Robust High-Resolution Video Matting with Temporal Guidance Supplementary

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

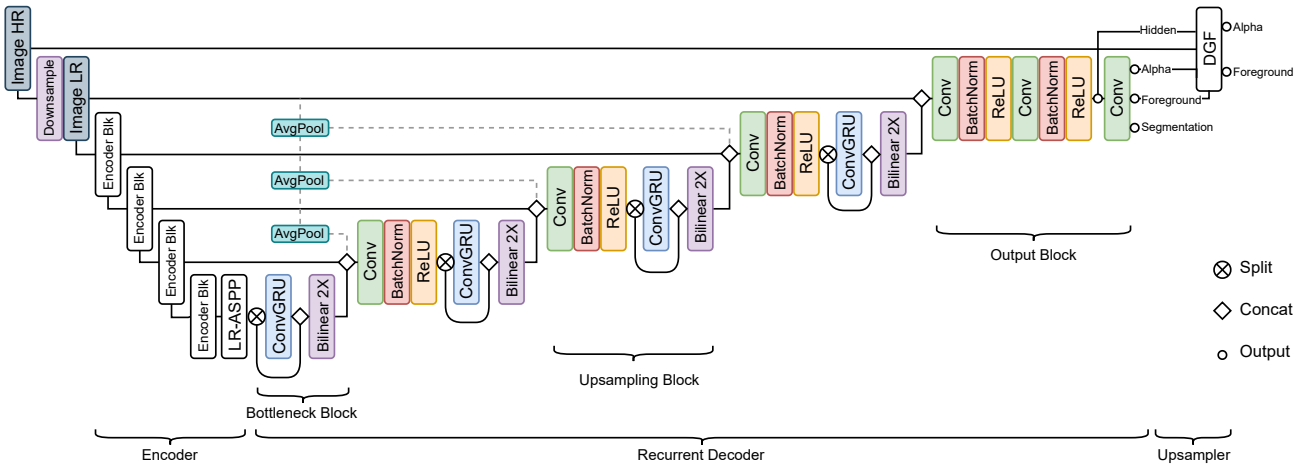


Figure 1: Model architecture. The diagram is copied from the main text for reference. Our network consists an encoder that extracts individual frame's features, a recurrent decoder that aggregates temporal information, and a Deep Guided Filter (DGF) module for high-resolution upsampling. When processing high-resolution frames, the input is first downsampled to pass through the encoder-decoder network.

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 1: 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 1 describes our network and its variants with feature channels. Our default network uses MobileNetV3-Large [5] backbone while the large variant uses ResNet50 [4] 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 [1, 2, 5]. 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.

*Work performed during an internship at ByteDance.

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 [9] for more specifications.

Our entire network is built and trained in PyTorch [8]. We clamp the alpha and foreground prediction outputs to $[0, 1]$ range without activation functions following [3, 6]. The clamp is done during both training and inference. The segmentation prediction output is sigmoid logits.

