Fast Real-Time Video Object Segmentation with a Tangled Memory Network

JIANBIAO MEI, MENGMENG WANG, and YU YANG, Zhejiang University
YANJUN LI, Zhejiang University City College
YONG LIU, Zhejiang University

In this article, we present a fast real-time tangled memory network that segments the objects effectively and efficiently for semi-supervised video object segmentation (VOS). We propose a tangled reference encoder and a memory bank organization mechanism based on a state estimator to fully utilize the mask features and alleviate memory overhead and computational burden brought by the unlimited memory bank used in many memory-based methods. First, the tangled memory network exploits the mask features that uncover abundant object information like edges and contours but are not fully explored in existing methods. Specifically, a tangled two-stream reference encoder is designed to extract and fuse the features from both RGB frames and the predicted masks. Second, to indicate the quality of the predicted mask and feedback the online prediction state for organizing the memory bank, we devise a target state estimator to learn the *IoU* score between the predicted mask and ground truth. Moreover, to accelerate the forward process and avoid memory overflow, we use a memory bank of fixed size to store historical features by designing a new efficient memory bank organization mechanism based on the mask state score provided by the state estimator. We conduct comprehensive experiments on the public benchmarks DAVIS and YouTube-VOS, demonstrating that our method obtains competitive results while running at high speed (66 FPS on the DAVIS16-val set).

CCS Concepts: • **Computing methodologies** → **Scene understanding**; **Video segmentation**; **Tracking;**

Additional Key Words and Phrases: Tangled memory network, two stream, state estimator, memory organization mechanism

ACM Reference format:
Jianbiao Mei, Mengmeng Wang, Yu Yang, Yanjun Li, and Yong Liu. 2023. Fast Real-Time Video Object Segmentation with a Tangled Memory Network. *ACM Trans. Intell. Syst. Technol.* 14, 3, Article 51 (April 2023), 21 pages.
https://doi.org/10.1145/3585076

J. Mei and M. Wang contributed equally to this research.

This work was supported by a Grant from The National Natural Science Foundation of China (No. U21A20484).

Authors' addresses: J. Mei, M. Wang, Y. Yang, and Y. Liu (corresponding author), Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China, 310027; emails: jianbiaomei@zju.edu.cn, mengmengwang@zju.edu.cn, yu.yang@zju.edu.cn, yongliu@iipc.zju.edu.cn; Y. Li, School of Information and Electrical Engineering, Zhejiang University City College, Hangzhou, China; email: liyanjun@zucc.edu.cn.

2157-6904/2023/04-ART51 $15.00
https://doi.org/10.1145/3585076

1 INTRODUCTION

Video Object Segmentation (VOS) is one of the most basic tasks in the computer vision community. There are many potential applications, such as video editing and autonomous driving. Generally, VOS contains unsupervised [16, 37, 45, 71], weakly supervised [14, 59, 66], semi-supervised [9, 13, 31], and interactive [4, 11] tasks. The unsupervised task, also named *zero-shot VOS* [30, 35, 72], predicts the object masks for a video without any prior knowledge. The weakly supervised task only uses weakly supervised labels such as points and scratches, whereas the interactive task involves human interactions. Recently, referring VOS [20, 60, 61], which segments the target object referred by a language expression, has also attracted the attention of many researchers. We focus on the semi-supervised task, which predicts target objects' segmentation masks over video sequences with only an initial mask. One of the most challenging problems in this task is how to learn a robust appearance representation of the target, as it may undergo all kinds of variations in videos like deformations, occlusions, motion blur, and so on, meanwhile running in real time.

Existing methods [5, 9, 18, 26, 28, 40–42, 51, 57, 58, 62, 67, 68] model the appearance representations mainly from the input RGB frames and regard masks as simple auxiliary. They pay less attention to further mining the edge features and object characteristics in the object's masks. Thus, their usages of masks are still not sufficient. For example, MaskTrack [42], RGMP [40], and many STM-based methods [26, 33, 41, 62] simply concatenate the frame and target mask as the input of the network. AGAME [18], FEELVOS [51], and CFBI [67] take target mask prediction as another channel of features or mainly treat the target mask as guidance for separating foreground and background pixels. Some methods, such as GIEL [9] and SITVOS [23] use another label encoder to extract high-level mask features and enhance the target appearance, but they neglect the low-level mask features, and the interactions between frame features and mask features are simple. We believe they do not dig deep into the mask features and miss many edge cues, which could benefit the segmentation quality. Different from these approaches, we aim to enable sufficient interactions between frame features and mask features and exploit the mask reconstruction to facilitate the learning of the mask stream for better utilization of object characteristics, including contours/edges features.

Our insight is that the mask features have ideal properties to aid efficient VOS: the mask separates the foreground and background much more clearly than RGB frames and, most importantly, contains rich contour/edge information. We designed a **Tangled Memory Network (TMN)** to delve into the mask features effectively and efficiently. Instead of using only one reference encoder to treat the RGB frame and mask equally, our TMN leverages a two-stream reference encoder to embed better and fuse two sources of information: RGB frame features and mask features. To enhance the interactions between the RGB and mask features, we inject features from the mask stream into the frame stream and reconstruct the input mask to learn informative cues in the mask features. With bi-modal interaction, TMN can select judicious cues for mask prediction. Additionally, as shown in Figure 1, our speed is much faster than these methods, which often are several orders of magnitude slower.

As well, due to the limited memory and inference speed requirements in practical applications, it is impossible to store features of all the past frames and masks in the memory, especially when processing long video sequences. Therefore, an efficient memory bank organization mechanism is imperative. Several methods are aware of this problem, but their solutions are always complicated or inefficient. For instance, EGMN [33] models the memory as a fully connected graph with a slow speed of 5 FPS on the DAVIS17-val set. AFB-URR [27] introduces a large adaptive feature bank to dynamically absorb new features and discard obsolete features, which is similar to our strategy. However, it runs only at a speed of 4 FPS on the DAVIS17-val set, since their memory

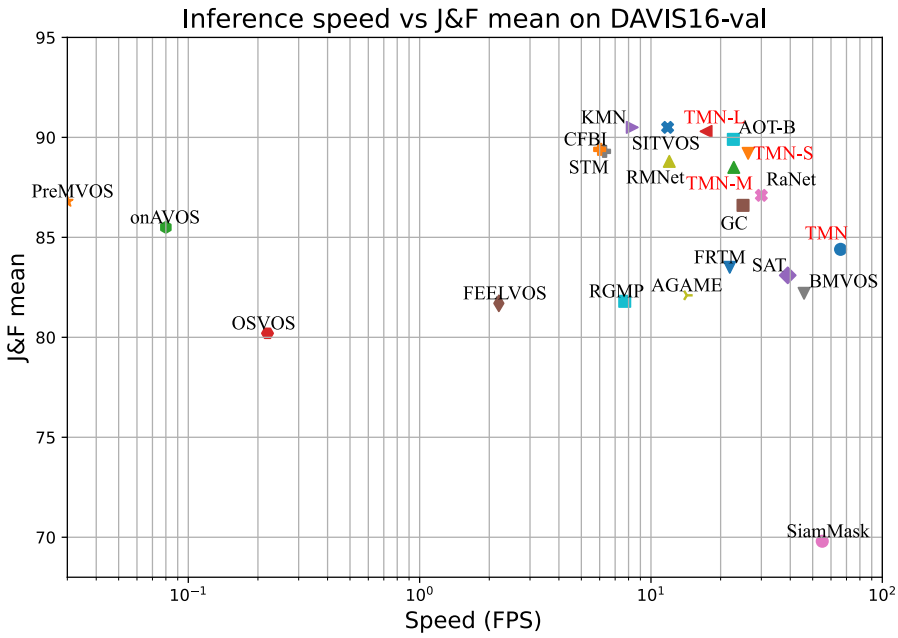

Fig. 1. The speed comparison on the DAVIS16-val set. $\mathcal{J}\&\mathcal{F}$ mean vs speed (480p resolution). Note that TMN is tested with half resolution (240p).

size is still large and their absorbing and removing strategy is complex. In this article, we devise a simple but effective memory bank organization mechanism to fix the memory size and avoid issues like memory overflow and limited speed. Generally, high-quality segmentation results of historical frames always contain more accurate target information and are conducive to subsequent frames' segmentation. In contrast, low-quality segmentation results will mislead subsequent mask prediction. Nevertheless, online VOS always lacks feedback about the output quality. Therefore, we first design a **Target State Estimator (TSE)** to acquire an indicator about the segmentation results by predicting the *IoU* between the mask prediction and its corresponding ground truth. Unlike the quality assessment module in the work of Liu et al. [32], which uses another encoder to take both the current frame and segmentation mask as input for acquiring more information, our TSE is more compact and reuses the features of past frames. Moreover, our TSE can serve as a state indicator to facilitate the learning of the reference encoder. Then, based on the predicted *IoU* score, we devise a simple but effective memory maintaining strategy to adaptively determine whether to put the features into the memory, keeping a fixed and dynamically updated memory bank.

