Video Object Segmentation with Episodic Graph Memory Networks

Xiankai Lu[1], Wenguan Wang[2](✉), Martin Danelljan[2], Tianfei Zhou[1], Jianbing Shen[1], and Luc Van Gool[2]

[1] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
carrierlxk@gmail.com
[2] ETH Zurich, Zürich, Switzerland
wenguanwang.ai@gmail.com
https://github.com/carrierlxk/GraphMemVOS

Abstract. How to make a segmentation model efficiently adapt to a specific video as well as online target appearance variations is a fundamental issue in the field of video object segmentation. In this work, a graph memory network is developed to address the novel idea of "learning to update the segmentation model". Specifically, we exploit an episodic memory network, organized as a fully connected graph, to store frames as nodes and capture cross-frame correlations by edges. Further, learnable controllers are embedded to ease memory reading and writing, as well as maintain a fixed memory scale. The structured, external memory design enables our model to comprehensively mine and quickly store new knowledge, even with limited visual information, and the differentiable memory controllers slowly learn an abstract method for storing useful representations in the memory and how to later use these representations for prediction, via gradient descent. In addition, the proposed graph memory network yields a neat yet principled framework, which can generalize well to both one-shot and zero-shot video object segmentation tasks. Extensive experiments on four challenging benchmark datasets verify that our graph memory network is able to facilitate the adaptation of the segmentation network for case-by-case video object segmentation.

Keywords: Video segmentation · Episodic graph memory · Learn to update

1 Introduction

Video object segmentation (VOS), as a core task in computer vision, aims to predict the target object in a video at the pixel level. Typically, according to

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58580-8_39) contains supplementary material, which is available to authorized users.

whether or not annotations are provided for the first frame during testing, VOS can be categorized into one-shot video object segmentation (O-VOS) [5,48] and zero-shot video object segmentation (Z-VOS) [52]. Provided with only first-frame annotations, O-VOS is to identify and segment the labeled object instances in the rest of the video [7,15,17,27,33,69]; whereas Z-VOS targets at automatically inferring the primary objects without any test-time indications [44,45].

O-VOS is a challenging task because there are no further assumptions regarding to the specific target object and the application scenes often contain similar distractor objects. To tackle these challenges, earlier methods typically perform network finetuning over each annotated object [5,50]. Though effective, this is quite time-consuming. Current popular solutions [11,31,49,51,70] are instead built upon an efficient *matching* based framework; they formulate the task as a differentiable matching procedure between the support set (*i.e.*, the first labeled frame or prior segmented frames) and query set (*i.e.*, current frame). Thus they can directly assign labels to the query frame, according to the pixel-wise similarity to the annotated first frame and/or previous processed frames.

Although omitting first-frame finetune and improving the performance to some extent, matching based O-VOS methods still suffer from several limitations. First, they typically learn a generic matching network and then apply it to test videos directly, failing to make full use of first-frame target-specific information. As a result, they cannot efficiently adapt to the input video. Second, as the segmentation targets may undergo appearance variation (*i.e.*, fast motion, occlusion), it is meaningful to perform online model updating. Third, matching based methods only modeling pair-relations between the query and each support frame, neglecting the rich context within the support set.

To address these issues, we take inspiration from the recent development of memory-augmented networks for few-shot learning [39,58] and develop a graph memory network to online adapt the segmentation model to a specific target in one single feed-forward pass. Specifically, by regarding O-VOS as episodic memory reasoning, our approach equips with the ability to slowly learn high-level knowledge for extracting useful representations from the offline training data, and the ability to rapidly fuse the unseen information from the first-frame annotation in the test video, via an external memory module. In this way, our model can internally modulate the output by learning to rapidly cache representation in the memory stores. During the segmentation, to maintain the variations of the object appearance, we perform memory updating by storing and recalling target information from the external memory. Therefore, we can implement the online model updating easily without extensive parameter optimization. In addition, the memory module, built upon the end-to-end memory network [43], is endowed with a graph structure to better mine the relations among memory cells.

The proposed graph memory network is neat and fast. For memory updating, instead of some prior matching based O-VOS models [31] inserting a new element into newly allocated position, our model performs message passing on the fixed-size graph memory without increasing memory consumption. Our model provides a principled framework; it generalizes Z-VOS task well, in which mainstream methods also lack the adaptation capability. As far as we know, this work represents the first effort that addresses both O-VOS and Z-VOS in a unified network design. Experiments on representative O-VOS datasets show the proposed method performs favorably against state-of-the-arts. Without bells and whistles, it also outperforms other competitors on Z-VOS datasets. These promising results demonstrate the efficacy and generalizability of our graph memory network.

2 Related Work

One-shot Video Object Segmentation (O-VOS) is to track the first-frame annotation to subsequent frames at pixel level. Traditional methods usually formulate this task as a *label propagation* process [1,35,40,53,57]. With the renaissance of the connectionism, deep learning based solutions become the domaniant [11,27,33,35] in the field of O-VOS. Among them are three representative strategies. One is the *segmentation-by-detection* scheme [5,6,10,26] that learns a video-specific representation about the first-frame annotated objects and then performs pixel-wise detection in the rest frames. Another one is the *propagation* based pipeline [10,45,60], which propagates segmented masks to fit objects in the upcoming frames. The third, *i.e.*, the more advanced, is the *matching* based strategy [7,11,15,17,25,31,49,51,69,70] which usually trains a prototypical Siamese matching network to find the most matching pixel (or embedding in the feature space) between the first frame (or a segmented frame) and the query frame, and then achieves label assignment accordingly. Some matching-based methods employ internal memory (*e.g.*, ConvLSTM [48,64]) or external memory (*e.g.*, [31,48]) to implicitly or explicitly store previously computed segmentation information for facilitating learning the evolution of objects over time. However, our utilization of memory differs from these methods substantially: **i)** we employ an external memory with learnable read-write controller to rapidly encode new video information for quick segmentation network updating; **ii)** compared to vanilla memory network [28,43], our graph memory network stores memory content in a structured manner that explicitly captures context in cells; **iii)** instead of writing new input to a newly located position [31], our memory is dynamically updated by iterative cell state renewing without increasing the memory size.

