MatAnyone: Stable Video Matting with Consistent Memory Propagation
— Supplementary Materials —

In this supplementary material, we provide additional discussions and results to supplement the main paper. In Section A, we present the network details of our MatAnyone. In Section B, we discuss more training details, including training schedules, training augmentations, and loss functions. In Section C, we provide more details on our new training and testing datasets, including the generation pipeline and some examples for demonstration. We present comprehensive results in Section D to further show our performance, including those for ablation studies and qualitative comparisons. It is noteworthy that we also include a demo video (Section D.6) to showcase a Hugging Face demo and additional results on real-world cases in video format.

Contents

A. Architecture

A.1. Network Designs

As illustrated in Fig. 3 in the main paper, our MatAnyone mainly has five important components: (1) an *encoder* for key and query transformation, (2) a *consistent memory propagation* module for pixel memory readout, (3) an object transformer [3] for memory grouping by object-level semantics, (4) a *decoder* for alpha matte decoding, (5) a *value encoder* for alpha matte encoding, which is used to update the alpha memory bank.

Encoder. We adopt ResNet-50 [5] for encoder following common practices in memory-based VOS [1–3]. Discarding the last convolution stage, we take $\times 16$ downsampled feature as F^t for key and query transformation, while features at scales $\times 8$, $\times 4$, $\times 2$, and $\times 1$ are used as skip connections for the decoder.

Consistent Memory Propagation. The process of consistent memory propagation is detailed in Fig. 3(b) in the main paper. *Alpha memory bank* serves as the main working memory for past information query as in [1, 3], which is updated every r^{th} frame across the whole time span. The query of the current frame to the alpha memory bank is implemented in an *attention* manner following [1, 3]. For the query $Q^{HW\times C}$ [1] and alpha memory bank $K^{THW\times C}$, $V^{THW\times C^v}$ [2], the affinity matrix $A \in [0,1]^{HW\times THW}$ of the query to alpha memory is computed as:

$$A_{ij} = \frac{exp(d(Q_i, K_j))}{\sum_z exp(d(Q_i, K_z))}, \tag{1}$$

where $d(\cdot,\cdot)$ is the anisotropic L2 function, H and W are the height and width at $\times 16$ downsampled input scale, and T is the number of memory frames stored in alpha memory bank. The queried values V_t^m in Fig. 3(b) in the main manuscript is obtained as:

$$V_t^m = AV_m. \tag{2}$$

In addition to that, we also maintain *last frame memory* solely for the uncertainty prediction module we propose, and it is updated every frame. The *boundary-area prediction* module is lightweight with one 1×1 convolution and two 3×3 convolutions. By taking the input of a concatenation of current frame feature K_t, last frame feature K_{t-1}, and last alpha matte prediction M_{t-1}, it outputs a one-channel change probability mask U_t of each query token, where higher U_t indicates such token is likely to change more in the alpha value compared with M_{t-1}. As mentioned in Sec. 3.1 in the manuscript, the ground truth U_t label is obtained by: $U_t^{GT} : |M_{t-1}^{GT} - M_t^{GT}| >= \delta$, where δ is set at 0 for segmentation data, and 0.001 for matting data as noise tolerance. Since U_t is predicted at a $\times 16$ downsampled scale in the memory space, the ground truth mask U_t^{GT} is also downsampled in the mode of `area`.

Object Transformer. Our object transformer is derived from Cutie [3] with three consecutive object transformer blocks. Pixel memory readout P^t obtained from the consistent memory propagation module is then grouped through several attention layers and feed-forward networks. In this way, the noise brought by low-level pixel matching could be effectively reduced for a more robust matching against distractors. We do not claim contributions for this module.

Decoder. Our decoder is inspired by common practices in VOS [1, 3] with modified designs specifically for the matting tasks. The mask decoder is VOS generally consists of two interactive upsampling from $\times 16$ to $\times 4$, and then a bilinear interpolation is applied to the input scale. However, since the boundary region for an alpha matte requires much more precision than a segmentation mask, we enrich the decoder with two more upsampling layers until $\times 1$, where skip connections from the encoder are applied at each scale to enhance the boundary precision.

Value Encoder. Similar to the encoder, we adopt ResNet-18 [5] for value encoder following common practices in memory-based VOS [1–3]. Different from the encoder for key and query, the value encoder takes the predicted alpha matte M^t as well as the image features as input, the encoded values are then used to update the alpha memory bank and last frame memory according to their updating rules.

B. Training

B.1. Training Schedules

Stage 1. To initialize our model on memory propagation learning, we train with our new video matting data *VM800*, which is of larger scale, higher quality, and better diversity than VideoMatte240K [12]. We use the AdamW [15] optimizer with a learning rate of 1×10^{-4} with a weight decay 0.001. The batch size is set to 16. We train with a short sequence length of

[1]We ignore the subscript t in Q_t for simplicity
[2]We ignore the subscript m in K_m and V_m for simplicity

Table A. **Training settings and losses used in different training stages.** † indicates that segmentation loss is computed as an auxiliary loss on a *segmentation* head, which will be abandoned during inference. Other than that, matting loss and core supervision loss are computed on the *matting* head for semantic stability in core regions and matting details in the boundary region.

Training Stage	#Iterations	Matting Data	Segmentation Data	Sequence Length	Matting Loss	Segmentation Loss†	Core Supervision Loss
Stage 1	85K	video	image & video	3 (80K) $\rightarrow$ 8 (5K)	✓	✓	
Stage 2	40K	video	image & video	8	✓	✓	✓
Stage 3	5K	image	image & video	8	✓	✓	✓

3 for 80K first, and then we train with a longer sequence length of 8 for another 5K for more complex scenarios. Video and image segmentation data COCO [14], SPD [17] and YouTubeVIS [18] are used to train the segmentation head parallel to the matting head at the same time, as previous practices [7, 11, 13].

Stage 2. We apply our key training strategy - *core-area supervision* in this stage. On the basis of the previous stage, we add additional supervision on the matting head with segmentation data to enhance the semantics robustness and generalizability towards real cases. In this stage, the learning rate is set to be 1×10^{-5}, and we train with a sequence length of 8 for 40K for both matting and segmentation data.

Stage 3. Due to the inferior quality of video matting data compared with image matting data annotated by humans, we finetune our model with image matting data instead for 5K with a 1×10^{-6} learning rate. Noticeable improvements in matting details, especially among boundary regions, could be seen after this stage.

