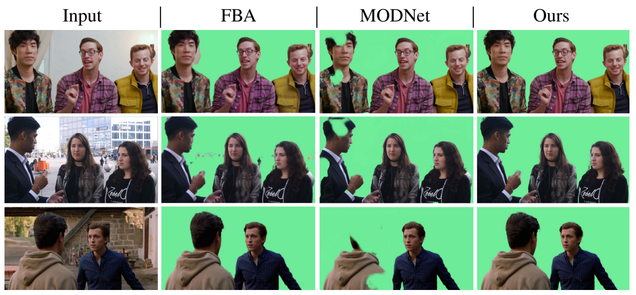
Figure 4 shows the change of average alpha MAD metric across all VM test clips over time. The error of our model drops significantly in the first 15 frames then the metric stays stable. MODNet, even with its neighbor frame



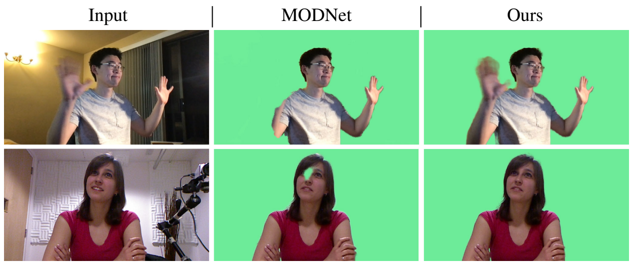
(a) Alpha Detail



(c) Cellphone Videos



(b) YouTube Videos



(d) Webcam Videos

Figure 3: Qualitative comparison. Our method produces more detailed alpha compared to others. When evaluating on YouTube, cellphone, and webcam videos, ours is consistently more robust than others. See supplementary for more results. YouTube videos are crawled from the Internet; cellphone videos are from a public dataset [17]; Some webcam examples are recorded while others are taken from [24].

smoothing trick, has large fluctuations in the metric. We also experiment with disabling recurrence in our network by passing zero tensors as the recurrent states. The quality and consistency worsen as expected. This proves that temporal information improves quality and consistency.

Figure 5 compares temporal coherence with MODNet on a video sample. Our method produces consistent results on the handrail region while MODNet produces flicker, which significantly degrade perceptual quality. Please see our supplementary for more results.

We further examine the recurrent hidden state. In Figure 6, we find that our network has automatically learned to reconstruct the background as it is revealed over time and keep this information in its recurrent channels to help future predictions. It also uses other recurrent channels to keep track of motion history. Our method even attempts to reconstruct the background when the videos contain camera movements and is capable of forgetting useless memory on shot cuts. More examples are in the supplementary.

6.2. Role of Segmentation Training Objective

Table 5 shows that our method is as robust as the semantic segmentation methods when evaluated on the subset of COCO validation images that contain humans and only on the human category. Our method achieves 61.50 mIOU, which is reasonably in between the performance of MobileNetV3 and DeepLabV3 trained on COCO considering the difference in model size. We also try to evaluate the robustness of our alpha output by thresholding $\alpha > 0.5$ as the binary mask and our method still achieves 60.88 mIOU, showing that the alpha prediction is also robust. For comparison, we train a separate model by initializing our

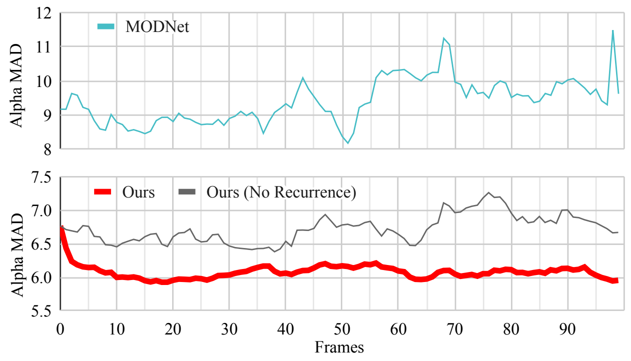


Figure 4: Average alpha MAD over time on VM without DGF. Our metric improves over time and is stable, showing that temporal information improves quality and consistency.