We implement several variants of our method. They are different in the input resolution and backbone. As demonstrated in Figure 1, TMN achieves competitive accuracy and runs faster than all other approaches. Additionally, TMN-S achieves the balance between accuracy and speed. The main contributions of this article are summarized as follows:

- We propose a fast real-time TMN to excavate the rich target information like contour and edge features contained in the mask, which is not fully exploited by existing methods. Several variants with different model sizes are implemented to adapt to different application platforms easily.
- We design a TSE to predict an *IoU* score, providing reliable mask prediction feedback during online inference. Moreover, based on this score, a simple memory bank organization mechanism is devised to maintain the memory bank efficiently.

- The effectiveness of our method is verified on three VOS benchmarks, including DAVIS2016/2017 [43, 44] and YouTube-VOS [63]. Our method achieves competitive results while running in highly real time.

2 RELATED WORK

Our work focuses on mask feature utilization and memory bank organization strategy in real-time semi-supervised VOS. Therefore, we mainly review and discuss recent semi-supervised VOS from the following three perspectives: memory-augmented VOS, mask utilization in VOS, and real-time VOS.

Memory-Augmented VOS. Recently, STM-based methods [5, 9, 22, 23, 26, 33, 39, 41, 46, 54, 70] have achieved state-of-the-art performance. Different from LSTM-based methods [50, 56, 63], they exploit the memory bank to store the past frames' information for robust target appearance. STM [41] first proposes a space-time memory structure for VOS, which puts features from historical frames into memory. After that, many memory-based methods [9, 22, 22, 25, 33, 39, 54, 62] were built on STM. For example, GC [25] develops a global context module representing a fixed-size feature to encode all past frames and use constant memory regardless of the video length. EGMN [33] proposes an episodic graph memory network and takes the memory as a fully connected graph to model visual relationships from correlated video frames [36]. MAST [22] combines the space-time memory network with self-supervised learning, which randomly discards and reconstructs a channel of Lab color space during training. AFB-URR [27] proposes an adaptive feature bank to absorb new features and discard obsolete features dynamically. FRTM [46] designs a memory bank to store previous RGB frame features and masks for fine-tuning target model parameters online. SwiftNet [54] elaborately compresses spatiotemporal redundancy in matching-based VOS via Pixel-Adaptive Memory. RMNet [62] proposes to replace STM's global-to-global matching with local-to-local matching. We also regard STM as the baseline, but our efficient memory maintaining is different from these methods in two folds: (1) our memory bank is dynamically updated while the size is small (3 in TMN) and fixed, and (2) the strategy of updating it is based on a predicted *IoU* score.

Mask Utilization in VOS. As the initial mask is provided in semi-supervised VOS, many methods [12, 27, 28, 33, 40–42, 47, 62] concatenated the raw mask with the RGB frame as the input of the feature extractor to encode target information. For instance, RGMP [40] and AGSS [28] used a four-channel Siamese feature extractor to encode the previous/current frame and the predicted/previous segmentation masks. STM-based methods [12, 37, 41, 47, 62] exploited a separate four-channel memory encoder for the reference frames with the predicted masks. Some other methods [8, 18, 51, 58, 62, 67] aggregated the raw mask or mask features with frame features for foreground discovery or target appearance enhancement. AGAME [18] concatenated the masks with the frame features to enhance the target appearances. The masks were furthermore used to initialize the appearance model. FEELVOS [51] and CFBI [67] mainly adopted the mask to separate the foreground and background and perform the nearest neighbor matching between the current frame and the first and previous frames in the feature space. RMNet [62] exploited the mask to provide the foreground region for regional memory reading. GIEL [9] and SITVOS [23] used another label encoder to extract high-level mask features and fused the high-level mask features and frame features to enhance the target appearance. SSTVOS [8] used the mask to get object affinity value from the transformer for predicting object masks. Unlike these approaches, we aim to enable sufficient interactions between frame features and mask features and exploit the mask reconstruction to facilitate the learning of the mask stream for better utilization of object characteristics, including contours/edges features.

Real-Time VOS. For efficiency, many real-time methods [3, 15, 49, 53, 55] integrate object tracking techniques to indicate target location and spatial area to avoid full-frame segmentation,

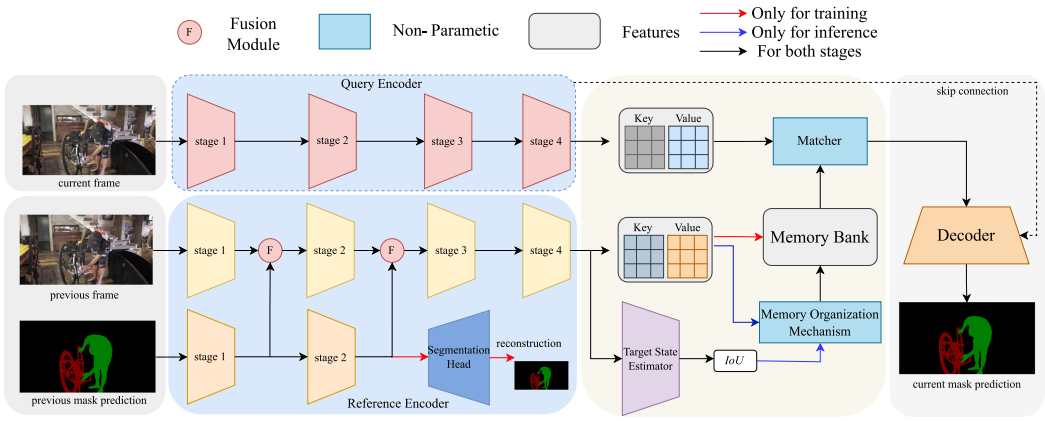

Fig. 2. Overview of our proposed TMN. The query encoder takes the query frame as input. The reference encoder consists of the mask stream and frame stream. The mask stream takes the object mask prediction ($softmax$ output) as input and reconstructs the predicted mask. The frame stream takes an RGB frame as input and fuses features from the mask stream. The TSE predicts the IoU between the predicted mask and ground truth to estimate the mask state. The memory organization mechanism stores or updates features from the previous frame in the memory based on the IoU to fix the memory bank size. The matcher retrieves information from the memory bank. The decoder takes matched features and features from the query encoder as the input and outputs the current frame's segmentation result.

combining the two problems of Video Object Segmentation and Tracking (VOST) [69]. SiamMask [55] and SAT [3] respectively combine SiamRPN [24] branch and SiamFC++ [65] to exploit the relationship between frame pairs [34] and estimate the current object bounding box as the reference of the search area in the next frame. FTAN-DTM [15] takes object segmentation as a sub-task of tracking, introducing the "tracking-by-detection" model into VOS. The integration of a tracker helps improve the inference speed, whereas tracking accuracy often limits these methods' performance. Some approaches adopt the single-frame reference strategy [18, 40, 58] or adaptive memory bank [27, 54] to reduce the memory size and improve the inference efficiency. However, their efficiency improvement is always limited. STM-cycle [26] and BMVOS [6] respectively use a small input resolution and light backbone to improve running speed, but their performance drops a lot. Our TMN also uses a small resolution and light backbone to improve efficiency, but our proposed tangled reference encoder effectively excavates the rich target information like contour and edge features contained in the reference masks, boosting performance significantly. Moreover, our dynamic memory bank with a fixed size reduces the memory size and can further improve efficiency. TMN achieves not only high-quality segmentation but also has fast inference speed (66 FPS). We also implement several variants with different model sizes to adapt to different application platforms easily.

3 APPROACH

The semi-supervised VOS problem can be simply defined as follows: given a video sequence of length T and a target mask in the first frame, the task is to segment sequential frames at every timestamp t according to a reference sets $\{(\mathbf{I}_i, \mathbf{M}_i)|i \in S, S \subseteq [1, t-1], 2 \leq t \leq T\}$, where $\mathbf{I}_i$ denotes the i-th frame and $\mathbf{M}_i$ denotes the corresponding object mask (ground truth when $i = 1$).

In this work, we design a novel TMN to efficiently predict accurate segmentation masks of a target object in videos. The overview of our framework is illustrated in Figure 2. It mainly consists of a query encoder, a reference encoder, a TSE, a matcher, a memory bank, and a decoder. When

segmenting a specific frame $\mathbf{I}_t$, the query encoder extracts the features of $\mathbf{I}_t$, and the reference encoder embeds $\mathbf{I}_{t-1}$ and $\mathbf{M}_{t-1}$. Based on the features of $\mathbf{I}_t$, the matcher retrieves information from the memory bank, which stores features of past frames and masks. Then, the decoder takes matched features and low-level features from the query encoder as the input and outputs the segmentation result $\mathbf{M}_t$. Our method is built upon a common baseline STM [41]. Since our core contributions lie on the mask utilization (Section 3.1), the TSE (Section 3.2), and the memory bank organization mechanism (Section 3.3), we mainly discuss them in the following. We also present the training and inference procedure of the proposed TMN (Section 3.4). The query encoder, matcher, and decoder are kept consistent with STM. We explain them in Section 3.4 briefly; please find more details in the work of Oh et al. [41].