B.2. Training Augmentations

Augmentations for Training Data. As discussed in the manuscript, video matting data are deficient in quantity and diversity. In order to enhance training data variety during the composition process, we follow RVM [13] to apply motion (*e.g.*, affine translation, scale, rotation, etc.) and temporal (*e.g.*, clip reversal, speed changes, etc.) augmentations to both foreground and background videos. Motion augmentations applied to image data also serve to synthesize video sequences from images, making it possible to fine-tune with higher-quality image data for details.

Augmentations for Given Mask. Since our setting is to receive the segmentation mask for the first frame and make alpha matte prediction for all the frames including the first one, it is important to have our model robust to the given mask. To generate the given mask in the training pipeline, we first obtain the original given mask. For segmentation data, it is just the ground truth (GT) for the first frame, while for matting data, it is the binarization result on the first-frame GT alpha matte, with a threshold of 50. *Erosion* or *dilation* is then applied with a probability of 40% each, with kernel sizes ranging from 1 to 5. In this way, we force the model to learn alpha predictions based on an inaccurate segmentation mask, also enhancing the model robustness towards memory readout if it is not so accurate during the predictions in following frames.

Augmentations for Assigned Object(s). The assignment of target object(s) as a segmentation mask for the first frame gives us flexibility for *instance video matting*. Given the strong prior, the model is still easy to be confused by other salient humans not assigned as target. To solve this, we find that a small modification in the video segmentation data pipeline has an obvious effect. In YouTubeVIS [18], for each video with human existence, suppose the number of human instances is H. Instead of combining all of them as one object (practice in previous auxiliary-free methods [13]), we randomly take $h \leq H$ instance as foreground, while unchosen instances are marked as background. In this way, we force the model to distinguish the target human object(s) even when other salient human object(s) exist, enhancing the robustness in object tracking for instance video matting even without instance mask for each frame as MaGGIe [8] has.

B.3. Loss Functions

Given that we take the first-frame segmentation mask alongside with input frames as input, our model needs to predict alpha matte starting from the first frame, which is different from VOS methods [1, 3]. In addition, since we also apply mask augmentation on the given segmentation mask, the prediction from the segmentation head should also start from the first frame. As a result, we need to apply losses on all $t \in [0, N]$ frames for both matting and segmentation heads.

There are mainly three kinds of losses involved in our training: (1) matting loss $\mathcal{L}^{mat}$; (2) segmentation loss $\mathcal{L}^{seg}$; (3) core supervision (CS) loss $\mathcal{L}^{cs}$, and their usages in different training stages are summarized in Table A.

Matting Loss. For frame t, suppose we have the predicted alpha matte M_t w.r.t. its ground-truth (GT) M_t^{GT}. We follow RVM [13] to employ L1 loss for semantics $\mathcal{L}_{l1}$, pyramid Laplacian loss [6] for matting details $\mathcal{L}_{lap}$, and temporal coherence loss [16] $\mathcal{L}_{tc}$ for flickering reduction:

$$\mathcal{L}_{l1} = \|M_t - M_t^{GT}\|_1, \tag{3}$$

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

$$\mathcal{L}_{tc} = \| \frac{\mathrm{d}M_t}{\mathrm{d}t} - \frac{\mathrm{d}M_t^{GT}}{\mathrm{d}t} \|_2, \tag{5}$$

The overall matting loss is summarized as:

$$\mathcal{L}^{mat} = \mathcal{L}_{l1} + 5\mathcal{L}_{lap} + \mathcal{L}_{tc}. \tag{6}$$

Segmentation Loss. For frame t, suppose we have the predicted segmentation mask S_t w.r.t. its ground-truth (GT) S_t^{GT} from the segmentation head. We employ common losses used in VOS [1, 3, 19], $\mathcal{L}_{ce}$ and $\mathcal{L}_{dice}$.

$$\mathcal{L}_{ce} = S_t^{GT}(-log(S_t)) + (1 - S_t^{GT})(-log(1 - S_t)), \tag{7}$$

$$\mathcal{L}_{dice} = 1 - \frac{2S_t S_t^{GT} + 1}{S_t + S_t^{GT} + 1}. \tag{8}$$

The overall segmentation loss is summarized as:

$$\mathcal{L}^{seg} = \mathcal{L}_{ce} + \mathcal{L}_{dice}. \tag{9}$$

Core Supervision Loss. For core-area supervision, we combine the region-specific losses: $\mathcal{L}_{core}$ for core region and $\mathcal{L}_{boundary}$ for boundary region as defined in **??** in the manuscript, and the overall core supervision loss is summarized as:

$$\mathcal{L}^{cs} = \mathcal{L}_{core} + 1.5\mathcal{L}_{boundary}. \tag{10}$$

C. Dataset

Table B. **Comparison on Datasets.** We compare our new training data and testing data with the old ones, in terms of the number of distinct foregrounds, sources, and whether harmonization is applied.

Datesets	VideoMatte240K (old train) [12]	**VM800** (new train)	VideoMatte (old test) [12]	**YouTubeMatte** (new test)
#Foregrounds	475	826	5	32
Sources	-	Storyblocks, Envato Elements, Motion Array	-	YouTube
Harmonized	-	-	x	✓

C.1. New Training Dataset - VM800

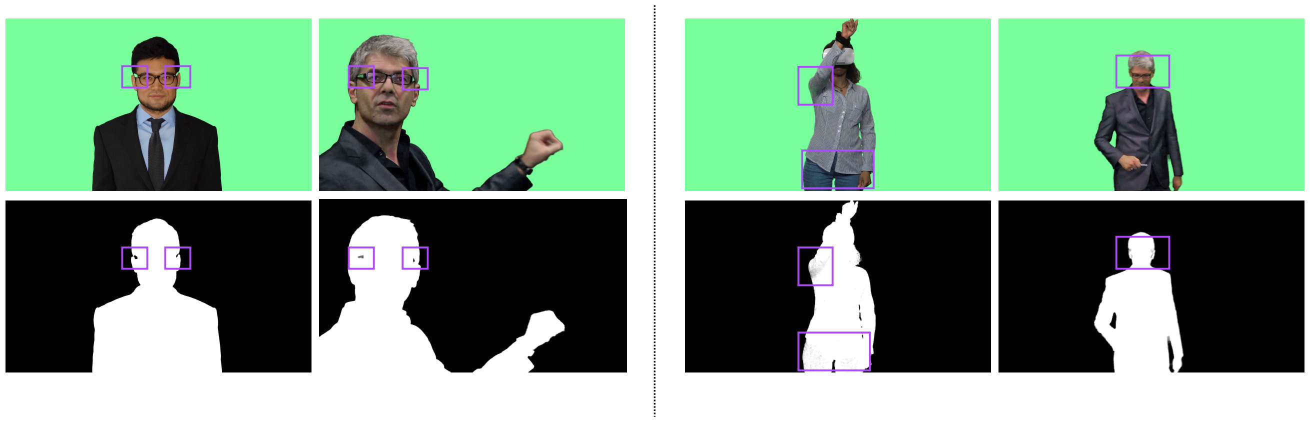
Overview. As summarized in Table B, our new training dataset *VM800* has almost **twice** the number of foreground videos than VideoMatte240K [12] in quantity. To enhance diversity and data distribution, our foreground green screen videos are downloaded from a total of *three* video footage websites: Storyblocks, Envato Elements, and Motion Array, and thus enjoy a diversity in hairstyles, outfits, and motion. In addition, we ensure the high quality of our VM800 dataset in fine detail and through careful manual selection.

