SWEM: Towards Real-Time Video Object Segmentation with Sequential Weighted Expectation-Maximization

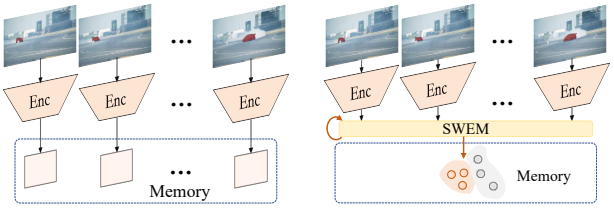
Zhihui Lin[1*], Tianyu Yang[2], Maomao Li[2], Ziyu Wang[3], Chun Yuan[4†], Wenhao Jiang[3], and Wei Liu[3]
[1]Department of Computer Science and Technologies, Tsinghua University, Beijing, China
[2]Tencent AI Lab, Shenzhen, China [3]Tencent Data Platform, Shenzhen, China
[4]Tsinghua Shenzhen International Graduate School, Peng Cheng Lab, Shenzhen, China
{lin-zh14@mails, yuanc@sz}.tsinghua.edu.cn tianyu-yang@outlook.com
{limaomao07, cswhjiang}@gmail.com wangziyukobe@163.com wl2223@columbia.edu

Abstract

*Matching-based methods, especially those based on space-time memory, are significantly ahead of other solutions in semi-supervised video object segmentation (VOS). However, continuously growing and redundant template features lead to an inefficient inference. To alleviate this, we propose a novel **S**equential **W**eighted **E**xpectation-**M**aximization (SWEM) network to greatly reduce the redundancy of memory features. Different from the previous methods which only detect feature redundancy between frames, SWEM merges both intra-frame and inter-frame similar features by leveraging the sequential weighted EM algorithm. Further, adaptive weights for frame features endow SWEM with the flexibility to represent hard samples, improving the discrimination of templates. Besides, the proposed method maintains a fixed number of template features in memory, which ensures the stable inference complexity of the VOS system. Extensive experiments on commonly used DAVIS and YouTube-VOS datasets verify the high efficiency (36 FPS) and high performance (84.3% $\mathcal{J}\&\mathcal{F}$ on DAVIS 2017 validation dataset) of SWEM.*

1. Introduction

Semi-supervised video object segmentation (VOS) has seized great interest recent years [3, 6, 10, 16, 18, 21, 25, 28, 32, 33, 37, 38, 42, 43, 47, 49, 51] in the computer vision community. It aims to segment the objects of interest from the background in a video, where only the mask annotation of the first frame is provided during testing. A group of early methods concentrate on on-line fine-tuning [2, 3, 19, 29, 30] with the first annotated frame. However, these method tends to suffer from model degradation caused by target appearance changes as video goes on. Besides, propagation-based methods use masks computed in previous frames to estimate masks in the current frame [7, 33, 45, 48], which is, however, vulnerable to occlusions and rapid motion.



(a) STM (b) SWEM

Figure 1. Instead of storing all past frames features as memory, just like STM [32] and following methods [6, 16, 37, 38] do, our SWEM sequentially updates a compact set of bases with a fixed size, greatly reducing the inter-frame and intra-frame redundancy.

Recently, matching-based VOS methods [5, 6, 16, 17, 25, 28, 37, 38, 42, 43, 49, 52] have achieved striking performance. Such matching-based methods first exploit previous frames to construct target templates, and then calculate the pixel-level correlations between the new coming frame embeddings and the target templates to perform the segmentation. As seen in Figure 1, the Space-Time Memory Network (STM) [32] and the following STM-like methods [6, 16, 37, 38, 49] leverage memory networks to store template features every T frames endlessly, which is prone to missing key-frame information and running out of memory for long-term videos. Besides, given that the inter-frame redundancy of video features would harm the efficiency of matching, another group of methods AFB_URR [25] and Swift [43] take advantage of the similarity of inter-frame features to selectively update partial features. Nonetheless, they all fail to balance the performance and efficiency through a hand-crafted similarity threshold.

Although past efforts have achieved promising results, we argue that both inter-frame redundancy and intra-frame one pose the main obstacles that prevent efficient template

*Work done during an internship at Tencent AI Lab
†Corresponding Author

matching. Here comes to a question that can we achieve a real-time VOS system by considering both the inter-frame and intra-frame redundancy simultaneously? In this paper, we will explore its feasibility.

Inspired by the Expectation-Maximization Attention (EMA) [21], we intend to construct a set of low-rank bases for memory features through Expectation-Maximization (EM) [9] iterations. Here, the number of bases is far less than that of image pixels. Thus, bases can be regarded as a more compact representation, which can greatly reduce the intra-frame redundancy. Instead of applying EM directly, we adopt Weighted Expectation-Maximization (WEM) with the predicted mask as the **fixed** weights to explicitly construct foreground and background bases in each frame. What's more, we also propose a weighted EM with **adaptive** weights, which give larger weights for hard samples during generating bases. Here, the hard samples refer to those pixels that are not well expressed by bases, but are important for object segmentation.

WEM can deal with the intra-frame redundancy effectively; however, inter-frame one remains unsolved. Applying WEM on a single frame is efficient, but the computation complexity will be dramatically increased if directly applying it to all growing memory features. To further reduce the inter-frame redundancy, we propose the Sequential Weighted Expectation-Maximization (SWEM), where features of only one frame participate in the EM iterations during the memory updating stage. The memory bases will be updated with the new frame features through similarities rather than a simple linear combination. Formally, this updating process is equivalent to a weighted average of all past frame features. As shown in Figure 1, compared with STM [32] which saves all historical frame information as the memory template of objects, our SWEM only updates a more compact set of bases sequentially, thus greatly reducing the inter-frame and intra-frame redundancy.