3.1 Mask Utilization

The ways of using the mask in existing methods [18, 28, 28, 40–42, 51, 58, 67] are not sufficient. They all do not fully exploit the edge features and object characteristics inside the object masks, which can help address the distraction problems and obtain sharper segmentation results. We design a two-stream reference encoder, including a classical frame stream and a mask stream, to excavate the edge and object information in the object mask. As shown in Figure 2, the frame stream employs the four stages of ResNet [10], whereas the mask stream only uses the first two stages of ResNet to encode the input object mask. The reasons are in two aspects: (1) the low-level mask features from the shallow layers contain fine-grained features such as edges and contours, which are just what we need, and (2) the target area information in the low-level mask features can make the foreground and background of RGB frames more contrastive. Moreover, inspired by CFBI [67], we believe the background of the predicted masks (after the $softmax$ operation) contains spatial information about similar targets in the environment, which can help handle distraction problems. Therefore, both the foreground and background of the object mask are concatenated and fed into the mask stream. Features of the mask stream after every used stage are injected into the frame stream through an AFC module [48]. We visualize several mask feature maps from stage 1 in Figure 3 (line 2), illustrating that the low-level features indeed learn the edge information and target area information. However, some details shown in red rectangles may remain coarse and need to be refined.

We believe that if a mask can be recovered well, its encoder should keep all the informative features. Thus, to make the mask stream learn the target information more accurately, we further design a **Segmentation Head (SH)** to reconstruct the input object mask. Inspired by DeepLab v2 [2], our SH consists of the atrous spatial pyramid pooling module and a convolution layer. Note that the SH only exists during training and brings no extra computational burdens for inference. We combine a cross-entropy loss $\mathcal{L}_{CE}$ and a mask IoU loss $\mathcal{L}_{IoU}$ as the reconstruction loss $\mathcal{L}_r$. The losses can be formulated as follows:

$$\mathcal{L}_{CE}(\mathbf{Y}_t, \mathbf{M}_{r,t}) = \frac{1}{|\Omega|} \sum_{u \in \Omega} [\mathbf{Y}_t^u \log(\mathbf{M}_{r,t}^u) + (1 - \mathbf{Y}_t^u) \log(1 - \mathbf{M}_{r,t}^u)], \tag{1}$$

$$\mathcal{L}_{IoU}(\mathbf{Y}_t, \mathbf{M}_{r,t}) = 1 - \frac{\sum_{u \in \Omega} \min(\mathbf{Y}_t^u, \mathbf{M}_{r,t}^u)}{\sum_{u \in \Omega} \max(\mathbf{Y}_t^u, \mathbf{M}_{r,t}^u)}, \tag{2}$$

$$\mathcal{L}_r(\mathbf{Y}_t, \mathbf{M}_{r,t}) = \mathcal{L}_{CE}(\mathbf{Y}_t, \mathbf{M}_{r,t}) + \alpha \cdot \mathcal{L}_{IoU}(\mathbf{Y}_t, \mathbf{M}_{r,t}), \tag{3}$$

where Ω denotes the set of all pixels in the object mask, and $\mathbf{Y}_t, \mathbf{M}_{r,t}$ denote ground truth and the reconstructed mask of the t-th frame, respectively. We showcase the feature maps from the mask

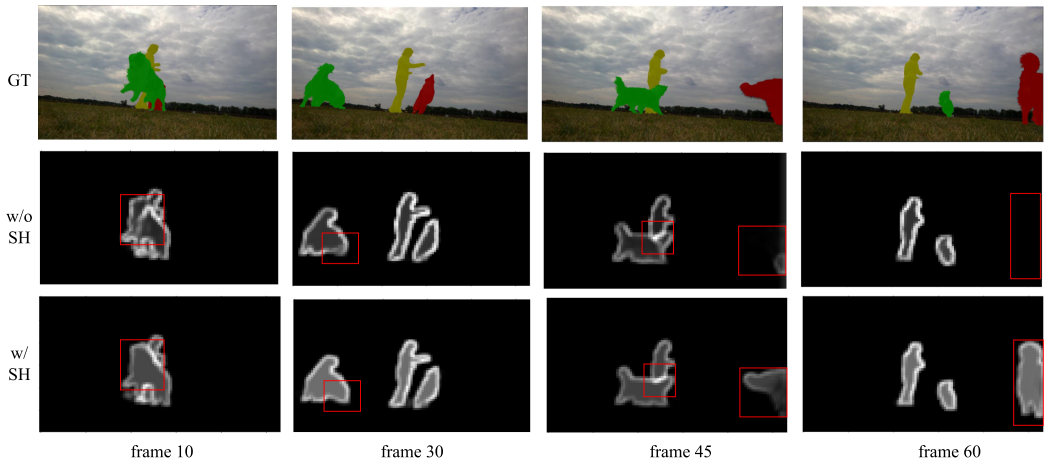

Fig. 3. Visualization and comparison of feature maps from stage 1 of the mask stream without/with an SH on the video (dogs-jump) in the DAVIS17-val set. The SH helps improve the accuracy of the boundary area, especially when the target object is occluded.

stream with an SH in line 3 of Figure 3. Compared to line 2 (without an SH), it shows sharper and more accurate edges of the target object.

3.2 Target State Estimator

Generally, there is no feedback or indicator to present the quality of the predicted mask during the online inference phase. However, low-quality object masks always mislead the reference encoder to focus on the wrong regions and contours. Then, the extracted features may pollute the memory bank, resulting in poor segmentation results. Additionally, we do not want to use the time-consuming online learning technique to refine the predicted mask-like method [26], which dramatically slows down the running speed. Thus, we need a state score to indicate the predicted mask quality. Based on the state score, the features from previous frames and masks can be dynamically merged into the memory bank.

To address the preceding problems, inspired by the work of Jiang et al. [17], we design a TSE to predict the *IoU* between the predicted mask and its corresponding ground truth as the mask state score $\hat{s}$, when memorizing target appearances from the previous frame and its predicted mask in the reference encoder. As shown in Figure 2, the TSE is behind the reference encoder, consisting of two 3×3 convolutions, a global average pooling layer, a 1×1 convolution, and a *sigmoid* operation. We take the output features of stage 4 from the frame stream as the input to the TSE. These features combine the semantic information from the frame stream and the edge information from the mask stream, which is useful for estimating the mask state. We take MSE loss as the *IoU* prediction loss $\mathcal{L}_e$:

$$\mathcal{L}_e(\mathbf{Y}_t, \mathbf{M}_t, \hat{s}_t) = \frac{1}{2} \cdot (\hat{s}_t - \mathcal{S}_{IoU}(\mathbf{Y}_t, \mathbf{M}_t))^2, \tag{4}$$

$$\mathcal{S}_{IoU}(\mathbf{Y}_t, \mathbf{M}_t)) = \frac{\sum_{u \in \Omega} \min(\mathbf{Y}_t^u, \mathbf{M}_t^u)}{\sum_{u \in \Omega} \max(\mathbf{Y}_t^u, \mathbf{M}_t^u)}, \tag{5}$$

where $\mathcal{S}_{IoU}$ is a function to calculate the *IoU* between the predicted mask $\mathbf{M}_t$ and ground truth $\mathbf{Y}_t$ of the t-th frame. $\hat{s}_t$ is the state score of the predicted mask $\mathbf{M}_t$. Ω denotes the set of all pixels in the object mask.

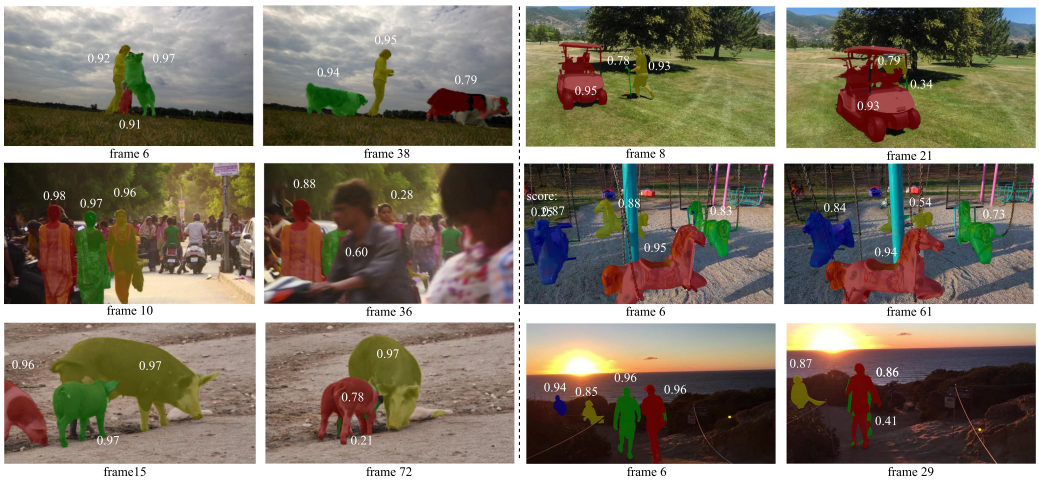

Fig. 4. Mask state score predicted by the TSE on videos (dogs-jump, India, pigs) in the DAVIS17-val set and (golf, carousel, people-sunset) in DAVIS17 test-dev. The state score reflects the quality of the segmentation result.

In Figure 4, we show the predicted mask and give the corresponding state score predicted by the TSE, demonstrating that our target estimator could indicate the confidence of the prediction.