C. Training

Algorithm 1: Training Procedures

```
for stage ∈ [1, 2, 3, 4] do
    for epoch do
        for iteration do
            LowResMattingPass(B, T, h, w)
            if stage ∈ [3, 4] then
                HighResMattingPass(B, T̂, ĥ, ŵ)
            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 [7] to reduce the GPU memory consumption. The training takes approximately 18, 2, 8, and 14 hours in each stage respectively.

D. Data Samples

Figure 2 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 3 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.

E. Additional Results

Figure 4 shows additional qualitative comparisons with MODNet. Our method is consistently more robust. Figure 5 compares temporal coherence with MODNet. MODNet has flicker on low-confidence regions whereas our results are coherent. Figure 6 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.

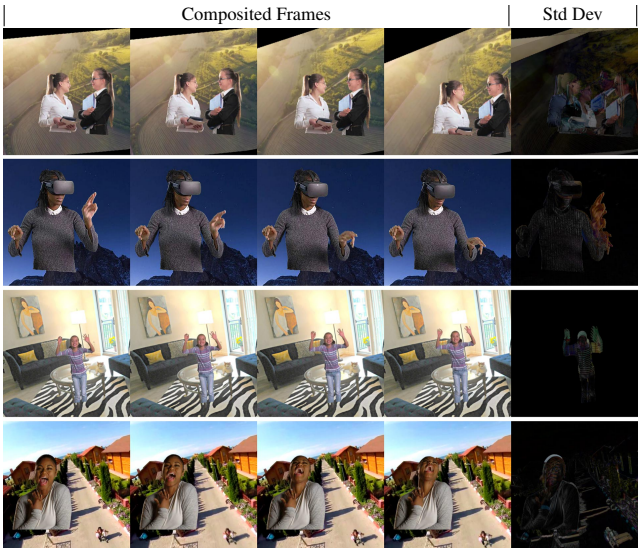


Figure 2: Composited training samples. Last column shows the standard deviation of each pixel across time to visualize motion.

Figure 3: Example testing samples. The augmentation is only applied on image foreground and background. The augmentation strength is weaker to make samples look more realistic.

Figure 4: More qualitative comparison with MODNet.

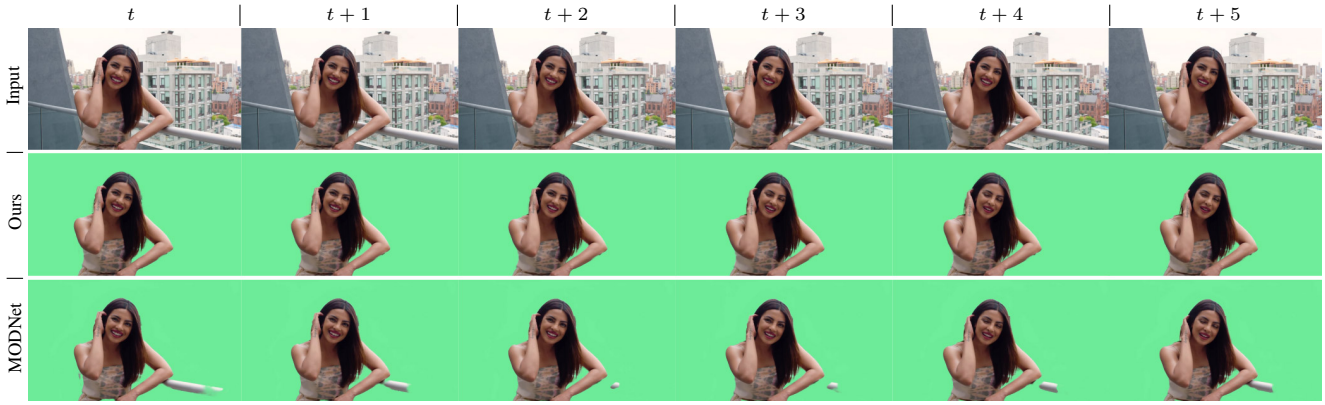
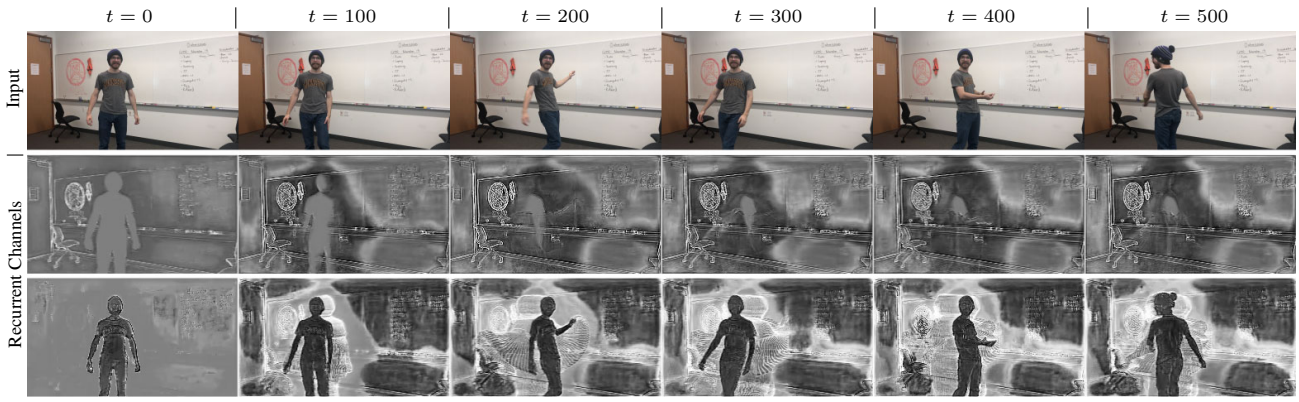
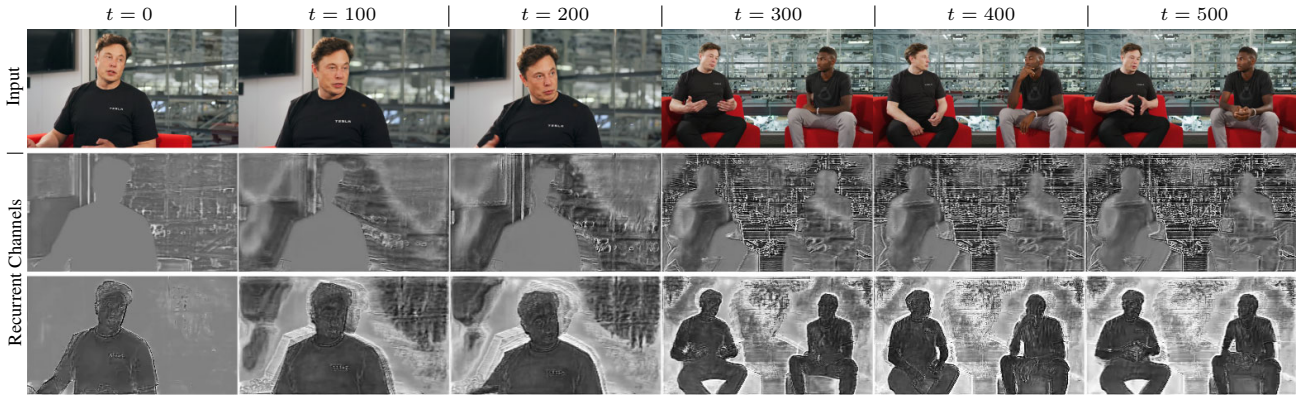


Figure 5: Temporal coherence comparison. Our result is temporally coherent, whereas MODNet produces flicker around the handrail. This is because MODNet processes every frame as indepdendent images, so its matting decision is not consistent.



(a) Video with a static background.



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

Figure 6: 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.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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