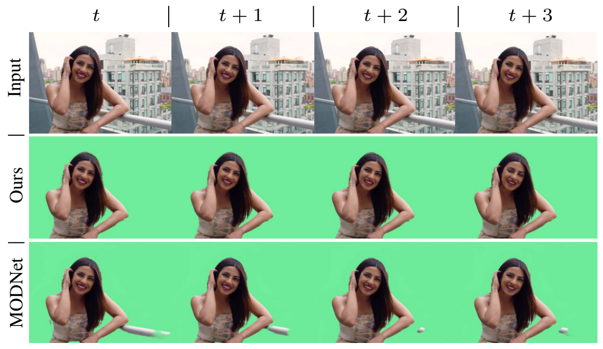


Figure 5: Temporal coherence comparison. MODNet's result has flicker on the handrail while ours is consistent.

MobileNetV3 encoder and LR-ASPP module with the pretrained weights on COCO and removing the segmentation objective. The model overfits to the synthetic matting data and regresses significantly on COCO performance, achieving only 38.24 mIOU.

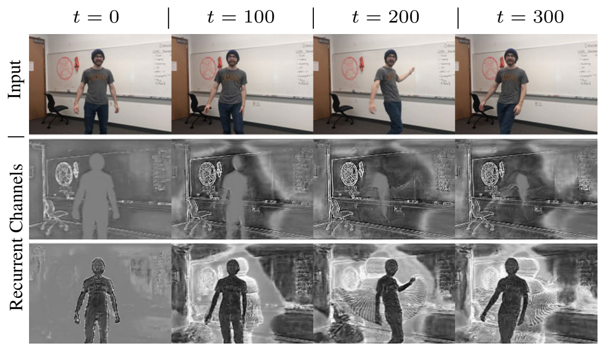


Figure 6: Two example channels in the recurrent hidden state. Our network learns to to reconstruct the background over time and keeps track of the motion history in its recurrent state.

Method	mIOU
DeepLabV3	68.93
MobileNetV3 + LR-ASPP	58.58
Ours (alpha output, no seg objective)	38.24
Ours (alpha output)	**60.88**
Ours (segmentation output)	**61.50**

Table 5: Segmentation performance on COCO validation set. Training with segmentation objective makes our method robust while training only with pre-trained weights regresses.

6.3. Role of Deep Guided Filter

Table 6 shows that DGF has only a small overhead in size and speed compared to FGF. DGF has a better Grad metric, indicating its high-resolution details are more accurate. DGF also produces more coherent results indicated by the dtSSD metric, likely because it takes hidden features from the recurrent decoder into consideration. The MAD and MSE metrics are inconclusive because they are dominated by segmentation-level errors, which are not corrected by either DGF or FGF.

Method	Params	FPS	MAD	MSE	Grad	dtSSD
Ours (FGF)	**3.748**	**109.4**	8.70	**4.13**	31.44	1.89
Ours	3.749	104.2	**8.67**	4.28	**30.06**	**1.64**

Table 6: Comparing switching DGF to FGF on D646. Parameters are measured in millions. FPS is measured in HD.

6.4. Static vs. Dynamic Backgrounds

Table 7 compares the performance on static and dynamic backgrounds. Dynamic backgrounds include both background object movements and camera movements. Our method can handle both cases and performs slightly better on static backgrounds, likely because it is easier to reconstruct pixel-aligned backgrounds as shown in Figure 6. On the other hand, BGMv2 performs badly on dynamic backgrounds and MODNet does not exhibit any preference. In metric, BGMv2 outperforms ours on static backgrounds, but it is expected to do worse in reality when the pre-captured background has misalignment.

Background	Method	MAD	MSE	Grad	dtSSD
Static	BGMv2*	**4.33**	**0.32**	**4.19**	**1.33**
	MODNet + FGF	11.04	5.42	15.80	3.10
	Ours	5.64	1.07	9.80	1.84
Dynamic	BGMv2	42.45	37.05	17.30	4.61
	MODNet + FGF	11.23	5.65	14.79	3.06
	Ours	**7.50**	**2.80**	**11.30**	**1.96**

Table 7: Comparing VM samples on static and dynamic backgrounds. Ours does better on static backgrounds but can handle both cases. Note that BGMv2 receives ground-truth static backgrounds, but in reality the backgrounds have misalignment.

6.5. Larger Model for Extra Performance

We experiment with switching the backbone to ResNet50 [12] and increasing the decoder channels. Table 8 shows the performance improvement. The large model is more suitable for server-side applications.