3.3 Memory Bank Maintainment

As the video clip's length grows, the reference set will become larger and larger, making it hard to store features from all the past frames and masks in the memory in long sequences. Since the device storage is limited, the matching process will be slow. Moreover, not all memory frames are necessary for segmenting the current frame since the features of consecutive frames are similar and historical frames with the poor prediction may pollute the memory bank. In this article, we propose to fix the memory bank size and filter the unnecessary memory features with the mask state score predicted by the proposed state estimator.

We fix the memory bank size to M, which means that only M pairs of key and value maps are saved in the memory. The first frame with the ground truth mask provides the most reliable information, thus we save features of the initial frame and mask by default. As Figure 5 shows, our memory bank organization mechanism can be expressed as follows: given the memory bank initialized by the features of the first frame $MB = \{(k_1, v_1, s_1)\}$, we first temporally enqueue the features of the previous frame $(k_{t-1}, v_{t-1}, s_{t-1})$ for current segmentation, where k_*, v_*, s_* denote the key maps, value maps, and mask state score, and the subscript denotes the timestamp. If $|MB| < M$, the new features will be stored in the memory every K frame directly, since the state score is always large in the beginning. Otherwise, the features (k_m, v_m, s_m) with the minimum mask state score in the memory are replaced with the new features (k_n, v_n, s_n) (if $s_n > s_{th}$), where s_{th} is a threshold for determining whether to drop the new features.

3.4 Training and Inference

Training. We use DAVIS2017 [44], YouTube-VOS [63], and COCO [29] to train our models end to end. The backbone is pretrained on ImageNet. During training, we sample clips from training video datasets or randomly sample images from COCO and perform affine transformations to form the video clips. When sampling clips from video datasets, we randomly skip frames, and the maximum

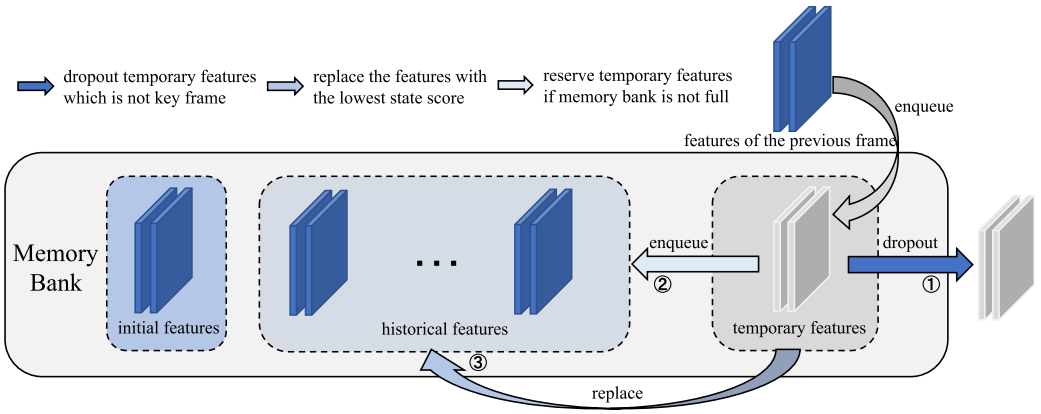

Fig. 5. Our proposed memory bank maintaining strategy. The initial frame features are stored in the memory bank by default. The features of the previous frame are enqueued in the memory bank as temporary features when segmenting the current frame. After that, the temporary features will be reserved, drop out, or used to replace the old features with the lowest state score.

skip is gradually increased during training to simulate the target appearance change over a long time. We recurrently train our network: at the beginning of training, the first frame and its ground truth are embedded into the key and value feature maps by the reference encoder and put into the memory bank to segment the second frame, then the second frame and its predicted mask are encoded and put into the memory bank to segment the third frame, and the remaining frames go on as in the preceding. The segmentation loss $\mathcal{L}_s$ has the same form as the reconstruct loss $\mathcal{L}_r$, which can be formulated as follows:

$$\mathcal{L}_s(\mathbf{Y}_t, \mathbf{M}_t) = \mathcal{L}_{CE}(\mathbf{Y}_t, \mathbf{M}_t) + \alpha \cdot \mathcal{L}_{IoU}(\mathbf{Y}_t, \mathbf{M}_t), \tag{6}$$

where $\mathbf{Y}_t, \mathbf{M}_t$ denote ground truth and the predicted mask of the t-th frame, respectively.

We train our TMN by combining the losses (Equations (3), (4), and (6)) together. In particular, we use the uncertainty loss proposed by Kendall et al. [19] to automatically balance losses. The total training loss can be formulated as follows:

$$\begin{aligned}\mathcal{L}_{total} = \frac{1}{(T-1)} \cdot \sum_{2 \le t \le T} \frac{1}{2} \{ & e^{-w_1} \cdot \mathcal{L}_s(\mathbf{Y}_t, \mathbf{M}_t) \\ & + e^{-w_2} \cdot [\mathcal{L}_r(\mathbf{Y}_{t-1}, \mathbf{M}_{r,t-1}) \\ & + \mathcal{L}_e(\mathbf{Y}_{t-1}, \mathbf{M}_{t-1}, \hat{s}_{t-1})] + w_1 + w_2 \},\end{aligned} \tag{7}$$

where $\mathbf{Y}_t, \mathbf{M}_t$ denote ground truth and the predicted mask of the t-th frame, respectively. $\mathbf{Y}_{t-1}, \mathbf{M}_{t-1}, \mathbf{M}_{r,t-1}$ denote ground truth, the predicted mask, and the reconstructed mask of the previous frame, respectively. $\hat{s}_{t-1}$ denotes the state score of $\mathbf{M}_{t-1}$. w_1 and w_2 are learnable parameters, and T is the length of a training video clip.

For multi-object segmentation, we run our model for each object independently. Then, the predicted masks for each object are merged by a soft aggregation operation used in the work of Oh et al. [40, 41].

Inference. In the online testing phase, our network uses past frames and predictions to segment the current frame. When segmenting the t-th frame, the query encoder, which uses the same backbone as the frame stream, takes the current frame as input, then outputs the key and value feature maps (k_t^Q, v_t^Q); the reference encoder takes the previous frame and corresponding predicted mask

as input and outputs the key and value feature maps (k^R_{t-1}, v^R_{t-1}); and the TSE outputs the mask state score $\hat{s}_{t-1}$. We use the memory bank organization mechanism proposed in Section 3.3 to store and update features $(k^R_{t-1}, v^R_{t-1}, \hat{s}_{t-1})$. After an attention-based matching operation as in the work of Oh et al. [41], the decoder takes the matched features and features from the query encoder as the input and outputs the current frame's mask $\mathbf{M}_t$. Similar to the training procedure, we run our model for each object independently for multi-object segmentation, then a soft aggregation operation is used to merge all predicted masks. Moreover, we choose the object with the maximum probability as the predicted object for every pixel in the merged predicted mask.

4 EXPERIMENTS

4.1 Implementation Details

We use DAVIS2017 [44], YouTube-VOS [63], and COCO [29] to train our model. We take ResNet34 [10] pretrained on ImageNet [7] as the backbone of our TMN. The input frames are resized into 240×427 for training. Additionally, the length T of the training video clip is set to 3. We follow Li et al. [26] to set α to 1. During training, the maximum frame skip increases by 5 every 20 training epochs, and we freeze all batch normalization layers and use the Adam [21] optimizer with a fixed learning rate of 1e-5 and $\beta = (0.9, 0.999)$ to minimize the total loss. For all experiments, the network is trained with batch size of 8 for 130 epochs, and it takes about 2 days to train our network with four NVIDIA TITAN RTX GPUs. In inference, we set $K = 5, M = 3, s_{th} = 0.85$ in all experiments unless specified. As well, we test our TMN on a single NVIDIA TITAN RTX GPU in all experiments. We implement several versions, namely TMN, TMN-S (small version), TMN-M (median version), and TMN-L (large version). Note that we do not pretrain TMN on COCO. Additionally, the difference between TMN and other versions is that the latter use different backbones, are pretrained on COCO, and take larger resolution (400×400) as the input during the main training. As well, at the inference stage, TMN takes half resolution (240×427) while other versions use full resolution (480×854).

4.2 Comparison with the State of the Art

We compare our approach with state-of-the-art methods on DAVIS [43, 44] and YouTube-VOS [63] benchmarks. The quantitative results demonstrate that our method obtains competitive results while running in real time. The evaluation metrics are provided by the DAVIS benchmark. $\mathcal{J}\&\mathcal{F}$ evaluates the general quality of the segmentation results, J evaluates the mask IoU, and F estimates the quality of contours.

Qualitative Results. The visualization results on DAVIS17-val are shown in Figure 6. SAT [3] and SiamMask [55] have real-time inference speed as shown in Table 1, and they are easy to collapse when the targets are occluded as shown in Figure 6. TMN not only has a high inference speed (66 FPS) but also is robust to object appearance changing and object occlusion. We also show visualizations on DAVIS17 and YouTube-VOS18 validation in Figure 7 and Figure 8, from which we can see that TMN performs well when segmenting fast-moving, highly similar, and occluded objects. However, TMN fails to segment some rare tiny objects (stick and golf), especially when they are severely occluded. Tiny object segmenting is also hard for most methods, and we do not make special designs for that.