Zero-shot Video Object Segmentation (Z-VOS) aims to segment primary objects in unconstrained videos. This task has been widely studied over several decades which also called unsupervised video object segmentation [19,22,35,71]. Traditional methods usually leverage motion [18,29] or saliency cues [12,54] to obtain a heuristic representation for inferring the primary objects. Recent methods were built upon fully convolutional networks. Early methods explored two-stream architectures [8,16,41,72] or variants of recurrent neural networks [42,45,55]. Recent ones address comprehensive foreground reasoning from a global view by non-local structures [24,52]. In this work, rather than these methods learning a universal video foreground object representation and hoping it could generalize well to all unseen scenarios, our episodic memory design allows target-adaption on-the-fly by learning to update the segmentation network.

Learning to Update in VOS. To learn more video-specific information, a direct way is to perform iterative network finetune in the first frame [5,50]. Some recent efforts instead applied meta-learning for model updating [4,61,66], whose basic ideas are similar: learning segmentation model parameters on training videos. This paper, in contrast, uses a graph memory network with learnable, lightweight controllers to assimilate a new video. Thus the model can quickly adapt to unseen scenes and appearance variants through the memory node (cell) updating rather than sensitive and expensive network parameter generation [61,66].

3 Proposed Algorithm

3.1 Preliminary: Episodic Memory Networks

Memory networks augment neural networks with an external memory component [28,43,58], which allow the network to explicitly access the past experiences. They have been shown effective in few-shot learning [39,62,63] and object tracking [67]. Recently, episodic external memory networks have been explored to solve reasoning issues in visual question answering and visual dialog [21,28,43]. The basic idea is to retrieve the information required to answer the question from the memory with trainable read and write operators. Given a collection of input representations, the episodic memory module chooses which parts of the inputs to focus on through the neural attention. It then produces a "memory summarization" representation taking into account the query as well as the stored memory. Each iteration in the episode provides the memory module with newly relevant information about the input. As a result, the memory module has the ability to retrieve new information in each iteration and obtain a new representation about the input.

3.2 Learning to Update

In the context of O-VOS [5], the goal is to learn from the annotated objects in the first frame (*support set*) and predict them in the subsequent frames (*query set*). To this end, traditional methods usually finetune the network and perform online learning for each specific video. In contrast, we construct an episodic memory based learner on variety of tasks (videos), sampled from the distribution of training tasks [4], such that the learned model performs well on new unseen tasks (test videos). Thus we tackle O-VOS as a **"learning to update"** segmentation network procedure [38]: **i)** extracting a task representation from the one-shot support set, and **ii)** updating the segmentation network for the query given the task representation. As shown in Fig. 1, we enhance an episodic memory network with graph structure (*i.e.*, graph memory network) to: **i)** instantly adapt the segmentation network to a specific object, rather than performing lots of finetuning iterations; and **ii)** make full use of context within video sequences. As a result, our graph memory network has two abilities: learn to adjust the segmentation network from one-shot support set during the model initialization phase and learn to take advantage of segmented frames to update the segmentation during frame processing phase. Our model thus can make efficient case-by-case adaption and online updating within one feed-forward process.

3.3 Graph Memory Network

The graph memory network consists of an external graph memory and learnable controllers for memory operating. The memory provides short-term storage for new knowledge encoding and its graph structure allows to fully explore the context. The controllers interact with the graph memory through a number of read and write operations and they are capable of long-term storage via slow updates of the weights. Through the controllers, our model learns a general strategy for the types of representations it should place into the memory and how it should later use these representations for segmentation predictions.

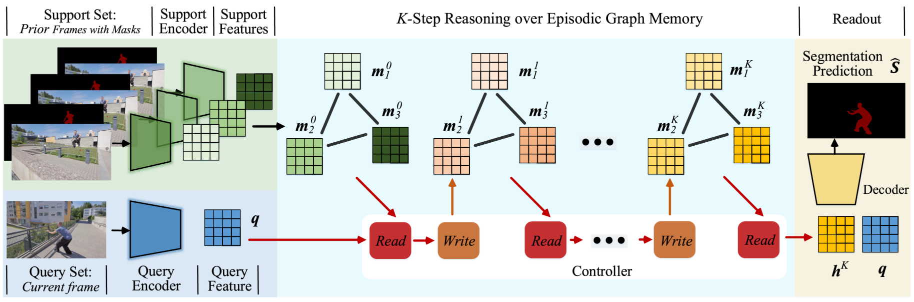

Fig. 1. Illustration of our graph memory based O-VOS method. Previous frames are fed together with the pre-defined or self-segmented masks to the support encoder to initialize graph memory nodes $\{\boldsymbol{m}_i^0\}_i$. Current frame is fed into the query encoder to output the query embedding $\boldsymbol{q}$. The graph memory interacts with $\boldsymbol{q}$ under several episodic reasoning (with learnable read and write controllers) to mine support context and generate video specific features. After K-step episodic reasoning, the decoder predicts segmentation mask $\hat{\boldsymbol{S}}$, based on the episodic feature $\boldsymbol{h}^K$ and query embedding $\boldsymbol{q}$.

The core idea of our graph memory network is to perform K steps of episodic reasoning to efficiently mine the structures in the memory and better capture target-specific information. Specifically, the memory is organized as a size-fixed, fully connected graph $\mathcal{G} = (\mathcal{M}, \mathcal{E})$, where node $m_i \in \mathcal{M}$ denotes i^{th} memory cell, and edge $e_{i,j} = (m_i, m_j) \in \mathcal{E}$ indicates the relation between cell m_i and m_j.

Given a query frame, the support set is considered as the combination of the first annotated frame and previously segmented frames. The graph memory is initialized from $N (= |\mathcal{M}|)$ frames, sampled from the support set. For each memory node m_i, its initial embedding $\boldsymbol{m}_i^0 \in mathbbR^{W \times H \times C}$ is generated by applying a fully convolutional memory encoder to the corresponding support frame to capture both the spatial visual feature as well as segmentation mask information.

Graph Memory Reading. A fully convolutional query encoder is also applied to the query frame to extract the visual feature $\boldsymbol{q} \in \mathbb{R}^{W\times H\times C}$. A learnable *read controller* first takes $\boldsymbol{q}$ as input and generates its initial state $\boldsymbol{h}^0$:

$$\boldsymbol{h}^0 = f_{\mathrm{P}}(\boldsymbol{q}) \in \mathbb{R}^{W\times H\times C}, \tag{1}$$

where $f_{\mathrm{P}}(\cdot)$ indicates a projection function.