Generation Pipeline. We employ Adobe After Effects in our data generation pipeline to extract alpha channels from green screen footage videos. Since the amount of green screen footage to be processed is huge, we would like to obtain the preliminary results with an automatic pipeline. We first use `Keylight` and set `Screen Color` to be the pixel value taken from the upper left corner for each frame. To obtain a clean alpha matte, we clip the values smaller than 20 to be 0 and those larger than 80 to be 255. To further enhance the alpha matte quality, we post-process with another two keying effects `Key Cleaner` and `Advanced Spill Supressor`, which are generally used together following `Keylight`. Since we are processing a video, we also turn on `reduce chatter` in `Key Cleaner` to reduce flickering in the boundary region. For batch processing, we compile the above process into a Javascript and XML file for After Effects to run with, and obtain a large batch of preliminary results for manual selection.

```
Keylight
    - Screen Color: pixel value of upper left corner
    - Screen Matte:
           - Clip Black: 20
           - Clip White: 80

Key Cleaner
    - radius: 1
    - reduce chatter: check

Advanced Spill Supressor
```



(a) Errors in reflective regions (e.g., glasses)

(b) Inhomogeneous in core regions (e.g., shadow)

Figure A. **Issues with VideoMatte240K [12].** (a) Errors in alpha values exist in reflective regions (*e.g.*, "a hole" on glasses). (b) Inhomogeneous alpha values exist in core regions (*e.g.*, caused by shadow), where the alpha value should be exactly 0 or 1.

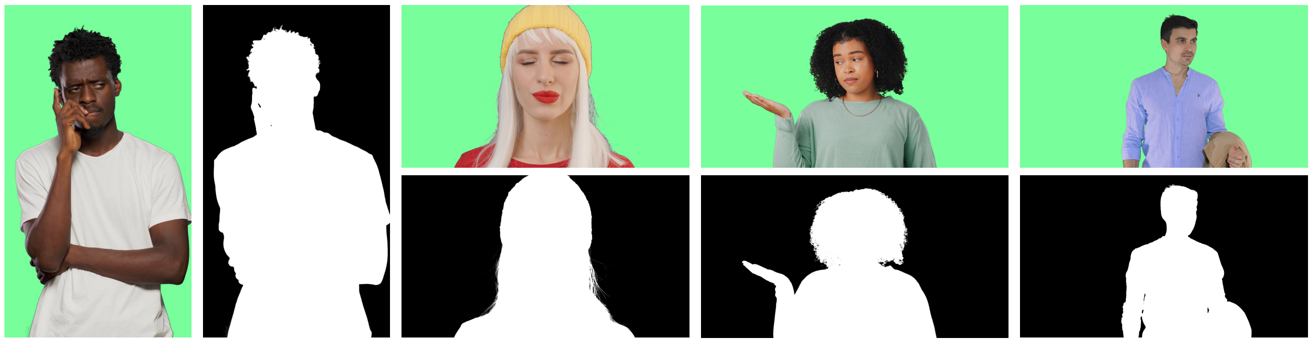


Figure B. **Gallery for our new training dataset VM800.** High-quality details in the boundary regions and diversity in terms of gender, hairstyles, and aspect ratios could be clearly observed.

Quality - Fine Details. The green screen foreground videos we downloaded are almost in a **4K** quality, and we also place a higher priority on those videos with more details (*e.g.*, hair) in our download choice. Fig. B shows the fine details in our VM800 dataset.

Quality - Careful Manual Selection. We notice that alpha mattes extracted with After Effects from green screen videos often encounter inhomogeneities in core regions. For example, reflective regions in the foreground will result in a near-zero value (*i.e.*, a hole) in the alpha matte, as shown in Fig. A(a). In addition, noise also exists in the green screen background, resulting in the fact the alpha values may not homogeneously equal 0, which should not be the case in the core region. Similarly, for foregrounds, colors that are similar to the background green, or shadow in the foreground, may also result in the alpha values not homogeneously equal to 1 in the core foreground region, making the alpha matte look noisy, as shown in Fig. A(b). Since VideoMatte240K [12] is also obtained with After Effects, we observe that alpha mattes with the above problems still exist, and thus taking such wrong ground truth for training will inevitably lead to problematic inference results (Fig. D(a)). As a result, we conduct careful manual selection to examine all our processed alpha mattes, and leave out those with the above problems. As shown in Fig. D(a), training with our VM800 will not lead to such problematic results.

C.2. New Test Dataset - YouTubeMatte

Overview. As summarized in Table B, our new synthetic benchmark *YouTubeMatte* has over **six times** larger than the number of distinct foreground videos in VideoMatte [12], making it a much more representative benchmark for evaluation with better diversity. In addition, the green screen videos for foregrounds are downloaded from YouTube at a scale of 1920×1080 with rich boundary details, thus enhancing its ability to discern matting precision in boundary regions. While the generation pipeline for YouTubeMatte is almost the same as that for VM800, **harmonization** [9], however, is applied when compositing the foreground on a background. Such an operation effectively makes YouTubeMatte a more challenging benchmark that is closer to the real distribution. As shown in Fig. C, while RVM [13] is confused by the harmonized frame, our method still yields robust performance.