DAVIS. Both DAVIS2016 and DAVIS2017 are evaluated. DAVIS2016 is a single-object annotated dataset containing 30 training video sequences and 20 validation video sequences. DAVIS2017 is a multi-objects expanded from DAVIS2016, containing 60 training video sequences and 30 validation video sequences. We report the results on the DAVIS16-val set in Table 1 and on the DAVIS17-val set in Table 2. Most methods employ ResNet50 or deeper backbones for higher performance, thus

Fig. 6. Qualitative comparisons on DAVIS2017. Compared to SAT [3] and SiamMask [55], TMN is more robust to the target appearance changing and handles object occlusion better.

Table 1. Comparison with the State of the Art on the DAVIS16 Validation Set

Method	Backbone	PT	OL	DAVIS16-val			
				FPS	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)
SAT(+YT) [3]	ResNet50			39	83.1	82.6	83.6
FEELVOS(+YT) [51]	DeepLabv3+			2.2	81.7	81.1	82.2
SiamMask(+YT) [55]	ResNet50			55	69.8	71.7	67.8
BMVOS (+YT) [6]	DenseNet-121			45.9	82.2	82.9	81.4
TMN(+YT)	ResNet34			**66.0**	**84.4**	**84.0**	**84.9**
FRTM(+YT) [46]	ResNet101		✓	21.9	83.5	–	–
PReMVOS [38]	ResNet101		✓	0.03	86.8	84.9	88.6
OnAVOS [52]	ResNet		✓	0.08	85.5	86.1	84.9
OSVOS [1]	VGG		✓	0.22	80.2	79.8	80.6
KMN(+YT) [47]	ResNet50	✓		8.3	90.5	89.5	91.50
GC [25]	ResNet50	✓		25	86.6	87.6	85.7
RaNet [58]	ResNet101	✓		30	85.5	85.5	85.4
AGAME(+YT) [18]	ResNet101	✓		14.3	82.1	82.0	82.2
RGMP [40]	ResNet50	✓		7.7	81.8	81.5	82.0
AOT-B(+YT) [68]	ResNet50	✓		22.7	89.9	88.8	90.9
RMNet(+YT) [62]	ResNet50	✓		12	88.8	88.9	88.7
RPCM(+YT) [64]	ResNet50	✓		–	90.6	87.1	94.0
SITVOS(+YT) [23]	ResNet50	✓		11.8	90.5	89.5	91.4
STM(+YT) [41]	ResNet50	✓		6.3	89.3	88.7	89.9
TMN-S(+YT)	ResNet18	✓		**26.3**	89.2	88.3	90.0
TMN-M(+YT)	ResNet34	✓		22.8	88.5	87.5	89.5
TMN-L(+YT)	ResNet50	✓		17.3	90.3	**89.7**	90.9

"OL" indicates the use of an online learning strategy. "PT" denotes pretraining on synthetic videos from static image datasets. "+YT" indicates the use of YouTube-VOS for training. Note that TMN-L, TMN-S, and TMN-M are trained with larger input resolution compared to TMN. Additionally, the runtimes of other methods are copied from corresponding papers.

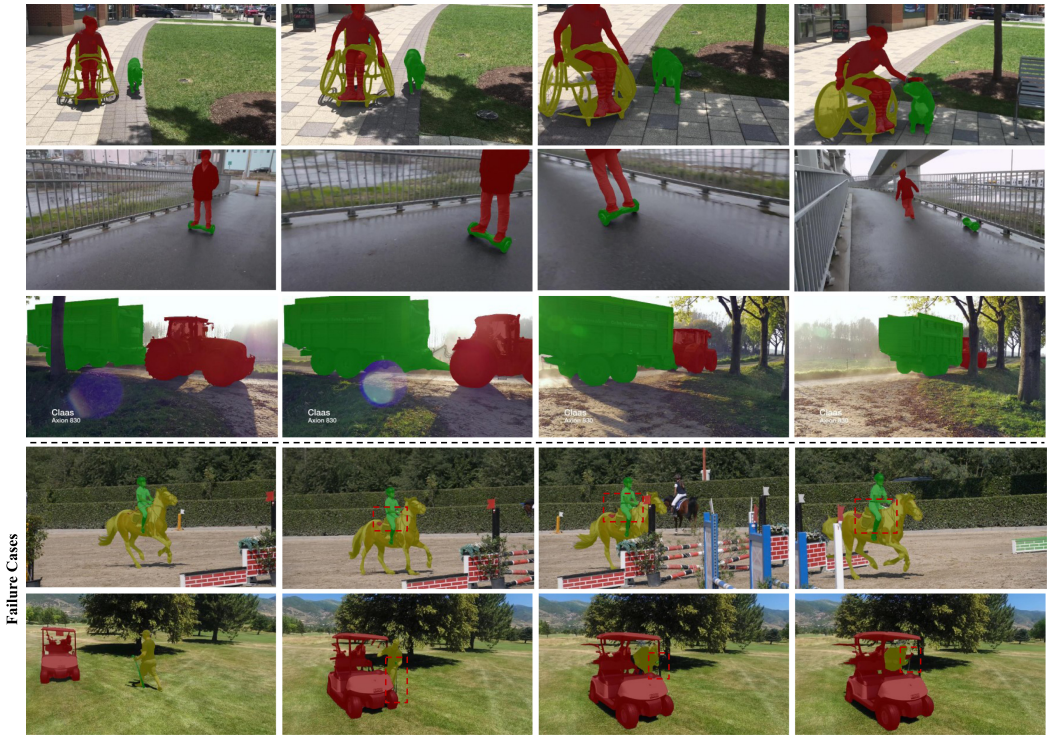

Fig. 7. Qualitative results on DAVIS2017. Visualizations of the first column are the ground truth of the first frame. TMN performs well when segmenting occluded objects and fast-moving objects, but TMN fails to segment some rare tiny objects (stick and golf), especially when they are severely occluded. Tiny object segmenting is also hard for most methods, and we do not make special designs for that.

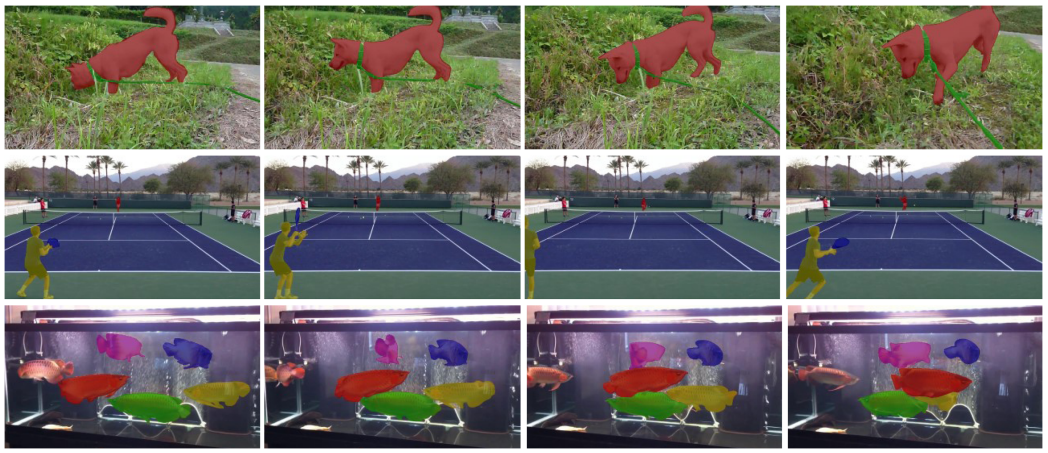

Fig. 8. Qualitative results on YouTube-VOS 2018 validation. Visualizations of the first column are the ground truth of the first frame. TMN performs well when segmenting highly similar objects, occluded objects, and fast-moving objects.

Table 2. Comparison with the State of the Art on the DAVIS17 Validation Set

Method	Backbone	PT	OL	DAVIS17-val		
				$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)
TVOS [70]	ResNet50			72.3	69.7	74.7
SAT(+YT) [3]	ResNet50			72.3	68.6	76.0
FEELVOS(+YT) [51]	DeepLabv3+			71.5	69.1	74.0
SiamMask(+YT) [55]	ResNet50			56.4	54.3	58.5
AGSS [28]	ResNet50			66.6	63.4	69.8
BMVOS [6]	DenseNet-121			72.7	70.7	74.7
TMN(+YT)	ResNet34			**74.3**	**71.5**	**77.0**
STM-cycle(+YT) [26]	ResNet50		✓	72.3	69.3	75.3
FRTM(+YT) [46]	ResNet101		✓	76.7	–	–
PReMVOS [38]	ResNet101		✓	77.8	73.9	81.7
OnAVOS [52]	ResNet		✓	67.9	64.5	71.2
OSVOS [1]	VGG		✓	60.3	56.7	63.9
AFB-URR [27]	ResNet50	✓		74.6	73.0	76.1
EGMN(+YT) [33]	ResNet50	✓		82.8	80.2	85.2
GC [25]	ResNet50	✓		71.4	69.3	73.5
RaNet [58]	ResNet101	✓		65.7	63.2	68.2
AGAME(+YT) [18]	ResNet101	✓		70.0	67.2	72.7
RGMP [40]	ResNet50	✓		66.7	64.8	68.6
SST(+YT) [8]	ResNet50	✓		82.5	79.9	85.1
GIEL(+YT) [9]	ResNet50	✓		82.7	80.2	**85.3**
SwiftNet(+YT) [54]	ResNet50	✓		81.1	78.3	83.9
AOT-B(+YT) [68]	ResNet50	✓		82.1	79.4	84.8
STM(+YT) [41]	ResNet50	✓		81.8	79.2	84.3
TMN-S(+YT)	ResNet18	✓		79.6	77.1	82.1
TMN-M(+YT)	ResNet34	✓		80.7	77.5	84.0
TMN-L(+YT)	ResNet50	✓		82.2	79.2	**85.3**