Our contributions can be summarized as follows:

- We propose a fast and robust matching-based method for VOS, dubbed Sequential Weighted Expectation-Maximization (SWEM) network, where a set of compact bases are constructed and updated sequentially, reducing both the inter- and intra-frame redundancy.
- We introduce an adaptive weights calculation approach for weighted EM, which makes the base features pay more attention to hard samples.
- Without bells and whistles, SWEM reaches a level close to state-of-the-art performance, while maintaining an inference speed of 36 FPS.

2. Related Work

Matching-based Methods for VOS. Recent years have seen a surge of interest in video object segmentation under the semi-supervised setting. A number of matching-based methods [5, 6, 16, 17, 25, 28, 32, 37, 38, 43, 44, 47, 49] regard the first or intermediate frames as a target template, which is then used to match the pixel-level feature embedding in the new frame. To obtain both long-term and short-term object appearance information, FEELVOS [42] and CFBI [52] match the current frame with both the first frame and the previous frame to obtain both global and local temporal dependencies. Besides, STM [32] and the following methods [6, 16, 37, 38, 49] store multiple memory templates from all previous frames as templates, which is redundant and time-consuming during matching. In contrast, we propose a novel method named SWEM, which only stores a set of low-rank and updated basis features for each target, making the target representation more compact and efficient.

Learning Fast and Robust VOS. Leaning a fast and robust model is a common goal since both accuracy and speed are important in practical applications [4, 23, 36, 44, 47, 53]. RANet [47] only uses the first frame as the target template for an acceptable speed. As tracker-based methods, SiamMask [44] and SAT [4] only process the region of interest. TVOS [53] directly propagates target masks according to feature similarity in the embedding space. In general, to achieve fast VOS, the previous methods sacrificed the integrity of the target representation, which substantially degrades segmentation performance. Swift [43] uses a variation-aware trigger module to compute the inter-frame difference to update frames with diverse dynamics. Further, only partial features that are significantly different from memory features will be updated.

In this work, we consider reducing inter- and intra-frame redundancy simultaneously. The proposed weighted EM greatly reduces the intra-frame redundancy by iteratively constructing compact base features for the whole frame. To diminish the inter-frame redundancy, we further extend weighted EM in a sequential manner, which can adaptively update the model without increasing the number of matching templates, thus making our model fast and robust.

3. Preliminaries

3.1. Expectation-Maximization Algorithm

The expectation-maximization (EM) [9] is an iteration-based algorithm, which can be used to estimate parameters of latent variable modes by maximizing the likelihood. The task is defined as estimating model parameters θ based on observation data set $\mathbf{X}$ and corresponding latent variables $\mathbf{Z}$. Each EM iteration involves two steps, the Expectation step (E step) and the Maximization step (M step). At the r-th iteration, E step finds the posterior $P(\mathbf{Z}|\mathbf{X}, \theta^{r-1})$ and computes the expectation:

$$\mathcal{Q}(\theta, \theta^{r-1}) = \sum P(\mathbf{Z}|\mathbf{X}, \theta^{r-1}) \ln P(\mathbf{X}, \mathbf{Z}|\theta). \quad (1)$$

M step estimates parameters by maximizing the above data likelihood:

$$\theta^r = \arg\max_\theta \mathcal{Q}(\theta, \theta^{r-1}). \tag{2}$$

The E step and the M step are alternately executed R times to achieve the convergence criterion.

3.2. Expectation-Maximization Attention

Expectation-Maximization Attention (EMA) [21] is proposed to formulate the attention mechanism [46] into an expectation-maximization manner. Specifically, instead of regarding *all pixels* as reconstruction bases, EMA iteratively estimates a much more compact set of bases for each image. EMA consists of three steps, namely *Responsibility Estimation* (RE), *Likelihood Maximization* (LM), and *Data Re-estimation* (DR). Denote $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N \in \mathbb{R}^{N\times C}$ as image feature, $\mathcal{M} = \{\boldsymbol{\mu}_k\}_{k=1}^K \in \mathbb{R}^{K\times C}$ as the randomly initialized base features, where N, C, and K indicate the numbers of pixels, channels, and bases. RE estimates the hidden variable $\mathbf{Z} = \{z_{nk}\}_{n=1,k=1}^{N,K} \in \mathbb{R}^{N\times K}$, where the responsibility z_{nk} represents the probability of the n-th pixel belonging to the k-th base:

$$z_{nk} = \frac{\exp(\mathbf{x}_n \boldsymbol{\mu}_k^\top / \tau)}{\sum_{j=1}^K \exp(\mathbf{x}_n \boldsymbol{\mu}_j^\top / \tau)}. \tag{3}$$

Here, τ is a hyper-parameter which controls the shape of distribution $\mathbf{Z}$. Then, LM updates base features $\mathcal{M}$ by applying the weighted average on feature $\mathbf{X}$. That is, the k-th base is updated by:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk}\mathbf{x}_n}{\sum_{n=1}^N z_{nk}}. \tag{4}$$

Note that RE and LM are iteratively executed R times until convergence. Finally, DR reconstructs a low-rank version of $\mathbf{X}$ using $\tilde{\mathbf{X}} = \mathbf{Z}\mathcal{M}$. Since K is much less than N, basis set $\mathcal{M}$ can be treated as a compact representation for image feature $\mathbf{X}$. Inspired by EMA, we consider replacing redundant memory features with more compact base features.