Method	Params	Size	FPS	MAD	MSE	Grad	dtSSD
Ours Large	26.890	102.9	71.1	**5.81**	**0.97**	**9.65**	**1.78**
Ours	**3.749**	**14.5**	**104.2**	6.57	1.93	10.55	1.90

Table 8: Large model uses ResNet50 backbone and has more decoder channels. Evaluated on VM in HD. Size is measured in MB.

6.6. Limitations

Our method prefers videos with clear target subjects. When there are people in the background, the subjects of interest become ambiguous. It also favors simpler backgrounds to produce more accurate matting. Figure 7 shows examples of challenging cases.

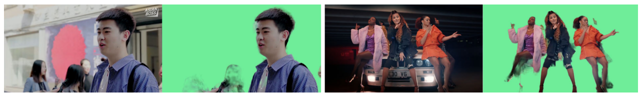


Figure 7: Challenging cases. People in the background make matting target ambiguous. Complex scenes make matting harder.

7. Conclusion

We have proposed a recurrent architecture for robust human video matting. Our method achieves new state-of-the-art while being lighter and faster. Our analysis shows that temporal information plays an important role in improving the quality and consistency. We also introduce a new training strategy to train our model on both matting and semantic segmentation objectives. This approach effectively enforces our model to be robust on various types of videos.

References

[1] 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.

[2] Nicolas Ballas, L. Yao, C. Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. *CoRR*, abs/1511.06432, 2016.

[3] Liang-Chieh Chen, G. Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *ArXiv*, abs/1706.05587, 2017.

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

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

[6] Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *ArXiv*, abs/1406.1078, 2014.

[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.

[8] M. Erofeev, Yury Gitman, D. Vatolin, Alexey Fedorov, and J. Wang. Perceptually motivated benchmark for video matting. In *BMVC*, 2015.

[9] Marco Forte and François Pitié. F, b, alpha matting. *CoRR*, abs/2003.07711, 2020.

[10] 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.

[11] Kaiming He, Jian Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1397–1409, 2013.

[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.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[14] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation, 2019.

[15] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[17] Yasamin Jafarian and Hyun Soo Park. Learning high fidelity depths of dressed humans by watching social media dance videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12753–12762, June 2021.

[18] Zhanghan Ke, Kaican Li, Yurou Zhou, Qiuhua Wu, Xiangyu Mao, Qiong Yan, and Rynson W.H. Lau. Is a green screen really necessary for real-time portrait matting? *ArXiv*, abs/2011.11961, 2020.

[19] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross B. Girshick. Pointrend: Image segmentation as rendering. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9796–9805, 2020.

[20] 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.

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

[22] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *Computer Vision and Pattern Regognition (CVPR)*, 2021.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[24] Marwa Mahmoud, Tadas Baltrušaitis, Peter Robinson, and Laurel Riek. 3d corpus of spontaneous complex mental states. In *Conference on Affective Computing and Intelligent Interaction*, 2011.

[25] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.

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

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[28] Andreas Pfeuffer, Karina Schulz, and K. Dietmayer. Semantic segmentation of video sequences with convolutional lstms. *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1441–1447, 2019.

[29] 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.

[30] Yu Qiao, Yuhao Liu, Xin Yang, Dongsheng Zhou, Mingliang Xu, Qiang Zhang, and Xiaopeng Wei. Attention-guided hierarchical structure aggregation for image matting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] Christoph Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *CVPR*, 2009.

[32] Olga Russakovsky, J. Deng, Hao Su, J. Krause, S. Satheesh, S. Ma, Zhiheng Huang, A. Karpathy, A. Khosla, Michael S. Bernstein, A. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[34] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *Computer Vision and Pattern Regognition (CVPR)*, 2020.

[35] 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.

[36] Xingjian Shi, Zhourong Chen, Hao Wang, D. Yeung, W. Wong, and W. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

[37] Vladislav Sovrasov. flops-counter.pytorch.

[38] 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.

[39] Yanan Sun, Guanzhi Wang, Qiao Gu, Chi-Keung Tang, and Yu-Wing Tai. Deep video matting via spatio-temporal alignment and aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[40] supervise.ly. Supervisely person dataset. *supervise.ly*, 2018.

[41] Pavel Tokmakov, Alahari Karteek, and C. Schmid. Learning video object segmentation with visual memory. In *ICCV*, 2017.

[42] C. Ventura, Miriam Bellver, Andreu Girbau, A. Salvador, F. Marqués, and Xavier Giró i Nieto. Rvos: End-to-end recurrent network for video object segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5272–5281, 2019.