"OL" indicates the use of an online learning strategy. "PT" denotes pretraining on synthetic videos from static image datasets. "+YT" indicates the use of YouTube-VOS for training. Note that TMN-L, TMN-S, and TMN-M are trained with larger input resolution compared to TMN.

their speed is limited. TMN is our fast version, which uses a small input resolution, lightweight backbone, and a fixed memory bank to accelerate the inference speed for real-time application scenarios such as edge devices. Thanks to our proposed tangled reference encoder, the TSE, and the quality-aware memory organization strategy, TMN's accuracy did not drop a lot. Under the same training strategy (without pretraining on the static image datasets) and inference strategy (without using the online learning strategy), TMN outperforms other methods in both J score and F score and runs the fastest, as shown in the first part of Table 1 and Table 2. This demonstrates that our TMN can fully exploit edge/contours and target area information in the mask features to boost performance. Additionally, even with a small backbone and input resolution, our TMN outperforms many methods like pretraining-based AGAME [18], RGMP [40], online learning based FRTM [46], and OSVOS [1] in Table 1 on the DAVIS16-val set. For instance, although OnAVOS [52] is superior to our TMN with a slight margin of 1.1% in the $J\&F$ score on the DAVIS16-val set, the speed of TMN is more than 811 times than it. Without bells and whistles, TMN also obtains promising performance and outperforms online learning based methods with large margins, such as 2% improvement against STM-cycle [26] (using cyclic training strategy on the baseline

Table 3. Comparison with the State of the Art on the YouTube-VOS 2018 Validation Set

Method	Backbone	PT	OL	YouTube-VOS 2018 val				
				Overall	$\mathcal{J}_s(\%)$	$\mathcal{J}_u(\%)$	$\mathcal{F}_s(\%)$	$\mathcal{F}_u(\%)$
TVOS [70]	ResNet50			67.8	67.1	63.0	69.4	71.6
SAT [3]	ResNet50			63.6	67.1	55.3	70.2	61.7
SiamMask [55]	ResNet50			52.8	60.2	45.1	58.2	47.7
RVOS [50]	ResNet101			56.8	63.6	45.5	67.2	51.0
AGSS [28]	ResNet50			71.3	71.3	65.5	75.2	73.1
S2S [63]	VGG16			64.4	71.0	55.5	70.0	61.2
STM* [41]	ResNet50			68.2	–	–	–	–
TMN	ResNet34			71.0	**74.1**	61.7	**78.2**	69.8
STM-cycle [26]	ResNet50		✓	70.8	72.2	62.8	76.3	71.9
FRTM [46]	ResNet101		✓	72.1	72.3	65.9	76.2	74.1
OnAVOS [52]	ResNet		✓	55.2	60.1	46.6	62.7	51.4
OSVOS [1]	VGG		✓	58.8	59.8	54.2	60.5	60.7
AFB-URR [27]	ResNet50	✓		79.6	78.8	74.1	83.1	82.6
EGMN [33]	ResNet50	✓		80.2	80.7	74.0	85.1	80.9
GC [25]	ResNet50	✓		73.2	72.6	68.9	75.6	75.7
STM [41]	ResNet50	✓		79.4	79.7	72.8	84.2	80.9
AGAME [18]	ResNet101	✓		66.1	67.8	60.8	–	–
RGMP [40]	ResNet50	✓		53.8	59.5	45.2	–	–
SITVOS [23]	ResNet50	✓		81.3	79.9	76.4	84.3	84.4
GIEL [9]	ResNet50	✓		80.6	80.7	75.0	85.0	81.9
SwiftNet [54]	ResNet50	✓		77.8	77.8	72.3	81.8	79.5
STM [41]	ResNet50	✓		79.4	79.7	72.8	84.2	80.9
TMN-S	ResNet18	✓		78.1	79.0	71.3	83.1	79.2
TMN-M	ResNet34	✓		77.4	77.8	71.0	82.2	78.8
TMN-L	ResNet50	✓		80.6	80.2	74.9	84.5	82.8

"OL" indicates the use of an online learning strategy. "PT" denotes pretraining on synthetic videos from static image datasets. STM* denotes STM with main training only. The subscripts of $\mathcal{J}$ and $\mathcal{F}$ on YouTube-VOS denote seen objects (s) and unseen objects (u). The metric overall means the average of $\mathcal{J}_s$, $\mathcal{J}_u$, $\mathcal{F}_s$, $\mathcal{F}_u$. Note that TMN-L, TMN-S, and TMN-M are trained with larger input resolution compared to TMN.

model) on the DAVIS17-val set. Our other versions also achieve comparable performance to state-of-the-art methods on the DAVIS16-val and DAVIS17-val sets. As Table 1 shows, TMN-L has competitive performance ($\mathcal{J}\&\mathcal{F}$ 90.3%) and fast inference speed (17.3 FPS). We noticed that TMN-L achieves the best results on the J score but not the best on the F score. We explain that some methods use specific techniques to improve their performance, such as KMN using the hide-and-seek training strategy, which helps improve the contour accuracy (F score) of segmentation results. Moreover, compared to the baseline STM, our TMN-L can achieve 1% gains in the F score. In Table 2, our TMN-L performs best on the F score and still surpasses the baseline STM by 1% on the F score, which further demonstrates the effectiveness of the utilization of mask features.

YouTube-VOS. The YouTube-VOS dataset is a large-scale dataset in VOS, containing 3,471 training videos and 474 validation videos. Additionally, each video contains a maximum of 12 objects. The validation set includes seen objects from 65 training categories and unseen objects from 26 categories, which is good for evaluating algorithms' generalization performance. We compare our method with other state-of-the-art methods in Table 3. Our fast version TMN outperforms most

Table 4. Speed Comparison Between Our Proposed TMN with Previous Real-Time VOS Methods on the DAVIS2016 Validation Set

Method	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)	FPS
SAT [3]	83.1	82.6	83.6	43.1
FRTM [46]	83.5	–	–	11.6
SiamMask [55]	69.8	71.7	67.8	56.2
STM-cycle [26]	83.8	**84.1**	83.5	48.1
BMVOS [6]	82.2	82.9	81.4	28.3
TMN	**84.4**	84.0	**84.9**	**62.2**

For a fair comparison, all methods are tested five times on the same platform (NVIDIA 1080 Ti), and the results are averaged.

methods in the first part, which are trained without pretraining on static images but slightly lower than AGSS on the overall score (0.3%). We explain that AGSS initialized their IAM module with pretrained weights of pretraining-based RGMP to accelerate convergence, which helps improve the performance on YouTube-VOS. TMN also performs better than most online learning methods except FRTM. Especially TMN surpasses the baseline method STM* by 2.8% in the overall score. Note that TMN uses a smaller backbone. This indicates that our tangled reference encoder and TSE can greatly boost the memory network's generalization performance while accelerating the inference process. Additionally, our TMN-L achieves comparable performance to the state-of-the-art methods, which are pretrained on static images. SITVOS slightly surpasses our TMN-L by 0.7% on the overall score due to the well-designed interactive transformer for feature matching with the larger memory bank. Such a structure helps exploit the robust spatiotemporal context of target objects while slowing down the inference speed. It is worth noting that TMN-L outperforms the baseline STM by 1.2% on the overall score, especially 1.9% on the F_u score, demonstrating that the effective use of edge/contours and target area information in mask features can improve the model's generalization ability and boost the performance.

Running Speed. We compare the inference speed of our proposed TMN with the previous real-time VOS methods. The running environment and hardware condition have an impact on the inference speed of the model. Additionally, since the running speeds of existing methods reported in those works are tested on different platforms, direct comparison is not fair. Therefore, we test the FPS of those real-time VOS methods (SAT [3], FRTM [46], SiamMask [55], STM-cycle [26], and BMVOS [6]) with official codes on the same platform (NVIDIA 1080 Ti). The running speed of all methods was tested five times on the DAVIS16-val set, and the results were averaged. As Table 4 shows, our TMN performs better and runs faster than these real-time VOS methods. Although SiamMask is comparable to TMN in terms of running speed, its performance is much worse, with 14.6% in the $J\&F$ score lower than TMN.

4.3 Ablation Study

We do all the ablation experiments on the DAVIS17-val set with our TMN, which takes ResNet34 [10] as the backbone and uses the input resolution 240×427. Here, we list the ablation studies of SH and fusing stages, foreground/background utilization, TSE, memory size and organization strategies, the backbone, input resolutions, and the fusion module. Note that the memory organization strategy in Tables 5, 6, and 7 is consistent with the baseline STM for fair comparison.