3.3. Redundancy of the Space-time Memory

To get a more intuitive understanding of feature redundancy in videos, we evaluate the inter-frame and intra-frame cosine similarities of videos features on the DAVIS 2017 [35] validation set using the image encoder of STM [32] as the feature extractor. For each pixel in the current frame, we first calculate its maximum similarity with all pixels in the previous frame. In this way, N such maximum similarities can be obtained. In Figure 2, we list the histogram of the maximum similarities, where the horizontal coordinate is the similarity range. Most of the similarities are larger than 0.6, and nearly 87% of similarities are larger than 0.9, indicating high inter-frame redundancy

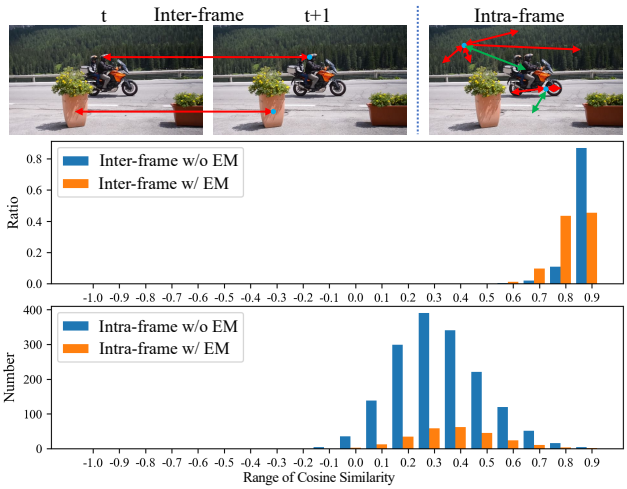


Figure 2. The illustration of inter-frame and intra-frame redundancy of video features.

in video sequences. In contrast, computing the maximum similarity for intra-frame redundancy measurement is not appropriate since spatial continuity would make most maximum similarities exceed 0.9. Thus, we calculate all pairwise similarities and count the average pair number of each frame under different similarities. The third line in Figure 2 shows the statistics. Most of the similarities between the two pixels in an image are positive, and more than 70% of them are larger than 0.3, which demonstrates the ubiquity of intra-frame redundancy.

To verify that the EM algorithm can find a more compact representation for image features and thus restrain the frame redundancy, we calculate the inter-frame and intra-frame similarity with a basis set rather than the entire image feature, where the basis set consists of 256 bases evaluated via EM iterations. Specifically, instead of calculating similarities between inter-frame bases, we calculate the maximum similarity between each frame feature and the base feature of the previous frame. As seen in Figure 2, more than 99% of inter-frame similarities are larger than 0.7. That is, although each frame has only 256 base features, which is far less than pixel number, it still meets the need of inter-frame matching. As for intra-frame similarities, although the similarity distribution is basically the same as that of the whole image feature, the number of large similarities has decreased significantly, which demonstrates the intra-frame redundancy is greatly reduced via EM iterations.

4. Proposed Approach

We first introduce the weighted EM, which leverages the predicted mask as weights to explicitly construct foreground and background bases separately in each frame. Furthermore, the **adaptive** weights make the model pay more attention to hard samples to improve the segmentation performance. Then, the core part of this work, the SWEM

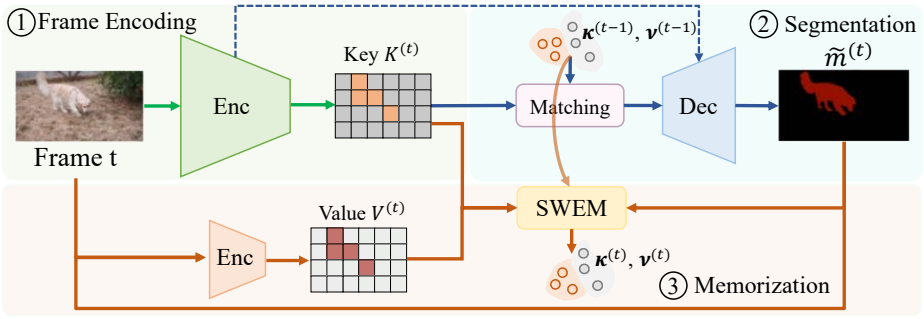


Figure 3. The matching-based pipeline of SWEM. The backbone network receives the t-th frame to capture general image features as **Key** $\mathbf{K}^{(t)}$. The features are used to match with target-specific memories. Through the matching process, the re-aggregated value and similarity map are obtained to be the target features for the final segmentation. Multi-level skip connections help refine the segmentation results for low to high resolution. The predicted mask is then employed with the intermediate image features to update bases via our SWEM.

algorithm is detailed, which shows how to convert **growing** frame features into **fixed-size** bases. Finally, we describe the matching-based pipeline of the proposed SWEM.

4.1. Weighted Expectation-Maximization

Although we have proved that using EM to find a more compact representation can reduce both inter and intra-frame redundancy, we argue that naively employing EM to learn a set of bases for memory features is not a reasonable solution in the VOS system. The reason here is that the memory bases would mix with both the foreground and background, which is unfavorable for object segmentation. Instead, it is desirable to build low-rank foreground and background bases separately. To this end, we leverage Weighted Expectation-Maximization (WEM) [1, 13, 14, 27, 40], which is widely used for weighted data clustering. When using WEM to produce bases for images, Eq. (4) would be modified as:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk} w_n \mathbf{x}_n}{\sum_{n=1}^{N} z_{nk} w_n}, \tag{5}$$

where w_n is the weight for $\mathbf{x}_n$. It is equivalent to "seeing the n-th feature w_n times" [14]. Note that WEM makes it possible to construct separate foreground and background bases for template matching, where the object foreground mask and the background mask of each frame can be used as the corresponding *fixed* weights to substitute w_n. In this way, any irregular target region can be represented by a set of bases with a fixed size, which greatly reduces intra-frame redundancy.

The essence of using WEM for compact representation learning is to perform clustering on all pixels in the image, and make different bases to represent each pixel. Because the number of bases is far less than that of pixels, the constructed segmentation target template would be incomplete to some extent and even be faced with the target lost situation. The expression degree of each pixel is different during basis set construction. Some pixels have little contribution for bases, but are very important for object segmentation, which are so-called hard samples. To ensure that the hard samples could be assigned larger weights during basis set construction, we propose to *adaptively* adjust the weights of pixels rather than directly employing the *fixed* weights calculated via foreground and background masks.