[43] 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.

[44] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter, 2019.

[45] 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.

[46] 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.

[47] 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.

A. Overview

We provide additional details in this supplementary. In Section B, we describe the details of our network architecture. In Section C, we explain the details on training. In Section D, we show examples of our composited matting data samples. In Section E, we show additional results from our method. We also attach video results in the supplementary. Please see our videos for better visualization.

B. Network

Backbone	$E_{\frac{1}{2}}$	$E_{\frac{1}{4}}$	$E_{\frac{1}{8}}$	$E_{\frac{1}{16}}$	AS	$D_{\frac{1}{16}}$	$D_{\frac{1}{8}}$	$D_{\frac{1}{4}}$	$D_{\frac{1}{2}}$	$D_{\frac{1}{1}}$
Ours	**16**	**24**	**40**	**960**	**128**	**128**	**80**	**40**	**32**	**16**
Ours Large	64	256	512	2048	256	256	128	64	32	16

Table 9: Feature channels at different scale. E_k and D_k denote encoder and decoder channels at k feature scale respectively. AS denotes LR-ASPP channels.

Table 9 describes our network and its variants with feature channels. Our default network uses MobileNetV3-Large [15] backbone while the large variant uses ResNet50 [12] backbone.

Encoder: The encoder backbone operates on individual frames and extracts feature maps of E_k channels at $k \in [\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}]$ scales. Unlike regular MobileNetV3 and ResNet backbones that continue to operate at $\frac{1}{32}$ scale, we modify the last block to use convolutions with a dilation rate of 2 and a stride of 1 following the design of [3, 4, 15]. The last feature map $E_{\frac{1}{16}}$ is given to the LR-ASPP module, which compresses it to AS channels.

Decoder: All ConvGRU layers operate on half of the channels by split and concatenation, so the recurrent hidden state has $\frac{D_k}{2}$ channels at scale k. For the upsampling blocks, the convolution, Batch Normalization, and ReLU stack compresses the concatenated features to D_k channels before splitting to ConvGRU. For the output block, the first two convolutions have 16 filters and the final hidden features has 16 channels. The final projection convolution outputs 5 channels, including 3-channel foreground, 1-channel alpha, and 1-channel segmentation predictions. All convolutions uses 3×3 kernels except the last projection uses a 1×1 kernel. The average poolings use 2×2 kernels with a stride of 2.

Deep Guided Filter: DGF contains a few 1×1 convolutions internally. We modify it to take the predicted foreground, alpha, and the final hidden features as inputs. All internal convolutions use 16 filters. Please refer to [44] for more specifications.

Our entire network is built and trained in PyTorch [27]. We clamp the alpha and foreground prediction outputs to $[0, 1]$ range without activation functions following [9, 22]. The clamp is done during both training and inference. The segmentation prediction output is sigmoid logits.

C. Training

Algorithm 1: Training Procedures

```
for stage $\in [1, 2, 3, 4]$ do
    for epoch do
        for iteration do
            LowResMattingPass($B, T, h, w$)
            if stage $\in [3, 4]$ then
                HighResMattingPass($B, \hat{T}, \hat{h}, \hat{w}$)
            if iteration % 2 = 0 then
                VideoSegmentationPass($B, T, h, w$)
            else
                ImageSegmentationPass($B', 1, h, w$)
```

Algorithm 1 shows the training loop of our proposed training strategy. The sequence length parameters T, $\hat{T}$ are set according to the stages, which is specified in our main text; the batch size parameters are set to $B = 4$, and $B' = B \times T$; The input resolutions are randomly sampled as $h, w \sim Uniform(256, 512)$ and $\hat{h}, \hat{w} \sim Uniform(1024, 2048)$.

Our network is trained using 4 Nvidia V100 32G GPUs. We use mixed precision training [25] to reduce the GPU memory consumption. The training takes approximately 18, 2, 8, and 14 hours in each stage respectively.

D. Data Samples

Figure 8 shows examples of composited training samples from the matting datasets. The clips contain natural movements when compositing with videos as well as artificial movements generated by the motion augmentation.