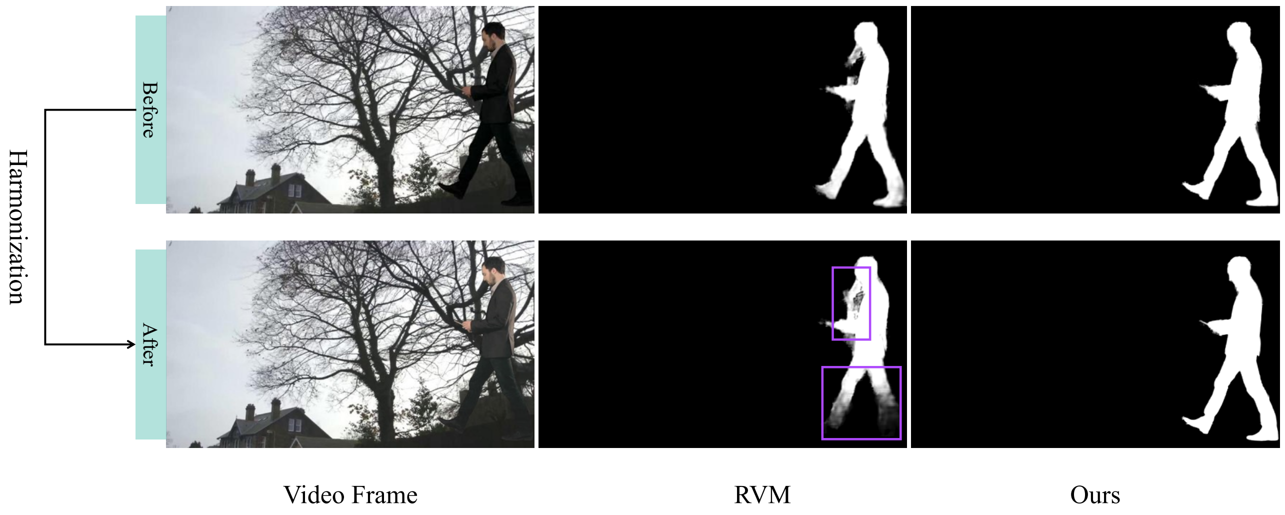


Figure C. **Harmonization on synthetic benchmarks and its effect on model performance.** Harmonization [9] is an operation that makes the composited frame more natural and realistic, which also effectively makes our YouTubeMatte a more challenging benchmark that is closer to the real distribution. It is observed that while RVM [13] is confused by the harmonized frame, our method still yields robust performance.

C.3. Real Benchmark and Evaluation

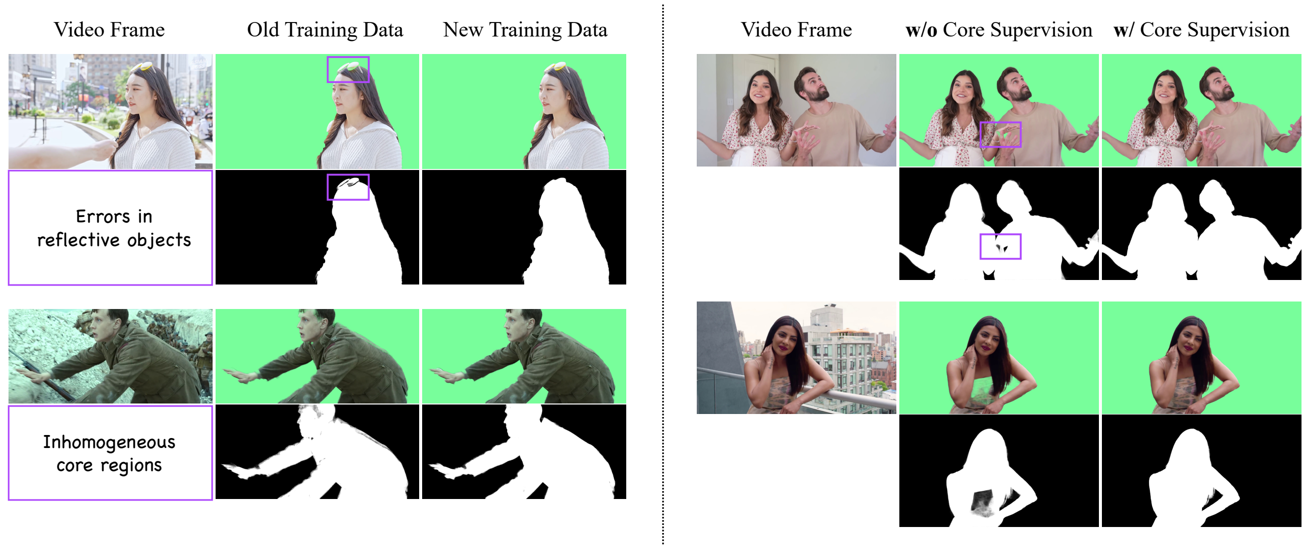
Overview. As a technique towards real-world applications (*e.g.*, virtual background in the online meeting), the synthetic benchmark is not enough to test the generalizability of video matting models. Although there are countless of real human videos for testing in the wild, the lack of GT alpha mattes makes them hard to serve as a real benchmark. Here, we select a subset of 25 real-world videos from [13], where a consecutive of 100 frames for each video are selected with no scene transition, to form our real benchmark. According to our definitions in Fig. 2(a) in the manuscript, we could also divide the evaluation metrics for core regions and for boundary separately, making evaluation for real benchmarks feasible.

Evaluation on Core Regions. Thanks to the recent success of VOS methods [1, 3], frame-wise segmentation masks could be generated with high precision. Here, we employ Cutie [3] for video segmentation results. We first obtain the trimap for each segmentation mask by applying dilation and erosion (with kernel size 21), and then compute the core mask where trimap values equal 0 or 1. In this way, the values of a segmentation mask within its core region could be considered as the GT alpha values for the core region, where common metrics including MAD and MSE for semantic accuracy, and dtSSD [4] for temporal coherency could be applied for evaluation.

D. More Results

D.1. Enhancement from New Training Data

As discussed in Sec. 4.1 in the manuscript and Section C.1 in the supplementary, our new training data VM800 is upgraded in quantity, quality, and diversity. In addition to the quantitative evaluation in Table 3 in the manuscript, we further show the enhancement from new training data by providing more results when comparing the model trained with VideoMatte240K [12] and the model trained with our VM800 in Fig. D(a).



(a) Enhancement from New Training Data

(b) Effectiveness of New Training Scheme

Figure D. **(a) Comparison on results trained with old training data (VideoMatte240K [12]) and new training data (our VM800).** It could be observed that training with old data will lead to errors in reflective objects (*e.g.*, holes on the sunglasses) and inhomogeneous alpha values in the core regions. However, both issues are fixed when training with our new data, indicating a higher quality. **(b) Comparison on results trained without and with core-area supervision.** It could be observed that training without it will lead to semantics error due to the weak supervision from real segmentation data, while training with core supervision largely improves semantics accuracy thanks to the stronger supervision enabled.

D.2. Effectiveness of Consistent Memory Propagation

As one of our key designs, the consistent memory propagation (CMP) module improves both stability in core regions and quality in boundary details. In addition to the quantitative evaluation in Table 3 in the manuscript, we give more qualitative results and analysis in Fig. E.