We estimate a confidence score for each pixel by a foreground-background binary classification. Specifically, after the E-step of WEM iteration, each pixel is classified by the foreground or background bases, the classification probability of the n-th pixel can be calculated as:

$$\begin{aligned} P^{fg}(\mathbf{x}_n) &= \frac{\sum_{k=1}^{K} \mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_k^{fg})}{\sum_{k=1}^{K} [\mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_k^{fg}) + \mathcal{K}(\mathbf{x}_n, \boldsymbol{\mu}_k^{bg})]}, \\ P^{bg}(\mathbf{x}_n) &= 1 - P^{fg}(\mathbf{x}_n), \end{aligned} \tag{6}$$

where $\boldsymbol{\mu}_k^{fg}$ and $\boldsymbol{\mu}_k^{bg}$ are foreground and background bases separately. $\mathcal{K}(\cdot)$ is a kernel function for calculating the similarity of two input features. Specifically, $\mathcal{K}(\mathbf{a}, \mathbf{b}) = \exp(\frac{\mathbf{a}\mathbf{b}^\top / \tau}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|})$. Eq. (6) can be treated as a coarse segmentation result, since it provides the result of whether each pixel corresponds to the foreground or the background. Besides, the final segmentation (i.e., the output of the network decoder) can be obtained, which is considered more accurate than the coarse one since it is additionally constrained by ground-truth annotations. If coarse segmentation of a pixel is consistent with the final one, this pixel can be regarded as an easy sample. Otherwise, it would be regarded as a hard sample. We believe that the inconsistency of hard samples is because these pixels are neglected during the base construction process, which makes them struggle to achieve the same result as the final segmentation. Supposing that m^{fg} and m^{bg} are soft masks of final segmentation, the *adaptive weights* are estimated by:

$$w_n^{fg} = m_n^{fg} P^{bg}(\mathbf{x}_n), \quad w_n^{bg} = m_n^{bg} P^{fg}(\mathbf{x}_n). \tag{7}$$

The more inconsistent the coarse and final segmentation are, the higher weights would be given for base construction.

4.2. Sequential Weighted EM

To reduce inter-frame redundancy, previous methods [25, 43] set a similarity threshold to ignore or merge similar features between frames. However, the segmentation performance and computational complexity are sensitive to this hand-crafted threshold. What's worse, it is hard to find an appropriate threshold to make a trade-off between performance and complexity. In this paper, we extend WEM in a sequential manner, yielding a Sequential Weighted EM (SWEM) algorithm to reduce both intra- and inter-frame redundancy without any threshold hyperparameter.

At the time step t, the ideal solution is to apply WEM to all previous $t-1$ frames for basis set construction. However, the growing scale of computing is unacceptable, which is not feasible for long-term segmentation. Therefore, we introduce a sequential weighted average of frame features when computing base assignment, where the weights are estimated with time-dependent responsibility $\mathbf{Z}^{(t)}$.

Concretely, we extend the WEM sequentially and reformulate Eq. (5) as:

$$\boldsymbol{\mu}_k^{(t)} = \frac{\sum_{i=1}^{t}\sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)} \mathbf{x}_n^{(t)}}{\sum_{i=1}^{t}\sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)}}. \tag{8}$$

Note that we implement Eq. (8) in a recursive manner, *i.e.*, the numerator α and denominator β are computed by $\boldsymbol{\alpha}_k^{(t)} = \boldsymbol{\alpha}_k^{(t-1)} + \sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)} \mathbf{x}_n^{(t)}$ and $\boldsymbol{\beta}_k^{(t)} = \boldsymbol{\beta}_k^{(t-1)} + \sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)}$, then $\boldsymbol{\mu}_k^{(t)} = \boldsymbol{\alpha}_k^{(t)} / \boldsymbol{\beta}_k^{(t)}$.

This sequential way of computing base assignment achieves more smooth and adaptable model updating. Instead of storing all frame bases, maintaining only one set of adaptable bases is undoubtedly more friendly to hardware and can also help realize a real-time VOS system. Algorithm1 shows the detailed pipeline of our SWEM at time step t.

It is also worth noting that the updating of bases is lazy in SWEM. Since $z_{nk}^{(t)}$ represents the similarity degree between $\mathbf{x}_n^{(t)}$ and $\boldsymbol{\mu}_k^{(t)}$, and if a base feature has more similar features with the current frame, it will be updated more quickly. This lazy updating strategy can help SWEM be more robust to noises and prevent drifts. In another way, $w_n^{(t)}$ also enables the hard samples to be updated faster.

4.3. Matching-based Pipeline

The overview of the proposed SWEM network is illustrated in Figure 3. The whole pipeline mainly consists of three stages, including 1) feature encoding, 2) segmentation, and 3) memorization.

Encoding. Similar to previous matching-based methods with Space-Time Memory [6, 16, 23, 25, 32, 37, 38, 43, 49], frames are encoded into **Key-Value** pairs ($\mathbf{K} \in \mathbb{R}^{N\times C}$ and $\mathbf{V} \in \mathbb{R}^{N\times C'}$) for memory query and read. We adopt the encoder structure of STCN [6] to extract image features. The key features are also reused for memorization. Specific network details are described in Section 5.1.

Algorithm 1: The SWEM at the time step t

Input:
features of frame t: $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times C}$,
mask of frame t: $m^{fg,(t)} \in [0,1]^N$ and $m^{bg,(t)} \in [0,1]^N$
bases at time step $t-1$: $\mathcal{M}^{fg,(t-1)} \in \mathbb{R}^{K\times C}$ and $\mathcal{M}^{bg,(t-1)} \in \mathbb{R}^{K\times C}$,
accumulated numerators and denominators: $\boldsymbol{\alpha}^{fg,(t-1)}$, $\boldsymbol{\alpha}^{bg,(t-1)}$ and $\boldsymbol{\beta}^{fg,(t-1)}, \boldsymbol{\beta}^{bg,(t-1)}$