At each episodic reasoning step $k \in \{1, ..., K\}$, the read controller interacts with the external graph memory by reading the content. Following the key-value retrieval mechanism in [21,28,43], we first compute the similarity between the query and each memory node m_i:

$$s_i^k = \frac{\boldsymbol{h}^{k-1} \cdot \boldsymbol{m}_i^{k-1}}{\|\boldsymbol{h}^{k-1}\|\|\boldsymbol{m}_i^{k-1}\|} \in [-1, 1]. \tag{2}$$

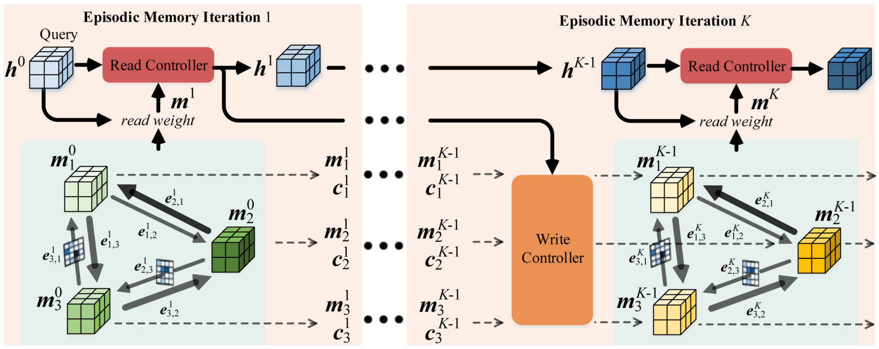

Fig. 2. Illustration of iterative reasoning over the episodic graph memory.

Next, we compute the read weight w_i^k by a softmax normalization function:

$$w_i^k = \exp(s_i^k) \Big/ \sum\nolimits_j \exp(s_j^k) \in [0, 1]. \tag{3}$$

Considering some nodes are noisy due to the underlying camera shift or out-of-view, w_i^k measures the confidence of memory cell m_i. Then the memory summarization $\boldsymbol{m}^k$ is retrieved using this weight to linearly combine the memory cell:

$$\boldsymbol{m}^k = \sum\nolimits_i w_i^k \boldsymbol{m}_i^{k-1} \in \mathbb{R}^{W\times H\times C}. \tag{4}$$

Through Eqs. (2–4), the memory module retrieves the memory cell most similar to $\boldsymbol{h}^k$ to obtain the memory summarization $\boldsymbol{m}^k$. Once reading the memory summarization, the read controller updates its state as follows:

$$\begin{aligned}
\widetilde{\boldsymbol{h}}^k &= \boldsymbol{W}_r^h * \boldsymbol{h}^{k-1} + \boldsymbol{U}_r^h * \boldsymbol{m}^k & \in \mathbb{R}^{W\times H\times C},\\
\boldsymbol{a}_r^k &= \sigma(\boldsymbol{W}_r^a * \boldsymbol{h}^{k-1} + \boldsymbol{U}_r^a * \boldsymbol{m}^k) & \in [0,1]^{W\times H\times C},\\
\boldsymbol{h}^k &= \boldsymbol{a}_r^k \circ \widetilde{\boldsymbol{h}}^k + (\boldsymbol{1} - \boldsymbol{a}_r^k) \circ \boldsymbol{h}^{k-1} & \in \mathbb{R}^{W\times H\times C},
\end{aligned} \tag{5}$$

where $\boldsymbol{W}$s and $\boldsymbol{U}$s are convolution kernels and σ indicates the *sigmoid* activation function. '$*$' and '$\circ$' represent the convolution operation and Hadamard product, respectively. The update gate $\boldsymbol{a}^k$ controls how much previous hidden state $\boldsymbol{h}^{k-1}$ to be kept. In this way, the hidden state of the controller encodes both the graph memory and query representations, which are necessary to produce an output.

Episodic Graph Memory Updating. After each pass through the memory summarization, we need to update the episodic graph memory with the new query input. At each step k, a learnable memory *write controller* updates each memory cell (*i.e.*, graph node) m_i by considering its previous state $\boldsymbol{m}_i^{k-1}$, current content from the read controller $\boldsymbol{h}^k$, and the states from other cells $\{\boldsymbol{m}_j^{k-1}\}_{j\neq i}$. Specifically, following [52], we first formulate the relation $\boldsymbol{e}_{i,j}^k$ from m_j to m_i as the inner-product similarity over their feature matrices:

$$\boldsymbol{e}_{i,j}^k = \boldsymbol{m}_i^{k-1}\ \boldsymbol{W}_e\ \boldsymbol{m}_j^{k-1\top} \in \mathbb{R}^{(WH)\times(WH)}, \tag{6}$$

where $\boldsymbol{W}_e \in \mathbb{R}^{C\times C}$ indicates a learnable weight matrix, and $\boldsymbol{m}_i^{k-1} \in \mathbb{R}^{(WH)\times C}$ and $\boldsymbol{m}_j^{k-1} \in \mathbb{R}^{(WH)\times C}$ are flattened into matrix representations. $\boldsymbol{e}_{i,j}^k$ stores similarity scores corresponding to all pairs of positions in $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$.

Then, for m_i, we compute the summarized information $\boldsymbol{c}_i^k$ from other cells, weighted by their normalized inner-product similarities:

$$\boldsymbol{c}_i^k = \sum\nolimits_{j\neq i} \text{softmax}(\boldsymbol{e}_{i,j}^k)\boldsymbol{m}_j^{k-1} \in \mathbb{R}^{W\times H\times C}, \tag{7}$$

where softmax$(\cdot)$ normalizes each row of the input.

After aggregating the information from neighbors, the memory write controller updates the state of m_i as:

$$\begin{aligned}
\widetilde{\boldsymbol{m}}_i^k &= \boldsymbol{W}_u^m * \boldsymbol{h}^k + \boldsymbol{U}_u^m * \boldsymbol{m}_i^{k-1} + \boldsymbol{V}_u^m * \boldsymbol{c}_i^k & \in \mathbb{R}^{W\times H\times C},\\
\boldsymbol{a}_u^k &= \sigma(\boldsymbol{W}_u^a * \boldsymbol{h}^k + \boldsymbol{U}_u^a * \boldsymbol{m}_i^{k-1} + \boldsymbol{V}_u^a * \boldsymbol{c}_i^k) & \in [0,1]^{W\times H\times C},\\
\boldsymbol{m}_i^k &= \boldsymbol{a}_u^k \circ \widetilde{\boldsymbol{m}}^k + (\boldsymbol{1} - \boldsymbol{a}_u^k) \circ \boldsymbol{m}^{k-1} & \in \mathbb{R}^{W\times H\times C}.
\end{aligned} \tag{8}$$

The graph memory updating allows each memory cell to embed the neighbor information into its representation so as to fully explore the context in the support set. Moreover, by iteratively reasoning over the graph structure, each memory cells encode the new query information and yield progressively improved representations. Compared with traditional memory network [58], the proposed graph memory network brings two advantages: **i)** the memory writing operation is fused into the memory updating procedure without increasing the memory size, and **ii)** avoiding designing complex memory writing strategies [21,39,58]. Figure 2 shows a detailed diagram of memory reading and updating.