D.3. Effectiveness of New Training Scheme

Our new training scheme introduces core-area supervision, which largely enhances the semantic accuracy and stability, as shown in Table 3 in the manuscript. More qualitative results are shown in Fig. D(b) for better visualization of its effects.

D.4. Effectiveness of Recurrent Refinement

As discussed in Sec. 3.3 in the manuscript, the sequential prediction in the memory-based paradigm enables recurrent refinement without the need for retraining during inference. By repeating the first frame n times and iteratively updating the first frame prediction based on the last-time prediction, the quality of the first frame alpha matte could be recurrently refined. We show in Fig. F that such recurrent refinement can not only (1) enhance the robustness to the given segmentation mask even when it is of low quality, but also (2) achieve matting details at an image-matting level when compared with an image matting method (*i.e.*, Matte Anything [20] in the last column).

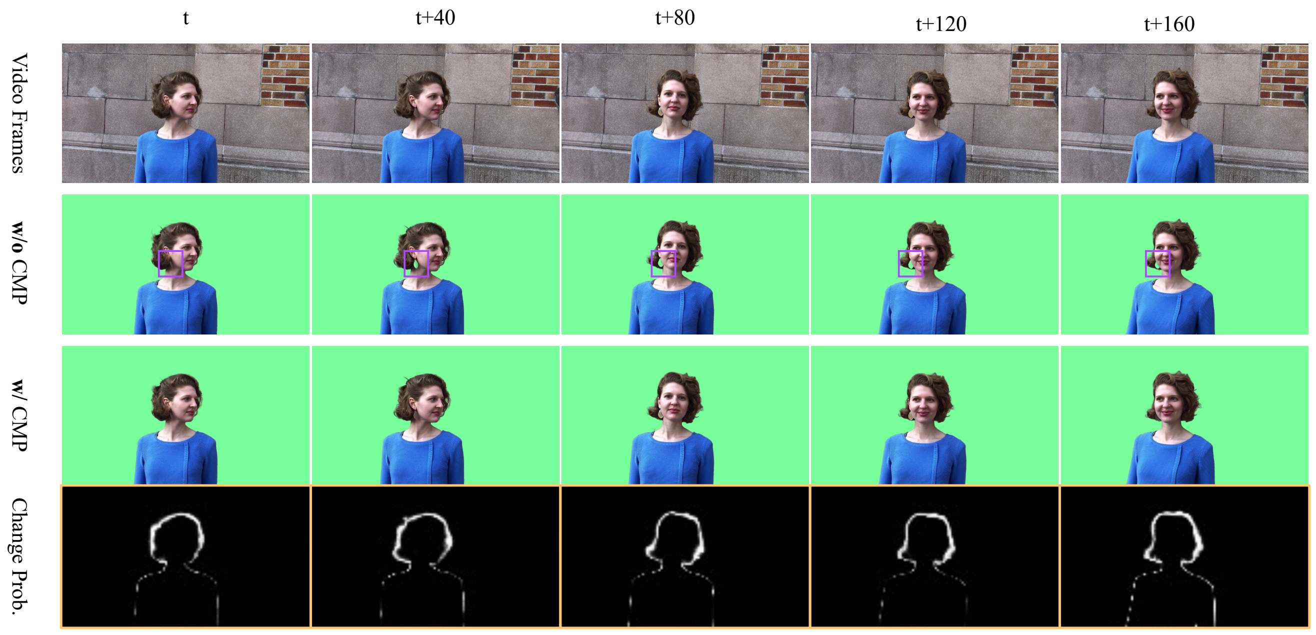


Figure E. **Comparison on results with and without Consistent Memory Propagation.** It could be observed that when CMP is not applied, semantic errors constantly exist across a wide span of video frames. However, when training with CMP, we observe from the "Change Probability" mask that usually our model only takes pixels near the boundary as "changed", and most of the inner regions (*i.e.*, earring) will mainly take the memory values from the last frame. As we can see on the figure, while predictions are both correct at time t, the model with CMP successfully keeps the correctness and gives stable results, while the model without CMP quickly breaks the correctness and never recovers.

D.5. More Qualitative Comparisons

In this subsection, we provide additional visual comparisons of our method with the state-of-the-art methods, including auxiliary-free (AF) method: RVM [13] and mask-guided methods: FTP-VM [7], and MaGGIe [8]. Fig. G presents the general video matting results on real videos. To further demonstrate the superiority of our model, Fig. H and Fig. I both showcase a challenging case respectively, where other methods mostly fail. In addition, Fig. J demonstrates the instance matting results compared with MaGGIe [8], a method with instance mask for *each* frame is given as guidance, while our model only has the segmentation mask for the *first* frame as guidance.

D.6. Demo Video

We also offer a demo video. This video showcases more video matting results and a hugging face demo for applicability, both on real-world videos.