Output:
bases $\mathcal{M}^{fg,(t)}$ and $\mathcal{M}^{bg,(t)}$

/* Superscript symbols fg and bg are omitted for simplicity. */

$\mathcal{M}^{(t)} \leftarrow \mathcal{M}^{(t-1)}$
$w^{(t)} \leftarrow m^{(t)}$
for $r = 1$ *to* R **do**
  // SW-**E** step, estimate responsibilities:
  $z_{nk}^{(t)} \leftarrow \frac{\mathcal{K}(\mathbf{x}_n^{(t)}, \boldsymbol{\mu}_k^{(t)})}{\sum_{j=1}^{K}\mathcal{K}(\mathbf{x}_n^{(t)}, \boldsymbol{\mu}_j^{(t)})}$
  // SW-**M** step, update bases:
  $\boldsymbol{\alpha}_k^{(t)} \leftarrow \boldsymbol{\alpha}_k^{(t-1)} + \sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)} \mathbf{x}_n^{(t)}$
  $\boldsymbol{\beta}_k^{(t)} \leftarrow \boldsymbol{\beta}_k^{(t-1)} + \sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)}$
  $\boldsymbol{\mu}_k^{(t)} \leftarrow \boldsymbol{\alpha}_k^{(t)} / \boldsymbol{\beta}_k^{(t)}$
  // SW-**W** step, calculate weights:
  $P^{fg}(\mathbf{x}_n^{(t)}) \leftarrow \frac{\sum_{k=1}^{K}\mathcal{K}(\mathbf{x}_n^{(t)}, \boldsymbol{\mu}_k^{fg,(t)})}{\sum_{k=1}^{K}[\mathcal{K}(\mathbf{x}_n^{(t)}, \boldsymbol{\mu}_k^{fg,(t)}) + \mathcal{K}(\mathbf{x}_n^{(t)}, \boldsymbol{\mu}_k^{bg,(t)})]}$
  $P^{bg}(\mathbf{x}_n^{(t)}) \leftarrow 1 - P^{fg}(\mathbf{x}_n^{(t)})$
  $w_n^{fg,(t)} \leftarrow m_n^{fg,(t)} P^{bg}(\mathbf{x}_n^{(t)})$
  $w_n^{bg,(t)} \leftarrow m_n^{bg,(t)} P^{fg}(\mathbf{x}_n^{(t)})$
end
$n = 1, 2, ..., N;\ k = 1, 2.., K$

Segmentation. The segmentation stage includes feature matching and mask decoding. At time step t, the **Key** features $\mathbf{K}^{(t)}$ are used as query to read memory features from $\boldsymbol{\kappa}^{(t-1)}$ and $\boldsymbol{\nu}^{(t-1)}$, where $\boldsymbol{\kappa}$ and $\boldsymbol{\nu}$ are base features corresponding to the key and value features and they are concatenations of foreground and background bases ($[\boldsymbol{\kappa}^{fg}; \boldsymbol{\kappa}^{bg}] \in \mathbb{R}^{2K\times C}$ and $[\boldsymbol{\nu}^{fg}; \boldsymbol{\nu}^{bg}] \in \mathbb{R}^{2K\times C'}$). A non-local [46] formed matching process is applied as follows:

$$\hat{\mathbf{V}}_n^{(t)} = \sum_{k=1}^{2K} \frac{\mathcal{K}(\mathbf{K}_n^{(t)}, \boldsymbol{\kappa}_k^{(t-1)})}{\sum_{j=1}^{2K}\mathcal{K}(\mathbf{K}_n^{(t)}, \boldsymbol{\kappa}_j^{(t-1)})} \boldsymbol{\nu}_k^{(t-1)}. \tag{9}$$

$\hat{\mathbf{V}}^{(t)}$ is a low-rank reconstruction using memory bases,

which is helpful to segmentation tasks. Different from the previous STM-like methods, our memory bases are explicitly separated into foreground and background. Therefore, the correlations $\mathcal{K}(\mathbf{K}_n^{(t)}, \boldsymbol{\kappa}_k^{fg,(t-1)})$ and $\mathcal{K}(\mathbf{K}_n^{(t)}, \boldsymbol{\kappa}_k^{bg,(t-1)})$ can also be used as important segmentation clues. However, the rank of base features is not fixed for different videos since unsorted correlations are not suitable as the inputs for CNNs. To tackle this problem, we design a permutation-invariant operation, which can produce segmentation clues from correlations. Define $\mathcal{K}_n^{fg,(t)} \in \mathbb{R}^K$ and $\mathcal{K}_n^{bg,(t)} \in \mathbb{R}^K$ as the correlations of $\mathbf{K}_n^{(t)}$ with all foreground and background bases, respectively. The permutation-invariant feature $\mathbf{S}^{(t)}$ can be calculated by:

$$\mathbf{S}_{nl}^{(t)} = \frac{\sum_{j \in \text{topl}(\mathcal{K}_n^{fg,(t)})} \mathcal{K}_{nj}^{fg,(t)}}{\sum_{j \in \text{topl}(\mathcal{K}_n^{fg,(t)})} \mathcal{K}_{nj}^{fg,(t)} + \sum_{j \in \text{topl}(\mathcal{K}_n^{bg,(t)})} \mathcal{K}_{nj}^{bg,(t)}}, \tag{10}$$

where $l = 1, 2, ..., L$. Note that $L \leq K$ is a hyperparameter to control the number of segmentation clues channel and the computation complexity. Besides, $\text{topl}(\cdot)$ means the top-l correlation values.

The decoder takes segmentation clues $[\hat{\mathbf{V}}^{(t)}; \mathbf{S}^{(t)}]$ as input to produce the final mask $\tilde{m}^{(t)}$. Additional skip-connections are also adopted to make use of low-level appearance features.