Final Segmentation Readout. After K steps of the episodic memory updating, we leverage the final state $\boldsymbol{h}^K$ from the memory read controller to support the prediction on the query:

$$\hat{\boldsymbol{S}} = f_{\mathrm{R}}(\boldsymbol{h}^K, \boldsymbol{q}) \in [0,1]^{W\times H\times 2}, \tag{9}$$

where the readout function $f_{\mathrm{R}}(\cdot)$ gives the final segmentation probability maps.

3.4 Full Network Architecture

Network Configuration. Our whole model is end-to-end fully convolutional. Both the query encoder and memory encoder have the same network architecture, except for the inputs. The query encoder takes an RGB query frame as input, while, for the memory encoder, input is an RGB support frame concatenated with the one-channel softmax object mask and one-hot label map [31]. For the graph memory, the *read controller* (Eq. (5)) and *write controller* (Eq. (8)) are all implemented using ConvGRU [2], with 1 × 1 convolutional kernels. The project function f_{P} (Eq. (1)) is also realized with 1 × 1 convolutional layer. Similar to [59], the readout function f_{R} (Eq. (9)) is implemented with a decoder network, which consists of four blocks with skip connections to the corresponding ResNet50 [14] blocks. The kernel size of each convolution layer in the decoder is set as 3 × 3, excepting the last 1 × 1 convolution layer. The final 2-channel segmentation prediction is obtained by a softmax operation. The query and memory encoders are implemented as the four convolution blocks of ResNet50, initialized by the weights pretrained on ImageNet. The other layers are randomly initialized. Considering memory encoder takes binary mask and instance label maps as input, extra 1 × 1 convolutional layers are used for encoding these inputs. The resulting features are added with RGB features at the first blocks of ResNet50.

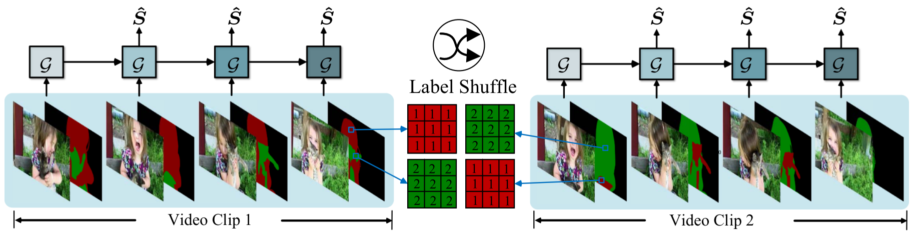

Fig. 3. From video clip to video clip, the instances with associated labels are shuffled. $\mathcal{G}$ denotes the graph memory network and $\hat{\boldsymbol{S}}$ is the output prediction.

Training. For O-VOS, we train our model following the "recurrence training" procedure [13,59]. Each training pass is formed by sampling a support set to build the graph memory and a relevant query set. The core heart of recurrence training is to mimic the inference procedure [4]. For each video, we sample $N+1$

frames to build a support set (first N frames) and a query set (last frame). Thus, the N support frames can be represented by a N-node memory graph. We apply a cross entropy loss for supervising training.

To prevent the graph memory from only memorizing the relation between the instance and the one-hot vector label, we employ the label shuffling strategy [39]. As shown in Fig. 3, every time we run a segmentation episode, we shuffle the one-hot instance labels [49], meaning sometimes the label of specific instance becomes 2 instead of 1 and vice versa. This encourages the segmentation network to learn to distinguish the specific instance in current frame by considering the current training samples rather than memorizing specific relation between the target and the given label. See Sect. 4.3 for detailed experiments for the efficacy of our label shuffling strategy.

To further boost performance, we extend the training set with synthetic videos [31,49,59]. Specifically, for a static image, the video generation technique [33] is adopted to obtain simulated video clips, through different transformation operations (*e.g.*, rotation, scaling, translation and sheering). The static images come from existing image segmentation datasets. After pre-training on the synthetic videos, we use the real video data for finetuning.

For Z-VOS, we follow a similar training protocol as O-VOS, but the input modality only has RGB data. We do not use the label shuffling strategy, as we focus on an object-level Z-VOS setting (*i.e.*, do not discriminate different object instances). More training details for Z-VOS and O-VOS can be found in Sect. 3.5.

Inference. After training, we directly apply the learned network to unseen test videos without online finetuning. For O-VOS, we process each testing video in a sequential manner. For the first N frames, we compute the memory summarization (Eq. (4)) directly and write these frames into the memory. From $(N+1)^{th}$ frame, after segmentation, we would use this frame to update the graph memory. Considering the first frame and its annotation always provide the most reliable information, we re-initialize the node which stores the information about the first frame. Therefore, we use the first annotated frame, last segmented frame and $N-2$ frames sampled from previous segmented frames, as well as their pre-defined or segmented masks to build the memory. For multiple-instances cases, we run our model for each instance independently and obtain a soft-max probability mask for each one. Considering the underlying probability overlap between different instances, we combine these results together with a *soft-aggregation* strategy [59]. Our network achieves fast segmentation speed of 0.2 s per frame.

For Z-VOS, we randomly sample N frames from the same video to build the graph memory, then we process each frame based on the constructed memory. Considering the global information is more important than local information for handling underlying object occlusions and camera movements, we process each frame independently by re-initializing the graph memory with globally sampled frames. Following common practice [8,42,45], we employ CRF [20] binarization and the whole processing speed is about 0.3 s per frame.