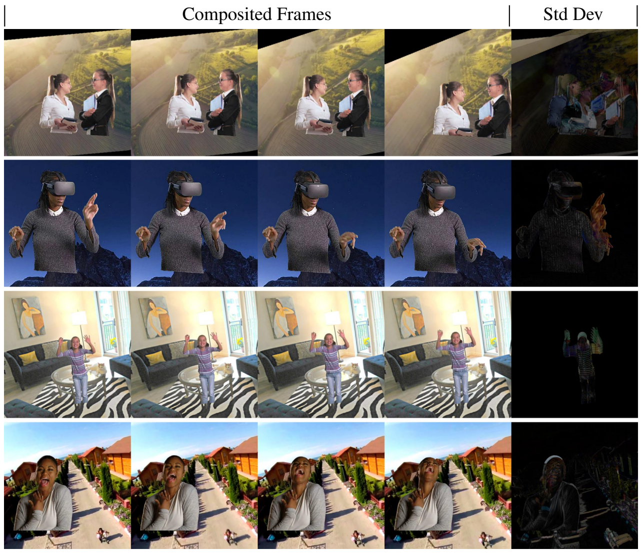


Figure 8: Composited training samples. Last column shows the standard deviation of each pixel across time to visualize motion.

Figure 9 shows examples of the composited testing samples. The testing samples only apply motion augmentation on image foreground and backgrounds. The motion augmentation only consists of affine transforms. The strength of the augmentation is also weaker compared to the training augmentation to make testing samples as realistic looking as possible.

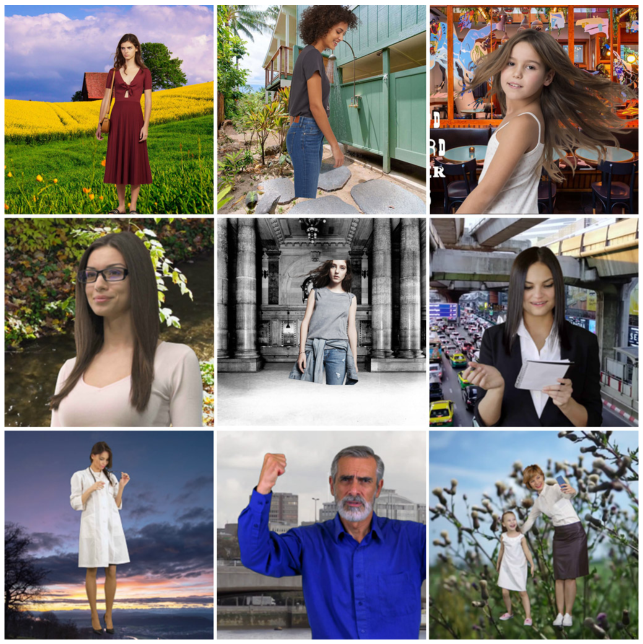


Figure 9: Example testing samples. The augmentation is only applied on image foreground and background. The augmentation strength is weaker to make samples look more realistic.

E. Additional Results

Figure 10 shows additional qualitative comparisons with MODNet. Our method is consistently more robust. Figure 11 compares temporal coherence with MODNet. MODNet has flicker on low-confidence regions whereas our results are coherent. Figure 12 shows additional examples of our model's recurrent hidden state. It shows that our model has learned to store useful temporal information in its recurrent state and is capable of forgetting useless information upon shot cuts.