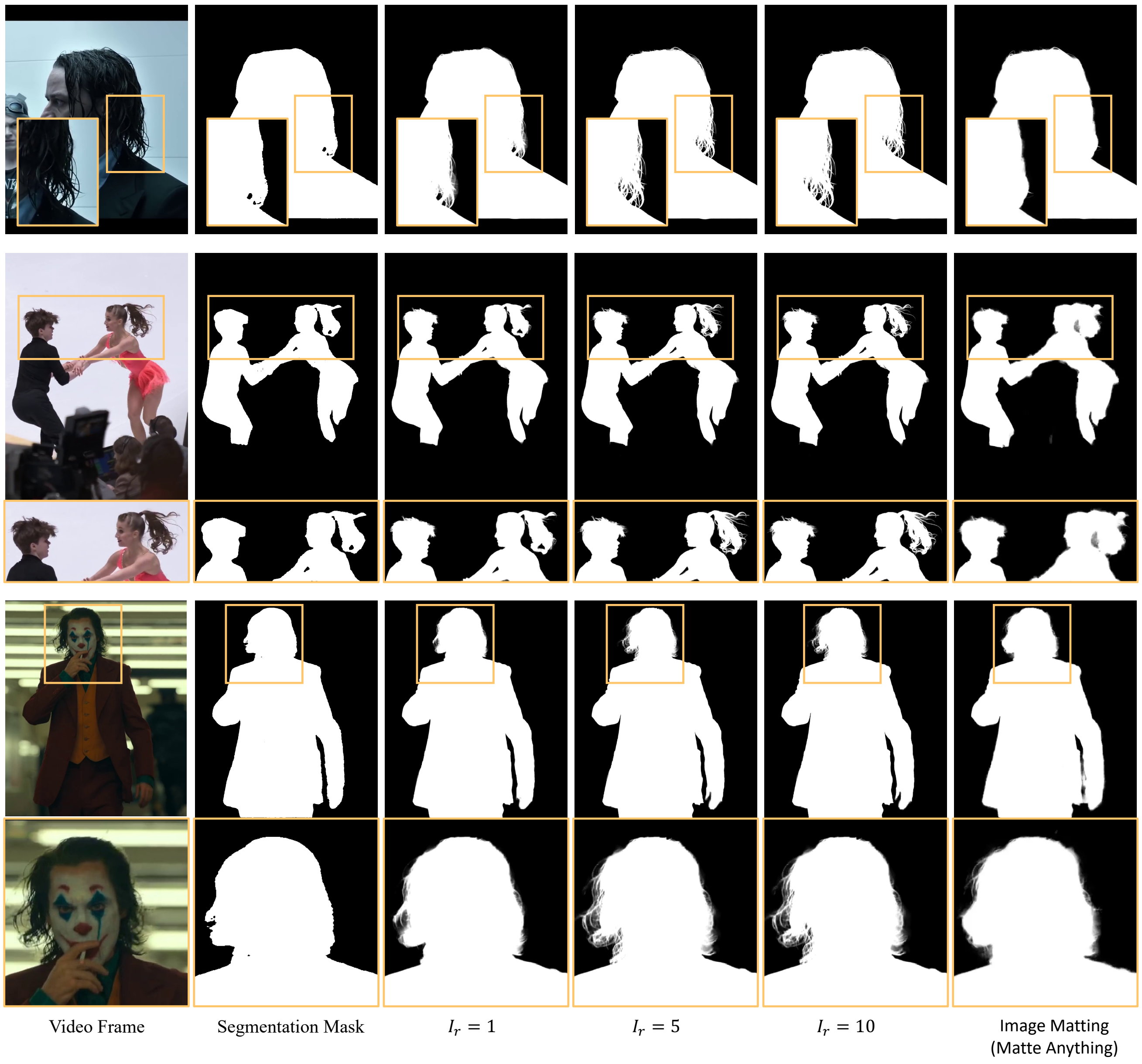


Figure F. **Comparison on results with iterative refinement.** A noticeable enhancement on details can be observed even with one iteration of refinement compared with the given segmentation mask. Within 10 iterations, our model is able to achieve matting details at an image-matting level, even better than Matte Anything [20], which is an image matting model also based on the results from SAM [10].

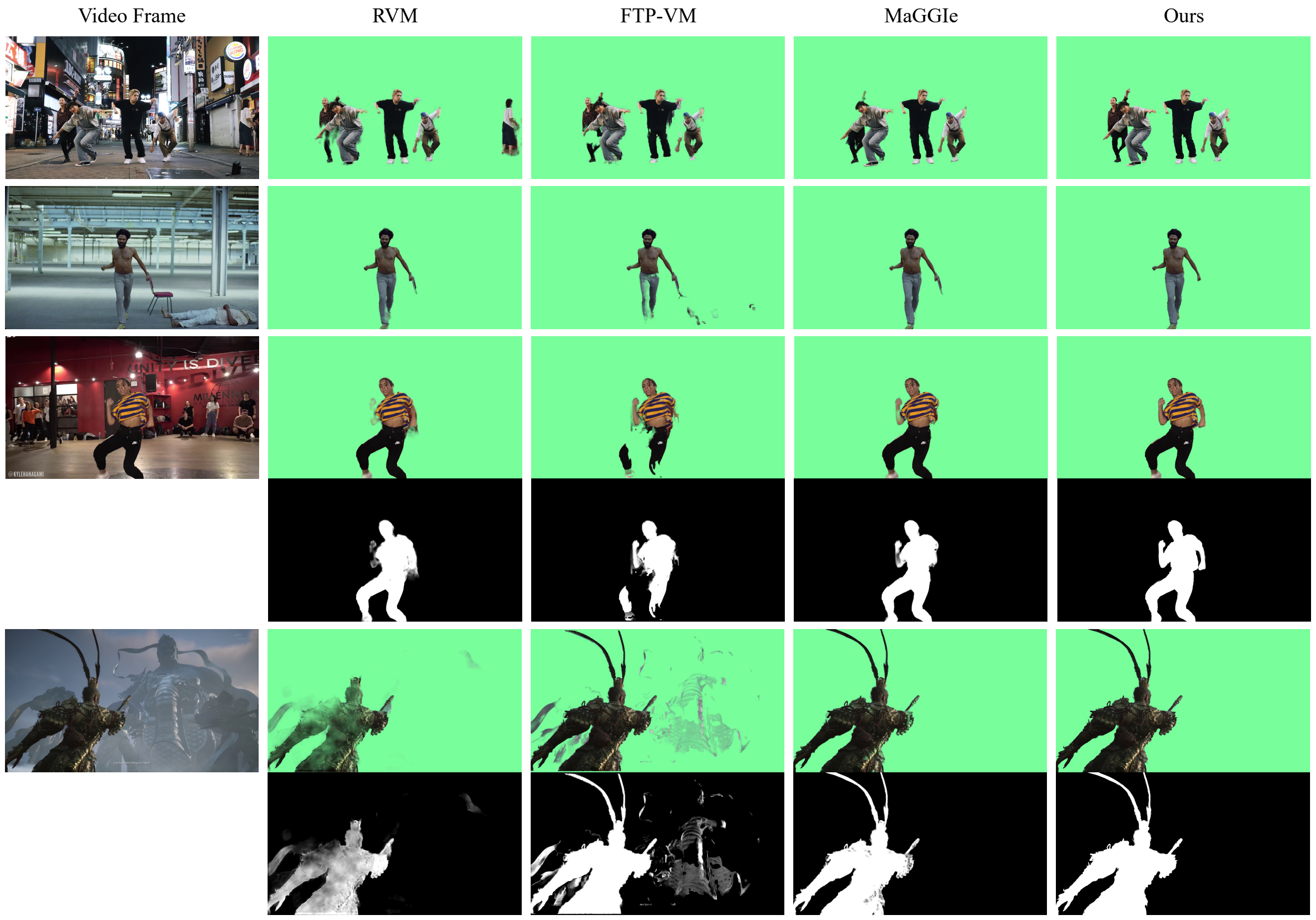


Figure G. **More qualitative comparisons on general video matting with SOTA methods.** We compare our MatAnyone with both auxiliary-free (AF) method: RVM [13] and mask-guided methods: FTP-VM [7], and MaGGIe [8]. It could be observed that our method significantly outperforms others in both detail extraction and semantic accuracy, across diverse and complex real scenarios. It is noteworthy that although sometimes MaGGIe [8] seems to give acceptable results when compositing with a green screen, its alpha matte turns out to be noisy (*i.e.*, inhomogeneous in the core foreground region and blurry in the boundary region), while our alpha matte is clean with fine-grained details in the boundary region. As a result, we also include alpha mattes for a more comprehensive comparison. **(Zoom in for best view)**