Memorization. After the segmentation, key features $\mathbf{K}^{(t)}$ are reused for the memorization stage. We adopt another ResNet-18 to re-encode the image-mask pair to obtain value features $\mathbf{V}^{(t)}$. The key bases are updated by $\boldsymbol{\kappa}^{(t)} = \text{SWEM}(\mathbf{K}^{(t)}, \tilde{\text{m}}^{(t)}, \boldsymbol{\kappa}^{(t-1)})$ which is described in Algorithm 1. To maintain the alignment between key and value, the updated value bases is calcudated by $\boldsymbol{\nu}_k^{(t)} = (\boldsymbol{\beta}_k^{(t-1)} \boldsymbol{\nu}_k^{(t-1)} + \sum_{n=1}^{N} z_{nk}^{(t)} w_n^{(t)} \mathbf{v}_n^{(t)}) / \boldsymbol{\beta}_k^{(t)}$, where $\boldsymbol{\beta}$, $\mathbf{Z}$ and w are all produced during the construction of key bases $\boldsymbol{\kappa}$.

5. Implementation Details

5.1. Network Structure

We adopt ResNet-50 [15] as the backbone to extract frame features and ResNet-18 for value feature extraction. All batch normalization layers are frozen. The stage 4 (res4) features are used for memorization and matching processes. These feature maps have a stride of 16 compared with the row image. The temperature hyper-parameter τ is set to 0.05. The number of base features in a group is set as $K = 128$, and the number of iterations R is set as 4 in the SWEM algorithm. We select top-64 ($L = 64$) correlation scores calculated by Eq. (10). For simplicity and fair comparison with STM [32], we use the same two-level decoder, which consists of two refining layers, and each layer contains two residual blocks.

K	FPS	DAVIS 2016 val		DAVIS 2017 val	
		$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$
32	37.3	88.4	87.6	80.2	77.7
64	36.8	88.9	88.0	80.9	78.4
128	36.4	89.5	88.6	81.9	79.3
256	35.5	89.5	88.5	82.0	79.4

Table 1. Ablation study on the number of bases K (with R=4).

5.2. Two-stage Training

Pre-training on static image datasets. Following the previous methods [25, 28, 32, 37, 47], we first perform the pre-training procedure on static image datasets [8, 12, 22, 26, 39]. The input frames are cropped into 384×384 for training. Three frames are generated based on a single image at each step, where the random affine transformations of shearing, rotation, scaling, and cropping are applied. The Adam optimizer [20] with the learning rate $1e$-5 is adopted for all training processes. Besides, we use the cross-entropy loss for the final segmentation.

Training on video datasets. After pre-training on images, we fine-tune the proposed SWEM on video datasets DAVIS 2017 [35] and the YouTube-VOS 2018 [50]. The training process is similar to image pre-training, where the main difference is that we sample the three frames from a video clip randomly instead of one single image. For multi-object frames, we randomly select less than 3 objects. We perform all experiments on a single NVIDIA Tesla V100 GPU with a batch size of 4.

6. Experiments

6.1. Ablation Study

We first analyze the impact of the number of bases K and that of SWEM iterations R, which are key factors affecting the efficiency of the model. Then we investigate the effect of adaptive weights in SWEM on model performance. We directly train all models on video datasets without pre-training on images. Models are evaluated on DAVIS 2016 [34] and DAVIS 2017 validation datasets.

The number of bases K. Table 1 shows the quantitative results and inference speed under different K values. The performance saturates at $K = 128$. When decreasing the number of bases, the performance degraded a lot, while this does not save too much computation, as seen in the inference speed. Therefore, we choose a relatively large $K = 128$ as the default setting.

The number of SWEM iterations R. The number of SWEM iterations affects the efficiency and convergence of bases construction. Table 3 shows results with $R = 1 \sim 7$. The inference speed is sensitive to the number of iterations. Every increase in R decreases the inference speed by $1 \sim 2$

Method	Pub.	I	FPS	DAVIS 2016 val			DAVIS 2017 val		
				$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{F}_M\uparrow$	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{F}_M\uparrow$
STM [32]	ICCV 2019	✓	6	86.5	84.8	88.1	71.6	69.2	74.0
AFB-URR [25]	NeuralPS 2020	✓	4	-	-	-	74.6	73.0	76.1
CFBI [52]	ECCV 2020		5	86.1	85.3	86.9	74.9	72.1	77.7
RANet [47]	ICCV 2019	✓	30	85.5	85.5	85.4	65.7	63.2	68.2
GC [23]	ECCV 2020	✓	25	86.6	**87.6**	85.7	71.4	69.3	73.5
TVOS [53]	CVPR 2020		37	-	-	-	72.3	69.9	74.7
SAT [4]	CVPR 2020		39	83.1	82.6	83.6	72.3	68.6	76.0
SWEM			36	**88.1**	87.3	**89.0**	**77.2**	**74.5**	**79.8**
STM [32](+**YV**)	ICCV 2019	✓	11*	89.3	88.7	89.9	81.7	79.2	84.3
CFBI [52](+**YV**)	ECCV 2020	✓	5	89.4	88.3	90.5	81.9	79.1	84.6
EGMN [28](+**YV**)	ECCV 2020	✓	5	-	-	-	82.8	80.2	85.2
KMN [37](+**YV**)	ECCV 2020	✓	8	90.5	89.5	91.5	82.8	80.0	85.6
SSTVOS [10] (+**YV**)	CVPR 2021		~ 7	-	-	-	82.5	79.9	85.1
RMNet [16] (+**YV**)	CVPR 2021	✓	12	88.8	88.9	88.7	83.5	81.0	86.0
LCM [16] (+**YV**)	CVPR 2021	✓	9	90.7	89.9	91.4	83.5	80.5	86.5
JOINT [31] (+**YV**)	ICCV 2021		4	-	-	-	83.5	80.8	86.2
DMN [24] (+**YV**)	ICCV 2021	✓	7	-	-	-	84.0	81.0	87.0
HMMN [38] (+**YV**)	ICCV 2021	✓	10	90.8	89.6	92.0	84.7	81.9	87.5
AOT [51] (+**YV**)	NeuralPS 2021	✓	19	91.0	89.7	92.3	83.0	80.3	85.7
Swift [43] (+**YV**)	CVPR 2021	✓	25	90.4	**90.5**	90.3	81.1	78.3	83.9
STCN [6] (+**YV**)	NeuralPS 2021	✓	26*	-	-	-	**85.4**	**82.2**	**88.6**
SWEM(+**YV**)		✓	36	**91.3**	89.9	**92.6**	84.3	81.2	87.4

Table 2. Comparisons with previous approaches on DAVIS 2016 and DAVIS 2017 validation sets. '**+YV**' denotes training with additional videos from YouTube-VOS. 'I' indicates the pre-training on image datasets. Note that our SWEM achieves results close to state-of-the-art performance at a speed of 36 FPS on a V100 GPU without IO time. Here, '*' represents the re-evaluation on our hardware for reference.