SH and Fusing Stages. In the mask stream, an SH attached to a certain backbone stage is used to reconstruct the predicted mask. We demonstrate the effectiveness of the SH, and then we change

Table 5. Ablation Study of the SH, Fusion Stage, and Fusion Types of the Foreground/Background of the Object Mask to TMN

	Stage				SH	Background			TSE	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)
	r1	r2	r3	r4		conv1	concat	branch				
1										70.1	67.7	72.4
2	✓				✓					70.7	68.2	73.3
3	✓	✓			✓					**71.5**	**69.2**	**73.9**
4	✓	✓	✓		✓					69.4	67.1	71.7
5	✓	✓	✓	✓	✓					71.0	68.5	73.5
6	✓	✓								70.7	68.2	73.2
7	✓	✓			✓	✓				71.9	69.7	74.0
8	✓	✓			✓		✓			**72.9**	**70.2**	**75.6**
9	✓	✓			✓			✓		70.8	68.4	73.3
10	✓	✓			✓		✓		✓	**73.8**	**71.1**	**76.4**

Table 6. Effectiveness of Different Fusion Modules of Our TMN on the DAVIS17 Validation Set [44]

Fusion Module	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)
"add"	73.0	70.3	75.7
"concat"	70.5	67.9	73.2
"AFC"	**73.8**	**71.1**	**76.4**

Table 7. Accuracy and Speed of Different Input Resolutions and Different Backbones of Our TMN on the DAVIS17 Validation Set [44]

Resolution	Backbone		$\mathcal{J}\&\mathcal{F}$ (%)	FPS
	ResNet34	ResNet50		
(240, 427)	✓		73.8	46.6
		✓	74.5	38.3
(320, 570)	✓		77.1	34.7
		✓	77.7	27.5

the number of the stages used in the mask stream to explore the fusion effect from different stages. The results are reported in Table 5. The numbers in the first column are the experiment order used in the following. Experiment 1 shows the quantitative results of our baseline without the mask stream (the RGB frame and corresponding mask are concatenated as the input of the frame stream) and the TSE. The baseline's architecture is consistent with STM, designed with normal concatenation operation in most memory-based methods. Experiments 2 through 5 show the quantitative results of adding our mask stream, which extracts mask features from different stages (r1, r2, r3, r4) of ResNet. Experiment 6 shows the necessity of our SH to reconstruct the predicted mask (compared with Experiment 3). In summary, fusing features from r1 and r2 obtains the best results, and adding the proposed SH further improves. We also find that the mask stream contributes slightly (0.6%, Experiment 6 vs Experiment 1) without SH. Thus, SH is vital for effective mask utilization.

Foreground/Background Utilization. The background of mask prediction contains spatial information about similar targets in the environment, which can help handle distraction problems. To make the foreground and background more contrastive, we test three ways to fuse the foreground

Table 8. Effectiveness of Our Memory Bank Organization Mechanism with Different Memory Sizes

Memory Size	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)	FPS
3	**74.3**	**71.5**	**77.0**	**51.4**
5	73.4	70.6	76.2	50.5
7	74.0	71.3	76.6	50.4
9	73.6	70.9	76.3	50.0
11	73.8	71.1	76.5	49.2
13	73.8	71.1	76.5	48.7
15	73.7	71.1	76.4	47.7
Unlimited	73.8	71.1	76.4	46.6

and background of mask prediction: (1) the foreground and background are added after passing through a convolution layer, respectively, and then fed into the mask stream, denoted as "conv1"; (2) the foreground and background are concatenated first and then fed into the mask stream, denoted as "concat"; and (3) two branches are used to encode the foreground and background, denoted as "branch" Table 5 (Experiments 7–9) show the quantitative results of different ways of foreground/background utilization. The second way obtains the best results, 1.4% gains against Experiment 3.

Target State Estimator. We do other experiments to analyze the effectiveness of our TSE. Table 5 (Experiment 10) shows the quantitative results of our network by adding the TSE. The results show that the TSE can boost the segmentation quality by 0.9% (Experiment 10 vs Experiment 8).

Fusion Module. We design different fusion modules to inject edge information from the mask stream into the frame stream effectively. We test three fusing ways: (1) features from two streams (i.e., mask stream and frame stream) are added directly, denoted as "add"; (2) features from two streams are concatenated first and then passed through a 1×1 convolution, denoted as "concat"; and (3) features from two streams are fused by an AFC module [48], denoted as "AFC." Table 6 shows the quantitative results of different fusion modules. Our TMN with the AFC module obtains the best results, 1.3% gain against "add" and 3.2% gain against "concat."

Input Resolution and Backbone. We adjust the input resolution and experiment with different backbones, ResNet34 and ResNet50 [10], as shown in Table 7. We can see that our model can achieve higher FPS with 240×427 resolution but better performance with 320×530 resolution. Additionally, we find that the input resolution has a greater impact on our model's performance than the backbone.

Impact of Different Memory Size and Memory Strategies. We tested how the memory size M impacts the performance in Table 8. When M is large enough (unlimited), our memory maintenance is the same with STM [41]. When $M = 3$, our TMN obtains the best results, demonstrating that our dynamic memory bank mechanism can effectively exclude noisy features and benefit segmentation quality, even though only three historical features are stored. Moreover, the memory overhead can be decreased by using three frames as the reference sets, consequently improving our model's efficiency. Additionally, we experiment with four types of memory organization strategies: (1) drop the old features and append the new features, denoted as "drop"; (2) replace the features with the minimum state score, denoted as "replace"; (3) inject the features with the minimum state score to the new features, denoted as "inject to new"; and (4) inject the new features to the features with the minimum state score, denoted as "inject to old." The procedure of injecting (k_1, v_1, s_1) to (k_2, v_2, s_2)

Table 9. Effectiveness of Different Memory Bank Organization Mechanisms

Strategy	$\mathcal{J}\&\mathcal{F}$ (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)	FPS
"drop"	73.9	71.2	76.6	**52.2**
"replace"	**74.3**	**71.5**	**77.0**	51.4
"inject to new"	73.8	71.1	76.4	49.4
"inject to old"	73.5	70.8	76.2	49.4

can be formulated as follows:

$$corr = softmax\left(\frac{k_2 \cdot k_1}{\sqrt{C}}\right), \tag{8}$$

$$\begin{aligned} \hat{k_1} &= corr \times k_1 \\ \hat{v_1} &= corr \times v_1 \end{aligned}, \tag{9}$$

$$\begin{aligned} \hat{k_2} &= \alpha k_2 + (1-\alpha)\hat{k_1} \\ \hat{v_2} &= \alpha v_2 + (1-\alpha)\hat{v_1}, \\ \hat{s_2} &= 2 \cdot s_1 \cdot \alpha \end{aligned} \tag{10}$$

where $(\hat{k_2}, \hat{v_2}, \hat{s_2})$ represent the fusing features and $\alpha = s_2/(s_1 + s_2)$. As Table 9 shows, it achieves the best performance by replacing the features with the minimum state score. Additionally, we noticed that fusing historical features does not bring performance improvement. We explained that the historical features with poor state scores might "pollute" the memory bank. We observed that the key frames are very important for VOS. Therefore, we use the state estimator to choose key frames with high state scores.

5 CONCLUSION

We proposed a TMN for real-time VOS. Specifically, we design a tangled reference encoder to exploit the mask's edge features and object characteristics, which benefit segmentation quality. Additionally, we developed a TSE to provide the mask state feedback. Based on the state score, we proposed a simple yet effective memory bank organization mechanism to alleviate the memory overhead and computational burden brought by the unlimited memory bank used in many memory-based methods. Our TMN is trained end-to-end without bells and whistles. It achieves competitive performance with a high speed (66 FPS on the DAVIS16-val set).

REFERENCES

[1] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. 2017. One-shot video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 221–230.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2017. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2017), 834–848.

[3] Xi Chen, Zuoxin Li, Ye Yuan, Gang Yu, Jianxin Shen, and Donglian Qi. 2020. State-aware tracker for real-time video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9384–9393.

[4] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. 2021. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5559–5568.

[5] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. 2021. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. *Advances in Neural Information Processing Systems* 34 (2021), 11781–11794.

[6] Suhwan Cho, Heansung Lee, Minjung Kim, Sungjun Jang, and Sangyoun Lee. 2022. Pixel-level bijective matching for video object segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 129–138.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Los Alamitos, CA, 248–255.

[8] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W. Taylor. 2021. SSTVOS: Sparse spatiotemporal transformers for video object segmentation. *arXiv preprint arXiv:2101.08833* (2021).

[9] Wenbin Ge, Xiankai Lu, and Jianbing Shen. 2021. Video object segmentation using global and instance embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16836–16845.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[11] Yuk Heo, Yeong Jun Koh, and Chang-Su Kim. 2021. Guided interactive video object segmentation using reliability-based attention maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7322–7330.

[12] Li Hu, Peng Zhang, Bang Zhang, Pan Pan, Yinghui Xu, and Rong Jin. 2021. Learning position and target consistency for memory-based video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4144–4154.

[13] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G. Schwing. 2018. VideoMatch: Matching based video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 54–70.

[14] Peiliang Huang, Junwei Han, Nian Liu, Jun Ren, and Dingwen Zhang. 2021. Scribble-supervised video object segmentation. *IEEE/CAA Journal of Automatica Sinica* 9, 2 (2021), 339–353.