3.5 Implementation Details

During pre-training, we randomly crop 384 × 384 patches from static image samples, and the video clip length is $N + 1 = 4$. During the main training, we crop 384 × 640 patches from real training videos. We randomly sample four temporal ordered frames from the same video with a maximum skip of 25 frames to build a training clip. Data augmentation techniques, like rotation, flip and saturation, are adopted to increase the data diversity. For O-VOS, we select a saliency dataset, MSRA10K [9], and semantic segmentation dataset, COCO [23], to synthesize videos. We use all the training videos in DAVIS$_{17}$ [36] and Youtube-VOS [64] for the main training. For Z-VOS, we use two image saliency datasets, MSRA10K [9] and DUT [65], to generate simulated videos. These two datasets are selected following conventions [24,45] for fair comparison. After that, we take advantage of the training set of DAVIS$_{16}$ [34] to finetune the network.

Our model is implemented on *PyTorch* and trained on four NVIDIA Tesla V100 GPUs with 32 GB memory per card. The batch size is set to 16. We optimize the loss function with Adam optimizer using "poly" learning schedule, with the base learning rate of 1e−5 and power of 1.0. The pre-training stage takes about 24 h and the main training stage takes about 16 h for O-VOS.

4 Experiments

To verify the effectiveness and generic applicability of the proposed method, we perform experiments on different VOS settings. In concrete, we first evaluate

Table 1. Evaluation of O-VOS on DAVIS$_{17}$ `val` set (Sect. 4.1), with region similarity $\mathcal{J}$, boundary accuracy $\mathcal{F}$ and average of $\mathcal{J}$&$\mathcal{F}$. Speed is also reported.

Method		OSMN [66]	SIMMASK [51]	FAVOS [7]	RVOS [48]	OSVOS [5]	AGAME [17]	OnAVOS [50]	RGMP [59]
$\mathcal{J}$&$\mathcal{F}$	Mean ↑	54.8	65.4	58.2	60.6	60.3	71.0	65.4	66.7
$\mathcal{J}$	Mean ↑	52.5	54.3	54.6	57.5	56.6	68.5	61.6	64.8
	Recall↑	60.9	62.8	61.1	65.2	63.8	78.4	67.4	74.1
	Decay↓	21.5	19.3	14.1	24.9	26.1	14.0	27.9	18.9
$\mathcal{F}$	Mean ↑	57.1	58.5	61.8	63.6	63.9	73.6	69.1	68.6
	Recall↑	66.1	67.5	72.3	73.2	73.8	83.4	75.4	77.7
	Decay↓	24.3	21.0	18.0	28.2	27.0	15.8	26.6	19.6
Times (s)		0.13	0.028	1.8	1.8	7.0	0.07	13	0.13
Method		OSVOS-S [27]	RANet [56]	FEELVOS [49]	CINM [3]	PReMVO [26]	DMMNet [70]	STM [31]	**Ours**
$\mathcal{J}$& $\mathcal{F}$	Mean ↑	68.0	65.7	71.5	70.6	77.8	70.7	81.8	**82.8**
$\mathcal{J}$	Mean ↑	64.7	63.2	69.1	67.2	73.9	68.1	79.2	**80.2**
	Recall↑	74.2	73.7	79.1	74.5	83.1	77.3	88.7	**90.1**
	Decay↓	15.1	18.6	17.5	24.6	16.2	16.8	**8.0**	6.0
$\mathcal{F}$	Mean ↑	71.3	68.2	74.0	74.0	81.8	73.3	84.3	**85.2**
	Recall↑	80.7	78.8	83.8	81.6	88.9	82.7	91.8	**93.3**
	Decay↓	18.5	19.7	20.1	26.2	19.5	23.5	**10.5**	8.4
Times (s)		4.5	0.13	0.5	38	70	2.7	0.18	0.2

our model on two O-VOS datasets (Sect. 4.1) and then test it on two Z-VOS datasets (Sect. 4.2). Finally, in Sect. 4.3, agnostic experiments are conducted for in-depth analysis.

4.1 Performance for O-VOS

Datasets. Experiments are conducted on two well-known O-VOS benchmarks: DAVIS$_{17}$ [36] and Youtube-VOS [64]. DAVIS$_{17}$ comprises 60 videos for training and 30 videos for validation. Each video contains one or multiple annotated object instances. Youtube-VOS is a large-scale dataset which is split into a `train` set (3,471 videos) and a `val` set (474 videos). The validation set is further divided into `seen` subset which has the same categories (65) as the `train` set and `unseen` subset with 26 unseen categories.

Evaluation Criteria. Following the standard evaluation protocol of DAVIS$_{17}$, the mean region similarity $\mathcal{J}$ and contour accuracy $\mathcal{F}$ are reported. For Youtube-VOS, these two metrics are separately computed for the `seen` and `unseen` sets.

Quantitative Results. The performance of our network on DAVIS$_{17}$ is shown in Table 1 with both online learning and offline approaches. Overall, our model outperforms all the contemporary methods and sets a new state-of-the-art in terms of mean $\mathcal{J}\&\mathcal{F}$ (82.8%), mean $\mathcal{J}$ (80.2%) and mean $\mathcal{F}$ (85.2%). Notably, our method obtains a much higher score for both region similarity and contour accuracy compared to several representative online learning methods: OSVOS [5], OnAVOS [50], AGAME [17] and DMMNet [70]. Furthermore, we report the segmentation speed and memory comparison by averaging the inference times for all instances. We observe that most segmentation-by-detection methods (*e.g.*, OSVOS [5]) consume small GPU memory but need a long time for first frame finetuning and online learning. Meanwhile, most matching based methods (*e.g.*, AGAME [17], FEELVOS [49], and RGMP [59]) achieve fast inference yet suffer from heavy memory cost. However, our method achieves better performance with fast speed and acceptable memory usage.

Moreover, we report the segmentation results on Youtube-VOS in Table 2. Our approach obtains a final score of 80.2%, significantly outperforming state-of-the-arts. Compared to memory-based method S2S [64], our model achieves much higher performance (*i.e.*, 80.2% *vs* 64.4%), which verifies the effectiveness of our external graph memory design. Moreover, our method performs favorably on both *seen* and *unseen* categories. Overall, our method achieves huge performance promotion over time-consuming online learning base methods without invoking online finetuning. This demonstrates the efficacy of our core idea of formulating O-VOS as a procedure of learning to update segmentation network.