FPS. $R = 4$ achieves the best trade-off between performance and efficiency.

Adaptive weights in SWEM. Without using adaptive weights (Eq. 7), our performance drops greatly (81.9% $\rightarrow$ 77.6%) while the improvement of inference speed is subtle (36.4 FPS $\rightarrow$ 38.4 FPS). Figure 4 shows the distribution of maximum matching similarities between features of the current frame and previous bases. Although SWEM with adaptive weights has fewer high similarities, it has more similarities above 0.6 than the one with fixed weights (93.4% v.s. 90.5%), guaranteeing fewer missing matches during inference. We also show the qualitative comparison between two kinds of weights in Figure 5. Compared with SWEM with adaptive weights, that with fixed weights is more prone to missing matches, resulting in drift issues.

R	FPS	DAVIS 2016 val		DAVIS 2017 val	
		$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$
1	41.5	87.7	87.3	77.9	75.1
2	39.4	88.7	88.0	79.5	76.9
3	38.3	88.8	87.9	80.8	78.1
4	36.4	89.5	88.6	81.9	79.3
5	34.5	89.1	88.2	81.2	78.4
6	33.0	89.0	88.3	79.8	77.0
7	31.8	88.6	87.8	79.8	77.1

Table 3. Ablation study on the number of SWEM iterations R (with K=128).

6.2. Comparison with SOTA

Datasets and evaluation metrics. We report the results on the DAVIS 2016, DAVIS 2017, and YouTube-VOS 2018 datasets using region similarity $\mathcal{J}$, contour accuracy $\mathcal{F}$, and their mean as metrics.

DAVIS 2016 and DAVIS 2017. Table 2 presents the quan-

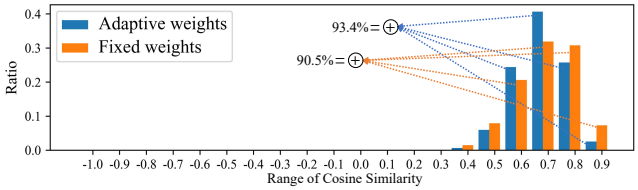


Figure 4. The distribution of maximum matching similarities between current frame features and previous bases.

Figure 5. The qualitative comparison between adaptive-weights and fixed-weights. SWEM with fixed weights (first line for each sample) is struggled to distinguish similar objects while the one with adaptive weights (second line) is competent for this problem. Weights are visualized at the third line for each sample, where the brighter the pixel, the harder it is. As for the person in the middle, the corresponding background bases pay more attention to objects (the remaining person) which are similar to the target.

titative comparisons with recent state-of-the-art video segmentation methods on the DAVIS 2016 and 2017 validation sets. Our method achieves the best $\mathcal{J}$ & $\mathcal{F}$ on both datasets among methods without pre-training on image datasets or additional video datasets. In detail, our method outperforms SAT [4], which runs at a similar speed (39 FPS) with ours, with a large margin on DAVIS2017 (+4.9% $\mathcal{J}$ & $\mathcal{F}$ score). Under the setting of using additional training data from YouTube-VOS (**+YV**), our SWEM surpasses all other top-performing methods. Note that STCN [6] uses a growing memory bank during inference, which harms the long-term segmentation while SWEM keeps the fixed number of bases and has stable computation complexity. SSTVOS [10], AOT [51] and JOINT [31] adopt transformer [41] backbones which are more powerful than ResNet-50. Hierarchical matching used in HMMN [38] also affects the segmentation efficiency deeply. We re-evaluated STM and STCN on our hardware and software environment for reference. Our SWEM is capable of achieving an inference speed of 36 FPS on the V100 GPU and 27 FPS on the 1080ti GPU.

YouTube-VOS 2018. We make a comparison between our SWEM with previous methods on the YouTube-VOS 2018 via the official evaluation server in Table 4. Note that although SWEM leverages the original ResNet-50 backbone and the identical decoder as STM [32], it achieves the 82.8% overall score which is very close to the state-of-the-art results. Besides, we provide more qualitative and quantitative comparisons in the Supplementary Material.

Method	$\mathcal{G}$	seen		unseen	
		$\mathcal{J}_M \uparrow$	$\mathcal{F}_M \uparrow$	$\mathcal{J}_M \uparrow$	$\mathcal{F}_M \uparrow$
STM [32]	79.4	79.7	84.2	72.8	80.9
AFB-URR [25]	79.6	78.8	83.1	74.1	82.6
EGMN [28]	80.2	80.7	85.1	74.0	80.9
KMN [37]	81.4	81.4	85.6	75.3	83.3
CFBI [52]	81.4	81.1	85.8	75.3	83.4
RMNet [16]	81.5	82.1	85.7	75.7	82.4
SSTVOS [10]	81.7	81.2	85.9	76.0	83.9
LCM [16]	82.0	82.2	86.7	75.7	83.4
DMN [24]	82.5	82.5	86.9	76.2	84.2
HMMN [38]	82.6	82.1	87.0	76.8	84.6
JOINT [31]	83.1	81.5	85.9	**78.7**	86.5
AOT [51]	**83.7**	**82.5**	**87.5**	77.9	**86.7**
SAT* [4]	63.6	67.1	70.2	55.3	61.7
TVOS* [53]	67.8	67.1	69.4	63.0	71.6
FRTM* [36]	72.1	72.3	76.2	65.9	74.1
GC* [23]	73.2	72.6	75.6	68.9	75.7
Swift* [11]	77.8	77.8	81.8	72.3	79.5
STCN* [6]	**83.0**	81.9	86.5	**77.9**	**85.7**
SWEM*	82.8	**82.4**	**86.9**	77.1	85.0