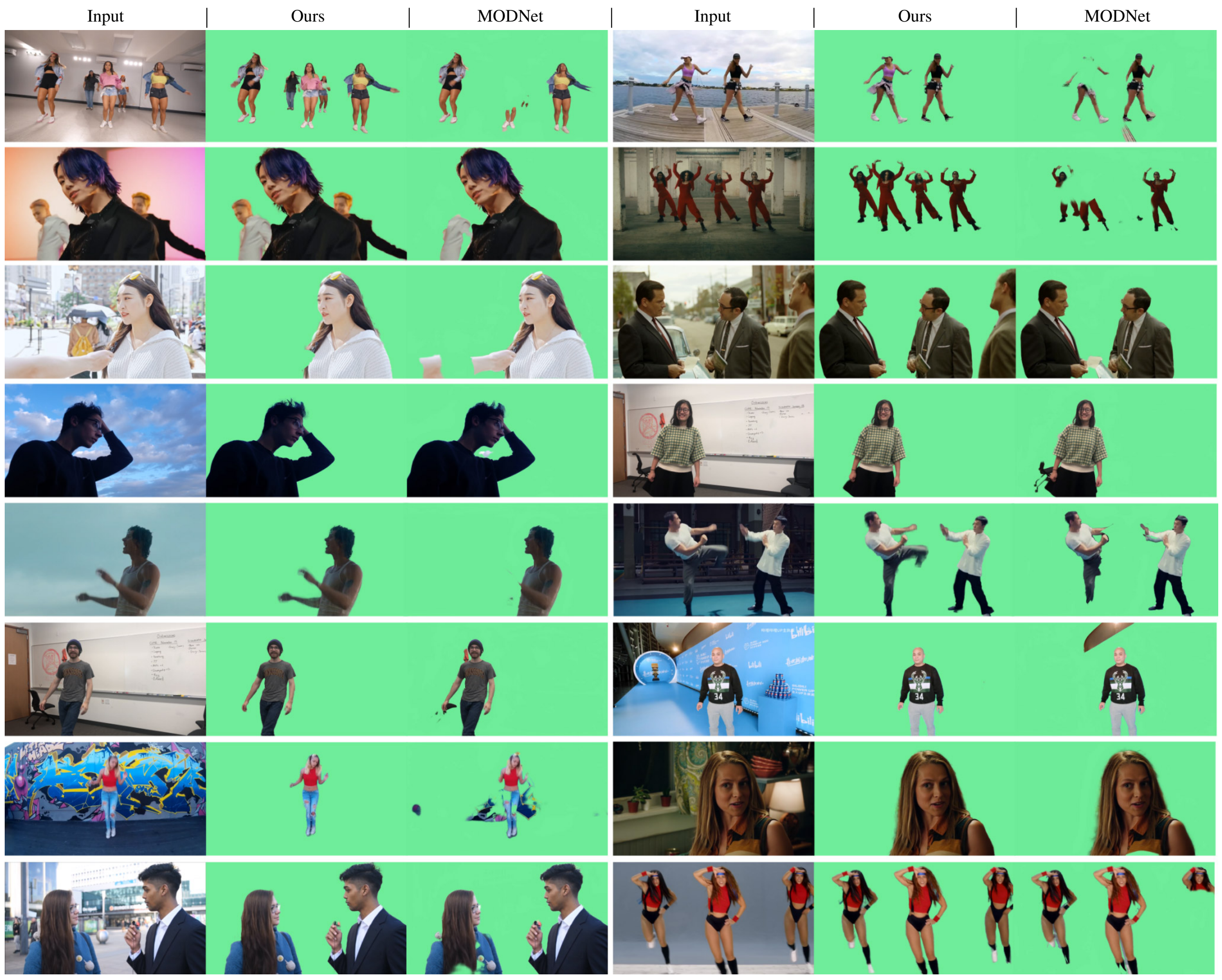


Figure 10: More qualitative comparison with MODNet.