Qualitative Results. In Fig. 4, we show qualitative results of our method at different time steps (uniformly sampled percentage w.r.t. the whole video length) on a few representative videos. Specifically, many instances in the first two DAVIS$_{17}$ videos undergo fast motion and background clutter. However, through the graph

Table 2. Evaluation of O-VOS on Youtube-VOS `val` set (Sect. 4.1), with region similarity $\mathcal{J}$ and boundary accuracy $\mathcal{F}$. "Overall": averaged over the four metrics

Method		MSK [33]	OSMN [66]	RGMP [59]	OnAVOS [50]	RVOS [48]	OSVOS [5]	S2SS2S [64]	AGAME [17]	PreMVOS [26]	DMMNet [70]	STM [31]	**Ours**
Overall		53.1	51.2	53.8	55.2	56.8	58.8	64.4	66.1	66.9	58.0	79.4	**80.2**
Seen	Mean $\mathcal{J}\uparrow$	59.9	60.0	59.5	60.1	63.6	59.8	71.0	67.8	71.4	60.3	79.7	**80.7**
	Mean $\mathcal{F}\uparrow$	59.5	60.1	–	62.7	67.2	60.5	70.0	–	75.9	63.5	84.2	**85.1**
Unseen	Mean $\mathcal{J}\uparrow$	45.0	40.6	45.2	46.6	45.5	54.2	55.5	60.8	56.5	50.6	72.8	**74.0**
	Mean $\mathcal{F}\uparrow$	47.9	44.0	–	51.4	51.0	60.7	61.2	–	63.7	57.4	80.9	**80.9**

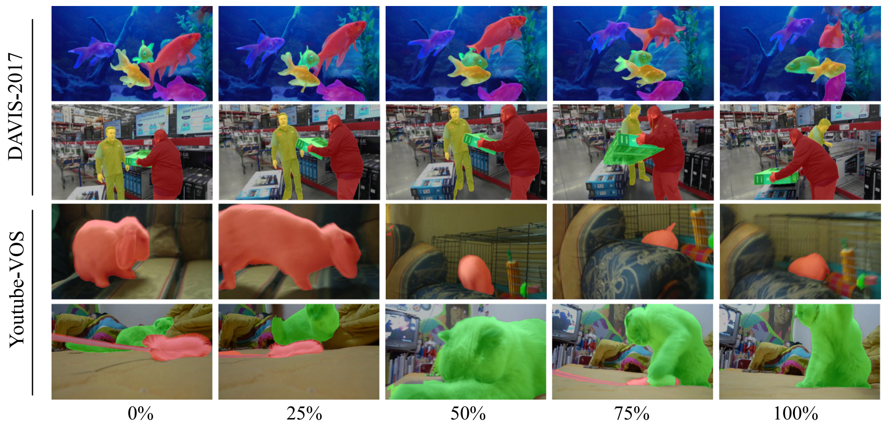

Fig. 4. Qualitative O-VOS results on DAVIS$_{17}$ and Youtube-VOS (Sect. 4.1).

memory mechanism, our segmentation network can handle these challenging factors well. The last two Youtube-VOS videos present challenges that the instances suffer occlusion and out-of-view. Once the occlusion ends, our graph memory allows the segmentation network to re-detect the target and segment it successfully.

4.2 Performance for Z-VOS

Datasets. Experiments are conducted on two challenging datasets: DAVIS$_{16}$ [34] and Youtube-Objects [37]. DAVIS$_{16}$ contains 50 videos with 3,455 frames, covering a wide range of challenges, such as fast motion and occlusion. It is split into a `train` set (30 videos) and a `val` set (20 videos). Youtube-Objects has 126 videos belonging to 10 categories and 25,673 frames in total. The `val` set of DAVIS$_{16}$ and whole Youtube-Objects are used for evaluation.

Evaluation Criteria. We follow the official evaluation protocols [32,34] and report the region similarity $\mathcal{J}$, boundary accuracy $\mathcal{F}$ and time stability $\mathcal{T}$ for DAVIS$_{16}$. Youtube-Objects is evaluated in terms of region similarity $\mathcal{J}$.

Quantitative Results. For DAVIS$_{16}$ [34], we compare our method with 17 state-of-the-arts from DAVIS$_{16}$ benchmark[1] in Table 3. Our method outperforms other competitors across most metrics. In particular, compared with recent

[1] https://davischallenge.org/davis2016/soa_compare.html.

Table 3. Evaluation of Z-VOS on DAVIS$_{16}$ `val` set [34] (Sect. 4.2), with region similarity $\mathcal{J}$, boundary accuracy $\mathcal{F}$ and time stability $\mathcal{T}$.

Method		MSG [29]	NLC [12]	CUT [18]	FST [32]	SFL [8]	LMP [44]	FSEG [16]	LVO [45]	UOVOS [73]
$\mathcal{J}$	Mean ↑	53.3	55.1	55.2	55.8	64.7	70.0	70.7	75.9	73.9
	Recall ↑	61.6	55.8	57.5	64.9	81.4	85.0	83.0	89.1	88.5
	Decay ↓	2.4	12.6	2.2	**0.0**	6.2	1.3	1.5	**0.0**	0.6
$\mathcal{F}$	Mean ↑	50.8	52.3	55.2	51.1	66.7	65.9	65.3	72.1	68.0
	Recall ↑	60.0	61.0	51.9	51.6	77.1	79.2	73.8	83.4	80.6
	Decay ↓	5.1	11.4	3.4	2.9	5.1	2.5	1.8	1.3	0.7
$\mathcal{T}$	Mean ↓	54.8	65.4	58.2	60.6	60.3	71.0	65.4	66.7	39.0
Method		ARP [19]	PDB [42]	MotAdapt [41]	LSMO [46]	AGS [55]	COSNet [24]	AGNN [52]	AnDiff [68]	**Ours**
$\mathcal{J}$	Mean ↑	76.2	77.2	77.2	78.2	79.7	80.5	80.7	81.7	**82.5**
	Recall ↑	89.1	91.1	93.1	87.8	91.1	93.1	94.0	90.9	**94.3**
	Decay ↓	7.0	0.9	5.0	4.1	1.9	4.4	0.03	2.2	4.2
$\mathcal{F}$	Mean ↑	65.3	72.1	70.6	74.5	77.4	79.4	79.1	80.5	**81.2**
	Recall ↑	83.4	83.5	84.4	84.7	85.8	89.5	**90.5**	85.1	90.3
	Decay ↓	7.9	-0.2	3.3	3.5	0.0	5.0	0.03	0.6	5.6
$\mathcal{T}$	Mean ↓	39.3	29.1	27.9	21.2	26.7	**18.4**	33.7	21.4	19.8

matching-based methods: COSNet [24], AGNN [52] and AnDiff [73], our method achieves an average $\mathcal{J}$ score of 82.5% which is 0.8% better than the second best method, AnDiff [73] despite the fact that it utilizes more training samples than ours. Compared with COSNet [24], our method achieves significant performance promotion of 2.0% and 1.8% in terms of mean $\mathcal{J}$ and mean $\mathcal{F}$, respectively. Notably, our method outperforms online learning based methods (*i.e.*, SFL [8], UVOS [73] and LSMO [46]) by a large margin.