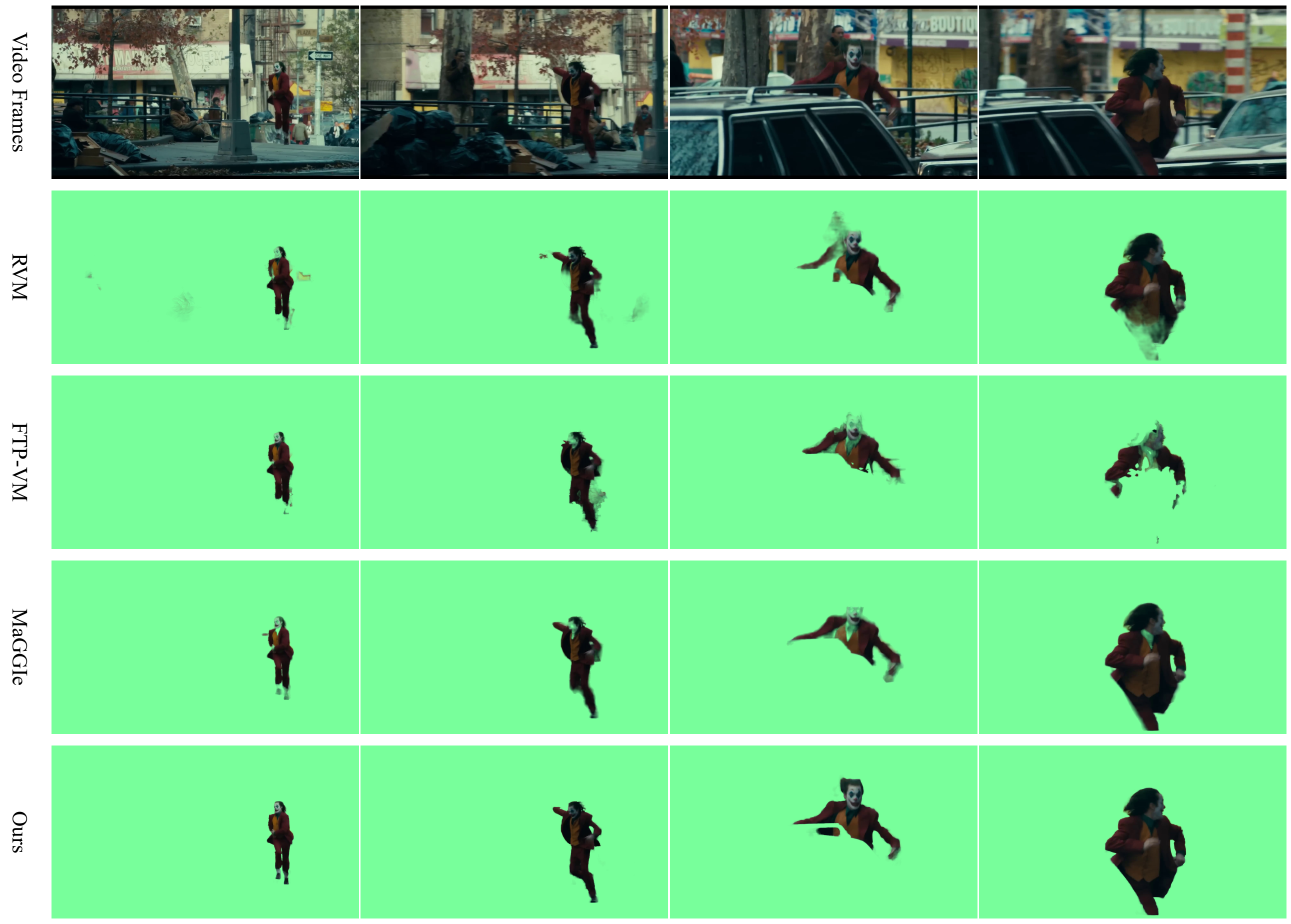


Figure H. **A challenging example of general video matting across a long time span.** We compare our MatAnyone with both auxiliary-free (AF) method: RVM [13] and mask-guided methods: FTP-VM [7], and MaGGIe [8]. It could be observed that our model is able to track the target object stably even when the object is moving fast in a highly complex scene, where all the other methods present noticeable failures. **(Zoom in for best view)**