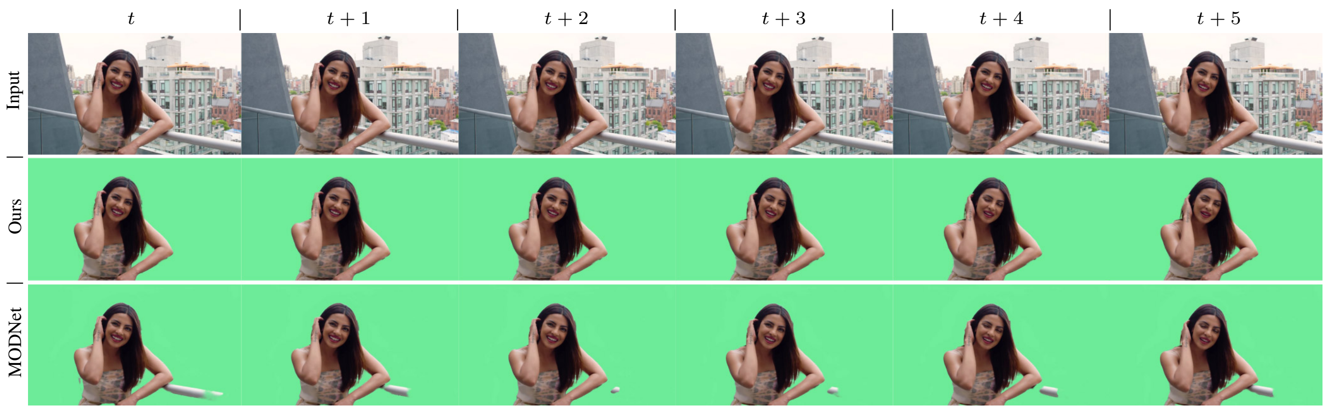
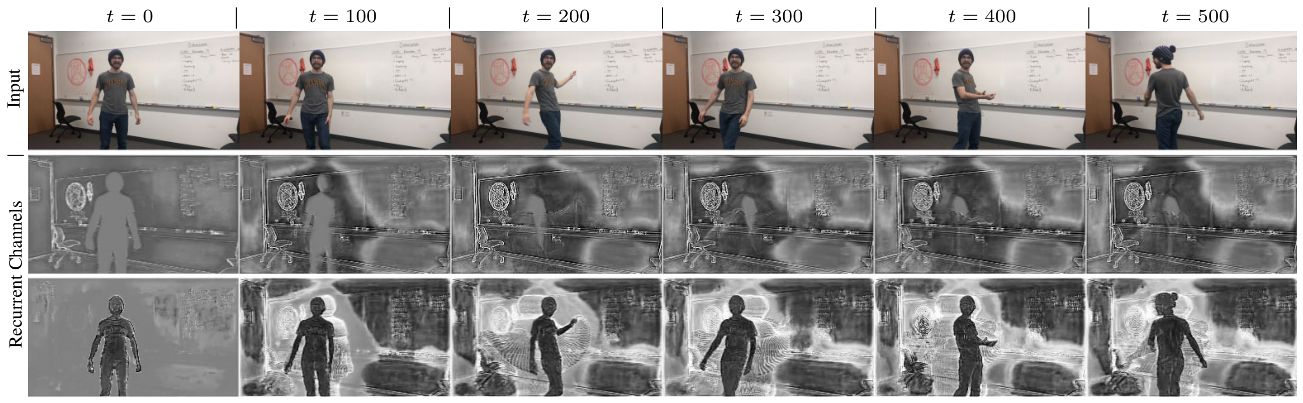
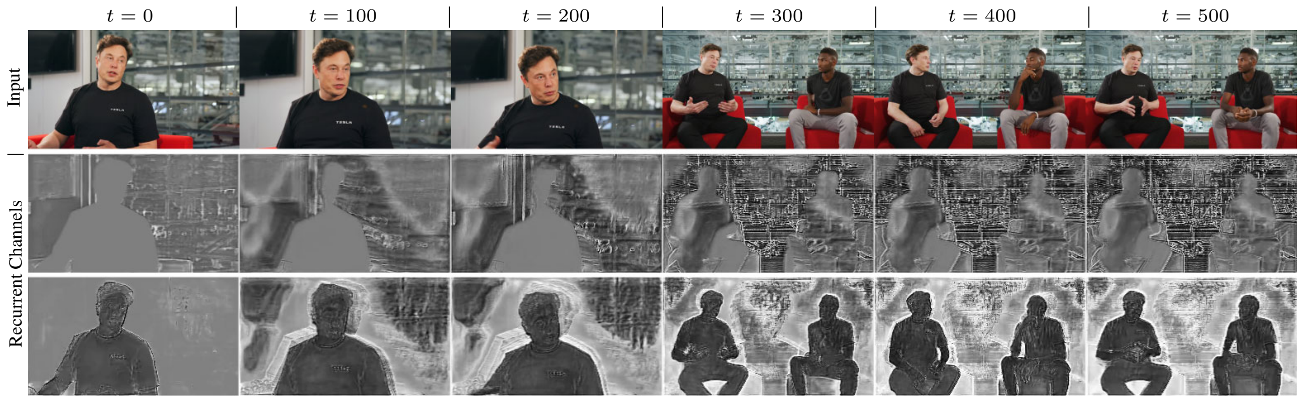


Figure 11: Temporal coherence comparison. Our result is temporally coherent, whereas MODNet produces flicker around the handrail. This is because MODNet processes every frame as indepdendent images, so its matting decision is not consistent.



(a) Video with a static background.



(b) Video with a handheld camera and cut shots.

Figure 12: More examples of the recurrent hidden states. The first example with the static background clearly shows our model reconstructs the occluded background region over time. The second example with a handheld camera shows that our model still attempts to reconstruct the background, and it has learned to forget useless recurrent states on shot cuts.