We further report results on Youtube-Objects in Table 4 with detailed category-wise performance as well as the final average $\mathcal{J}$ score. As seen, our method surpasses other competitors significantly (reaching 71.4% mean $\mathcal{J}$), especially compared with recent matching based methods [24,52]. Overall, our model consistently yields promising results over different datasets, which clearly illustrates its superior performance and powerful generalizability.

Table 4. Evaluation of Z-VOS on Youtube-Objects [37] (Sect. 4.2). "Mean $\mathcal{J}$↑" denotes the results averaged over all the categories.

Method	LTV [30]	FST [32]	COSEG [47]	ARP [19]	LVO [45]	PDB [42]	FSEG [16]	SFL [8]	MotAdapt [41]	LSMO [46]	AGS [55]	COSNet [24]	AGNN [52]	**Ours**
Airplane (6)	13.7	70.9	69.3	73.6	86.2	78.0	81.7	65.6	77.2	60.5	87.7	81.1	81.1	86.1
Bird (6)	12.2	70.6	76.0	56.1	81.0	80.0	63.8	65.4	42.2	59.3	76.7	75.7	75.9	75.7
Boat (15)	10.8	42.5	53.5	57.8	68.5	58.9	72.3	59.9	49.3	62.1	72.2	71.3	70.7	68.6
Car (7)	23.7	65.2	70.4	33.9	69.3	76.5	74.9	64.0	68.6	72.3	78.6	77.6	78.1	82.4
Cat (16)	18.6	52.1	66.8	30.5	58.8	63.0	68.4	58.9	46.3	66.3	69.2	66.5	67.9	65.9
Cow (20)	16.3	44.5	49.0	41.8	68.5	64.1	68.0	51.1	64.2	67.9	64.6	69.8	69.7	70.5
Dog (27)	18.2	65.3	47.5	36.8	61.7	70.1	69.4	54.1	66.1	70.0	73.3	76.8	77.4	77.1
Horse (14)	11.5	53.5	55.7	44.3	53.9	67.6	60.4	64.8	64.8	65.4	64.4	67.4	67.3	72.2
Motorbike (10)	10.6	44.2	39.5	48.9	60.8	58.3	62.7	52.6	44.6	55.5	62.1	67.7	68.3	63.8
Train (5)	19.6	29.6	53.4	39.2	66.3	35.2	62.2	34.0	42.3	38.0	48.2	46.8	47.8	47.8
Mean $\mathcal{J}$↑	15.5	53.8	58.1	46.2	67.5	65.4	68.4	57.0	58.1	64.3	69.7	70.5	70.8	**71.4**

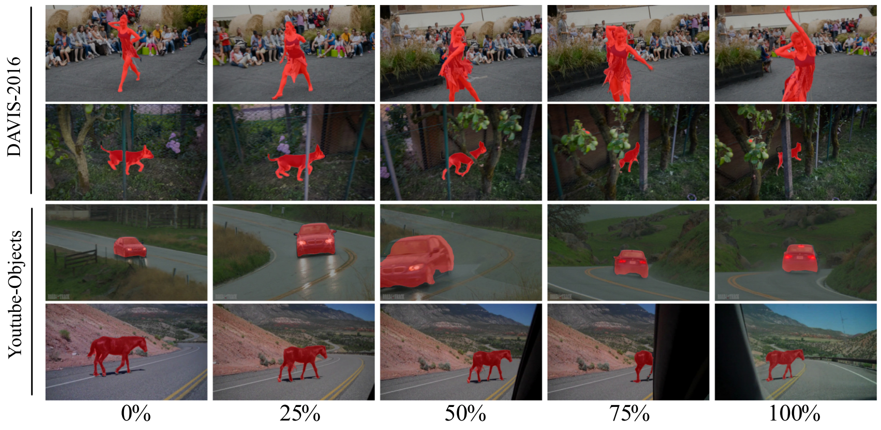

Fig. 5. Qualitative Z-VOS results on DAVIS$_{16}$ and Youtube-Objects (Sect. 4.2).

Quantitative Results. Figure 5 depicts some representative visual results on DAVIS$_{16}$ and Youtube-Objects. As can be observed, our method handles well these challenging scenes, typically with fast motion, partial occlusion and view changes, even without explicit foreground object indication.

4.3 Diagnostic Experiments

In this section, we analyze the contribution of different model components to the final performance. Specifically, we take O-VOS and Z-VOS as exemplar and evaluate all ablated versions on DAVIS$_{17}$ [36] and DAVIS$_{16}$ [34], respectively. The experimental results are evaluated by mean $\mathcal{J}$ and mean $\mathcal{F}$. For each ablated version, we retrain the model from scratch using the same protocol. From the whole results comparison in Table 5, we can draw several essential conclusions.

Table 5. Ablation study of our graph memory network (Sect. 4.3).

Aspect	Method	One-shot VOS DAVIS$_{17}$ [36]		Zero-shot VOS DAVIS$_{16}$ [34]	
		Mean $\mathcal{J}\uparrow$	Mean $\mathcal{F}\uparrow$	Mean $\mathcal{J}\uparrow$	Mean $\mathcal{F}\uparrow$
Full model	Graph memory (3 nodes, 3 episodes)	80.0	85.9	82.5	81.2
Backbone	Direct infer. *w/o* graph memory	73.5	78.4	73.2	72.5
Graph structure	2 nodes	76.0	81.4	78.5	76.8
	4 nodes	79.5	84.6	82.5	81.2
	5 nodes	80.0	85.9	82.5	81.2
State updating	$K = 0$	78.1	82.2	81.2	79.7
	$K = 1$	78.9	83.3	81.6	80.3
	$K = 2$	79.3	84.8	82.0	80.8
	$K = 4$	80.0	85.9	82.5	81.2
Training	*w/o* label shuffling	78.5	82.7	–	–

Graph Memory Network: First, with the proposed graph memory network, our method yields significant performance improvements (+6.5%, +7.5% and +9.3%, +8.7% in terms of mean $\mathcal{J}$ and mean $\mathcal{F}$) than the backbone over different VOS settings. This supports our view that the graph memory network with differential controllers can learn to update the segmentation network effectively.