[15] Xuhua Huang, Jiarui Xu, Yu-Wing Tai, and Chi-Keung Tang. 2020. Fast video object segmentation with temporal aggregation network and dynamic template matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8879–8889.

[16] Ge-Peng Ji, Keren Fu, Zhe Wu, Deng-Ping Fan, Jianbing Shen, and Ling Shao. 2021. Full-duplex strategy for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4922–4933.

[17] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. 2018. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 784–799.

[18] Joakim Johnander, Martin Danelljan, Emil Brissman, Fahad Shahbaz Khan, and Michael Felsberg. 2019. A generative appearance model for end-to-end video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8953–8962.

[19] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7482–7491.

[20] Anna Khoreva, Anna Rohrbach, and Bernt Schiele. 2018. Video object segmentation with language referring expressions. In *Proceedings of the Asian Conference on Computer Vision*. 123–141.

[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Zihang Lai, Erika Lu, and Weidi Xie. 2020. MAST: A memory-augmented self-supervised tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6479–6488.

[23] Meng Lan, Jing Zhang, Fengxiang He, and Lefei Zhang. 2022. Siamese network with interactive transformer for video object segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 1228–1236.

[24] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. 2018. High performance visual tracking with Siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8971–8980.

[25] Yu Li, Zhuoran Shen, and Ying Shan. 2020. Fast video object segmentation using the global context module. In *Proceedings of the European Conference on Computer Vision*. 735–750.

[26] Yuxi Li, Ning Xu, Jinlong Peng, John See, and Weiyao Lin. 2020. Delving into the cyclic mechanism in semi-supervised video object segmentation. *arXiv preprint arXiv:2010.12176* (2020).

[27] Yongqing Liang, Xin Li, Navid Jafari, and Qin Chen. 2020. Video object segmentation with adaptive feature bank and uncertain-region refinement. *arXiv preprint arXiv:2010.07958* (2020).

[28] Huaijia Lin, Xiaojuan Qi, and Jiaya Jia. 2019. AGSS-VOS: Attention guided single-shot video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3949–3957.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*. 740–755.

[30] Chang Liu, Wenguan Wang, Jianbing Shen, and Ling Shao. 2018. Stereo video object segmentation using stereoscopic foreground trajectories. *IEEE Transactions on Cybernetics* 49, 10 (2018), 3665–3676.

[31] Weide Liu, Guosheng Lin, Tianyi Zhang, and Zichuan Liu. 2020. Guided co-segmentation network for fast video object segmentation. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 4 (2020), 1607–1617.
[32] Yong Liu, Ran Yu, Xinyuan Zhao, and Yujiu Yang. 2021. Quality-aware and selective prior enhancement memory network for video object segmentation. In *Proceedings of the CVPR Workshop*, Vol. 2.
[33] Xinkai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. 2020. Video object segmentation with episodic graph memory networks. *arXiv preprint arXiv:2007.07020* (2020).
[34] Xiankai Lu, Wenguan Wang, Chao Ma, Jianbing Shen, Ling Shao, and Fatih Porikli. 2019. See more, know more: Unsupervised video object segmentation with co-attention Siamese networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3623–3632.
[35] Xiankai Lu, Wenguan Wang, Jianbing Shen, David Crandall, and Jiebo Luo. 2022. Zero-shot video object segmentation with co-attention Siamese networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 4 (2022), 2228–2242.
[36] Xiankai Lu, Wenguan Wang, Jianbing Shen, David J. Crandall, and Luc Van Gool. 2021. Segmenting objects from relational visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 11 (2021), 7885–7897.
[37] Xiankai Lu, Wenguan Wang, Jianbing Shen, Yu-Wing Tai, David J. Crandall, and Steven C. H. Hoi. 2020. Learning video object segmentation from unlabeled videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8960–8970.
[38] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. 2018. PReMVOS: Proposal-generation, refinement and merging for video object segmentation. In *Computer Vision—ACCV 2018*. Lecture Notes in Computer Science, Vol. 11364. Springer, 565–580.
[39] Jianbiao Mei, Mengmeng Wang, Yeneng Lin, Yi Yuan, and Yong Liu. 2021. TransVOS: Video object segmentation with transformers. *arXiv preprint arXiv:2106.00588* (2021).
[40] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. 2018. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7376–7385.
[41] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. 2019. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9226–9235.
[42] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. 2017. Learning video object segmentation from static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2663–2672.
[43] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. 2016. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 724–732.
[44] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. 2017. The 2017 DAVIS Challenge on video object segmentation. *arXiv preprint arXiv:1704.00675* (2017).
[45] Sucheng Ren, Wenxi Liu, Yongtuo Liu, Haoxin Chen, Guoqiang Han, and Shengfeng He. 2021. Reciprocal transformations for unsupervised video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15455–15464.
[46] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. 2020. Learning fast and robust target models for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7406–7415.
[47] Hongje Seong, Junhyuk Hyun, and Euntai Kim. 2020. Kernelized memory network for video object segmentation. In *Proceedings of the European Conference on Computer Vision*. 629–645.
[48] Lei Sun, Kailun Yang, Xinxin Hu, Weijian Hu, and Kaiwei Wang. 2020. Real-time fusion network for RGB-D semantic segmentation incorporating unexpected obstacle detection for road-driving images. *IEEE Robotics and Automation Letters* 5, 4 (2020), 5558–5565.
[49] Mingjie Sun, Jimin Xiao, Eng Gee Lim, Bingfeng Zhang, and Yao Zhao. 2020. Fast template matching and update for video object tracking and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10791–10799.
[50] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro-i-Nieto. 2019. RVOS: End-to-end recurrent network for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5277–5286.
[51] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. 2019. FEELVOS: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9481–9490.
[52] Paul Voigtlaender and Bastian Leibe. 2017. Online adaptation of convolutional neural networks for video object segmentation. *arXiv preprint arXiv:1706.09364* (2017).

[53] Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. 2020. Siam R-CNN: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6578–6588.
[54] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. 2021. SwiftNet: Real-time video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1296–1305.
[55] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. 2019. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1328–1338.
[56] Wenguan Wang, Jianbing Shen, Xiankai Lu, Steven C. H. Hoi, and Haibin Ling. 2020. Paying attention to video object pattern understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 7 (2020), 2413–2428.
[57] Wenguan Wang, Jianbing Shen, Fatih Porikli, and Ruigang Yang. 2018. Semi-supervised video object segmentation with super-trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 4 (2018), 985–998.
[58] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. 2019. RANet: Ranking attention network for fast video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3978–3987.
[59] Lili Wei, Congyan Lang, Liqian Liang, Songhe Feng, Tao Wang, and Shidi Chen. 2022. Weakly supervised video object segmentation via dual-attention cross-branch fusion. *ACM Transactions on Intelligent Systems and Technology* 13, 3 (2022), 1–20.
[60] Dongming Wu, Xingping Dong, Ling Shao, and Jianbing Shen. 2022. Multi-level representation learning with semantic alignment for referring video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4996–5005.
[61] Jiannan Wu, Yi Jiang, Peize Sun, Zehuan Yuan, and Ping Luo. 2022. Language as queries for referring video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4974–4984.
[62] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, and Wenxiu Sun. 2021. Efficient regional memory network for video object segmentation. *arXiv preprint arXiv:2103.12934* (2021).
[63] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. 2018. YouTube-VOS: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 585–601.
[64] Xiaohao Xu, Jinglu Wang, Xiao Li, and Yan Lu. 2022. Reliable propagation-correction modulation for video object segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 2946–2954.
[65] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. 2020. SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 12549–12556.
[66] Le Yang, Junwei Han, Dingwen Zhang, Nian Liu, and Dong Zhang. 2018. Segmentation in weakly labeled videos via a semantic ranking and optical warping network. *IEEE Transactions on Image Processing* 27, 8 (2018), 4025–4037.
[67] Zongxin Yang, Yunchao Wei, and Yi Yang. 2020. Collaborative video object segmentation by foreground-background integration. In *Proceedings of the European Conference on Computer Vision*. 332–348.
[68] Zongxin Yang, Yunchao Wei, and Yi Yang. 2021. Associating objects with transformers for video object segmentation. *Advances in Neural Information Processing Systems* 34 (2021).
[69] Rui Yao, Guosheng Lin, Shixiong Xia, Jiaqi Zhao, and Yong Zhou. 2020. Video object segmentation and tracking: A survey. *ACM Transactions on Intelligent Systems and Technology* 11, 4 (2020), 1–47.
[70] Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin. 2020. A transductive approach for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6949–6958.
[71] Zongji Zhao, Sanyuan Zhao, and Jianbing Shen. 2021. Real-time and light-weighted unsupervised video object segmentation network. *Pattern Recognition* 120 (2021), 108120.
[72] Tianfei Zhou, Jianwu Li, Shunzhou Wang, Ran Tao, and Jianbing Shen. 2020. MATNet: Motion-attentive transition network for zero-shot video object segmentation. *IEEE Transactions on Image Processing* 29 (2020), 8326–8338.

Received 2 August 2022; revised 10 February 2023; accepted 13 February 2023