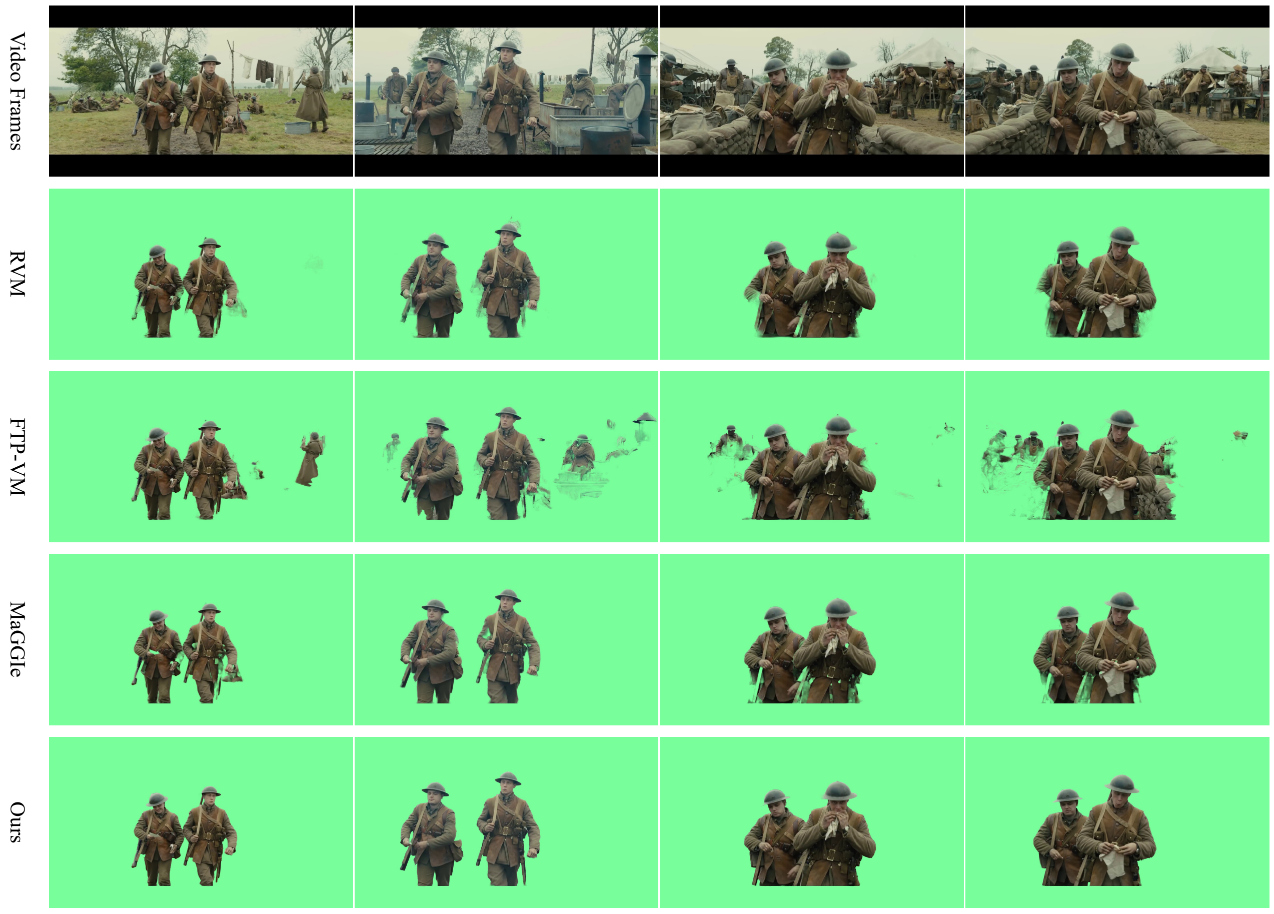


Figure I. **Another challenging example of general video matting across a long time span.** We compare our MatAnyone with both auxiliary-free (AF) method: RVM [13] and mask-guided methods: FTP-VM [7], and MaGGIe [8]. This example showcases that our model is able to track the target objects even in a highly ambiguous background, where the colors for foreground and background are similar, and also multiple humans in the background. In addition, it also demonstrates when there is more than one target object, our model is still able to handle this challenging case well. **(Zoom in for best view)**

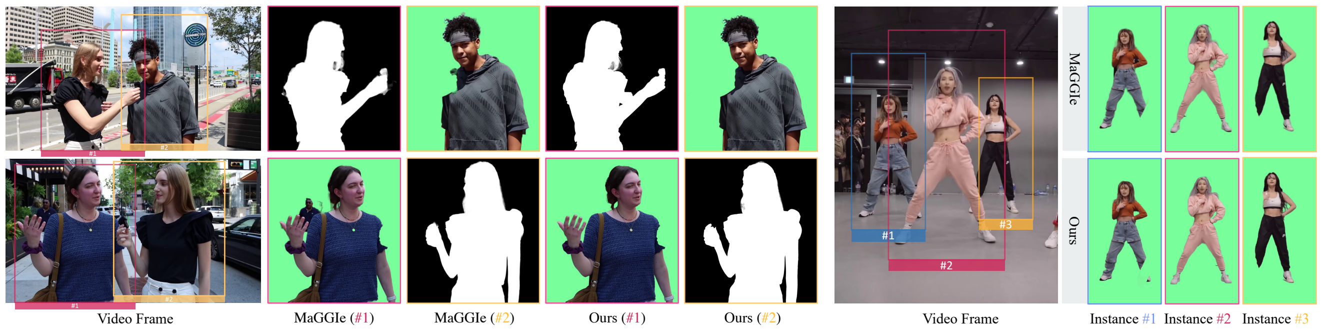


Figure J. **More qualitative comparisons on instance matting.** We compare our MatAnyone with MaGGIe [8], a mask-guided method that requires the instance mask for *each* frame, while our method only requires the mask for the *first* frame. It could be observed that even with such strong given prior, MaGGIe still performs below our method in terms of semantic accuracy in the core regions. Moreover, in terms of the boundary regions, by examining the details there, we could clearly observe that the details generated by MaGGIe are blurry and far from fine-grained compared with our results. **(Zoom in for best view)**

References

[1] Ho Kei Cheng and Alexander G. Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 2, 3, 4, 6

[2] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *NeurIPs*, 2021.

[3] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *CVPR*, 2024. 2, 3, 4, 6

[4] Mikhail Erofeev, Yury Gitman, Dmitriy S Vatolin, Alexey Fedorov, and Jue Wang. Perceptually motivated benchmark for video matting. In *BMVC*, 2015. 7

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[6] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. In *ICCV*, 2019. 3

[7] Wei-Lun Huang and Ming-Sui Lee. End-to-end video matting with trimap propagation. In *CVPR*, 2023. 3, 8, 10, 11, 12

[8] Chuong Huynh, Seoung Wug Oh, , Abhinav Shrivastava, and Joon-Young Lee. MaGGIe: Masked guided gradual human instance matting. In *CVPR*, 2024. 3, 8, 10, 11, 12

[9] Zhanghan Ke, Chunyi Sun, Lei Zhu, Ke Xu, and Rynson WH Lau. Harmonizer: Learning to perform white-box image and video harmonization. In *ECCV*, 2022. 6

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 9

[11] Chung-Ching Lin, Jiang Wang, Kun Luo, Kevin Lin, Linjie Li, Lijuan Wang, and Zicheng Liu. Adaptive human matting for dynamic videos. In *CVPR*, 2023. 3

[12] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *CVPR*, 2021. 2, 4, 5, 6, 7

[13] Shanchuan Lin, Linjie Yang, Imran Saleemi, and Soumyadip Sengupta. Robust high-resolution video matting with temporal guidance. In *WACV*, 2022. 3, 6, 8, 10, 11, 12

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 3

[15] I Loshchilov. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[16] Yanan Sun, Guanzhi Wang, Qiao Gu, Chi-Keung Tang, and Yu-Wing Tai. Deep video matting via spatio-temporal alignment and aggregation. In *CVPR*, 2021. 3

[17] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. In *ICCV*, 2017. 3

[18] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019. 3

[19] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 4

[20] Jingfeng Yao, Xinggang Wang, Lang Ye, and Wenyu Liu. Matte Anything: Interactive natural image matting with segment anything model. *Image and Vision Computing*, page 105067, 2024. 7, 9