Table 4. Comparison with state-of-the-art methods on the YouTube-VOS 2018 validation dataset. We report all of the mean Jaccard ($\mathcal{J}$), the boundary ($\mathcal{F}$) scores for seen and unseen categories, and the overall scores $\mathcal{G}$. Besides, we use '*' to indicate those methods with an inference speed $>$ 20 FPS. Note SSTVOS, JOINT and AOT are transformer-based methods.

7. Conclusion

In this paper, we proposed a fast yet robust model for semi-supervised video object segmentation dubbed Sequential Weighted Expectation-Maximum (SWEM) network, which is capable of constructing compact target templates with low redundancy for pixel-wise matching. The weighted EM algorithm is used to construct bases for foreground and background features separately and reduce the intra-frame redundancy. We also proposed to compute adaptive weights instead of fixed weights when generating bases, which forces bases to pay more attention to hard samples, so as to reduce the missing match. We extended the weighted EM to sequential weighted EM to process sequential data and completely reduce the inter-frame redundancy. Overall, our method achieves a performance close to the state-of-the-art on VOS at 36 FPS.

Acknowledgement

This work was supported by NSFC project Grant No.U1833101, SZSTI Grant No.JCYJ20190809172201639 and WDZC20200820200655001, the Joint Research Center of Tencent and Tsinghua.

References

[1] Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012. 4

[2] Linchao Bao, Baoyuan Wu, and Wei Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5977–5986, 2018. 1

[3] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 221–230, 2017. 1

[4] Xi Chen, Zuoxin Li, Ye Yuan, Gang Yu, Jianxin Shen, and Donglian Qi. State-aware tracker for real-time video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 7, 8

[5] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1189–1198, 2018. 1, 2

[6] 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. 1, 2, 5, 7, 8

[7] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and accurate online video object segmentation via tracking parts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7415–7424, 2018. 1

[8] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE TPAMI*, 37(3):569–582, 2015. 6

[9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 2

[10] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5912–5921, 2021. 1, 7, 8

[11] Bhat G. et al. Learning what to learn for video object segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. 8

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 6

[13] Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. SIAM, 2012. 4

[14] Israel Dejene Gebru, Xavier Alameda-Pineda, Florence Forbes, and Radu Horaud. Em algorithms for weighted-data clustering with application to audio-visual scene analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(12):2402–2415, 2016. 4

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[16] Li Hu, Peng Zhang, Bang Zhang, Pan Pan, Yinghui Xu, and Rong Jin. Learning position and target consistency for memory-based video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4144–4154, 2021. 1, 2, 5, 7, 8

[17] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. Videomatch: Matching based video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 54–70, 2018. 1, 2

[18] Joakim Johnander, Martin Danelljan, Emil Brissman, Fahad Shahbaz Khan, and Michael Felsberg. A generative appearance model for end-to-end video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[19] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for object tracking. In *The DAVIS Challenge on Video Object Segmentation*, 2017. 1

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[21] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3

[22] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 280–287, 2014. 6

[23] Shan Y. Li Y., Shen Z. Fast video object segmentation using the global context module. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 5, 7, 8

[24] Shuxian Liang, Xu Shen, Jianqiang Huang, and Xian-Sheng Hua. Video object segmentation with dynamic memory networks and adaptive object alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8065–8074, 2021. 7, 8

[25] Yongqing Liang, Xin Li, Navid Jafari, and Qin Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In *Advances in neural information processing systems (NeurIPS)*, 2020. 1, 2, 5, 6, 7, 8

[26] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[27] Bo Long, Zhongfei Zhang, Xiaoyun Wu, and Philip S Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592, 2006. 4

[28] Xinkai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. Video object segmentation with episodic graph memory networks. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 6, 7, 8

[29] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*, pages 565–580. Springer, 2018. 1

[30] K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. Video object segmentation without temporal information. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1515–1530, 2018. 1

[31] Yunyao Mao, Ning Wang, Wengang Zhou, and Houqiang Li. Joint inductive and transductive learning for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9670–9679, 2021. 7, 8

[32] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 5, 6, 7, 8

[33] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2663–2672, 2017. 1

[34] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016. 6

[35] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 3, 6

[36] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 8

[37] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 5, 6, 7, 8

[38] Hongje Seong, Seoung Wug Oh, Joon-Young Lee, Seongwon Lee, Suhyeon Lee, and Euntai Kim. Hierarchical memory matching network for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12889–12898, 2021. 1, 2, 5, 7, 8

[39] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):717–729, 2015. 6

[40] George C Tseng. Penalized and weighted k-means for clustering with scattered objects and prior information in high-throughput biological data. *Bioinformatics*, 23(17):2247–2255, 2007. 4

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 8

[42] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[43] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. Swiftnet: Real-time video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1296–1305, 2021. 1, 2, 5, 7

[44] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1338, 2019. 2

[45] Wenguan Wang, Jianbing Shen, Fatih Porikli, and Ruigang Yang. Semi-supervised video object segmentation with super-trajectories. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):985–998, 2018. 1

[46] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3, 5

[47] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 6, 7

[48] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7376–7385, 2018. 1

[49] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, and Wenxiu Sun. Efficient regional memory network for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1286–1295, 2021. 1, 2, 5

[50] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 585–601, 2018. 6

[51] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 1, 7, 8

[52] Yang Y. Yang Z., Wei Y. Collaborative video object segmentation by foreground-background integration. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 7, 8

[53] Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin. A transductive approach for video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 7, 8