Memory Size: We next investigate the influence of memory size on the final performance (1^{st} and 3^{rd}–5^{th} rows). We find only a 3-node memory is enough for achieving good performance, further verifying the efficacy of our memory design.

Iterative Memory Reasoning: It is also of interest to assess the impact of our iterative memory updating strategy. When $K = 0$, it means no update for graph memory network, therefore the state of the network is fixed without online learning. In this case, the results deteriorate significantly. We further observe that more steps can boost the performance ($1 \rightarrow 3$) and when the step is increased to certain extent ($K = 4$), the performance remains almost unchanged.

Label Shuffling: Finally we study the effect of our label shuffling strategy. Comparing results on the first and last rows, we can easily observe that shuffling instance labels during network training indeed promotes O-VOS performance.

5 Conclusion

This paper integrates a novel graph memory mechanism to efficiently adapt the segmentation network to specific videos without catastrophic inference/finetune. Through episodic reasoning the memory graph, the proposed model is capable of generating video-specific memory summarization which benefits the final segmentation prediction significantly. Meanwhile, the online model updating can be implemented via learnable memory controllers. Our method is effective and principle, which can be easily extended to Z-VOS setting. Extensive experimental results on four challenging datasets demonstrate its promising performance.

Acknowledgements. This work was partially supported by Zhejiang Lab's Open Fund (No. 2019KD0AB04), Zhejiang Lab's International Talent Fund for Young Professionals, CCF-Tencent Open Fund, ETH Zürich Fund (OK), an Huawei Technologies Oy (Finland) project, an Amazon AWS grant, and Nvidia..

References

1. Badrinarayanan, V., Galasso, F., Cipolla, R.: Label propagation in video sequences. In: CVPR (2010)
2. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: ICLR (2016)
3. Bao, L., Wu, B., Liu, W.: CNN in MRF: video object segmentation via inference in a CNN-based higher-order spatio-temporal MRF. In: CVPR (2018)
4. Behl, H.S., Najafi, M., Arnab, A., Torr, P.H.: Meta learning deep visual words for fast video object segmentation. In: NeurIPS Workshop (2019)

5. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: CVPR (2017)
6. Chen, Y., Pont-Tuset, J., Montes, A., Van Gool, L.: Blazingly fast video object segmentation with pixel-wise metric learning. In: CVPR (2018)
7. Cheng, J., Tsai, Y.H., Hung, W.C., Wang, S., Yang, M.H.: Fast and accurate online video object segmentation via tracking parts. In: CVPR (2018)
8. Cheng, J., Tsai, Y.H., Wang, S., Yang, M.H.: SegFlow: joint learning for video object segmentation and optical flow. In: ICCV (2017)
9. Cheng, M.M., Mitra, N.J., Huang, X., Torr, P.H., Hu, S.M.: Global contrast based salient region detection. IEEE TPAMI **37**(3), 569–582 (2015)
10. Ci, H., Wang, C., Wang, Y.: Video object segmentation by learning location-sensitive embeddings. In: ECCV (2018)
11. Duarte, K., Rawat, Y.S., Shah, M.: CapsuleVOS: semi-supervised video object segmentation using capsule routing. In: ICCV (2019)
12. Faktor, A., Irani, M.: Video segmentation by non-local consensus voting. In: BMVC (2014)
13. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Hu, Y.-T., Huang, J.-B., Schwing, A.G.: VideoMatch: matching based video object segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11212, pp. 56–73. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01237-3_4
16. Jain, S.D., Xiong, B., Grauman, K.: FusionSeg: learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In: CVPR (2017)
17. Johnander, J., Danelljan, M., Brissman, E., Khan, F.S., Felsberg, M.: A generative appearance model for end-to-end video object segmentation. In: CVPR (2019)
18. Keuper, M., Andres, B., Brox, T.: Motion trajectory segmentation via minimum cost multicuts. In: ICCV (2015)
19. Koh, Y.J., Kim, C.: Primary object segmentation in videos based on region augmentation and reduction. In: CVPR (2017)
20. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected CRFs with gaussian edge potentials. In: NIPS (2011)
21. Kumar, A., et al.: Ask me anything: dynamic memory networks for natural language processing. In: ICML (2016)
22. Lee, Y.J., Kim, J., Grauman, K.: Key-segments for video object segmentation. In: ICCV (2011)
23. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
24. Lu, X., Wang, W., Ma, C., Shen, J., Shao, L., Porikli, F.: See more, know more: unsupervised video object segmentation with co-attention Siamese networks. In: CVPR (2019)
25. Lu, X., Wang, W., Shen, J., Tai, Y.W., Crandall, D.J., Hoi, S.C.: Learning video object segmentation from unlabeled videos. In: CVPR (2020)
26. Luiten, J., Voigtlaender, P., Leibe, B.: PReMVOS: proposal-generation, refinement and merging for video object segmentation. In: ACCV (2018)
27. Maninis, K.K., et al.: Video object segmentation without temporal information. IEEE TPAMI **41**(6), 1515–1530 (2018)

28. Miller, A., Fisch, A., Dodge, J., Karimi, A.H., Bordes, A., Weston, J.: Key-value memory networks for directly reading documents. In: EMNLP (2016)
29. Ochs, P., Brox, T.: Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In: ICCV (2011)
30. Ochs, P., Malik, J., Brox, T.: Segmentation of moving objects by long term video analysis. IEEE TPAMI **36**(6), 1187–1200 (2014)
31. Oh, S.W., Lee, J.Y., Xu, N., Kim, S.J.: Video object segmentation using space-time memory networks. In: ICCV (2019)
32. Papazoglou, A., Ferrari, V.: Fast object segmentation in unconstrained video. In: ICCV (2013)
33. Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., Sorkine-Hornung, A.: Learning video object segmentation from static images. In: CVPR (2017)
34. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: CVPR (2016)
35. Perazzi, F., Wang, O., Gross, M., Sorkine-Hornung, A.: Fully connected object proposals for video segmentation. In: CVPR (2015)
36. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 DAVIS challenge on video object segmentation. arXiv preprint arXiv:1704.00675 (2017)
37. Prest, A., Leistner, C., Civera, J., Schmid, C., Ferrari, V.: Learning object class detectors from weakly annotated video. In: CVPR (2012)
38. Rakelly, K., Shelhamer, E., Darrell, T., Efros, A.A., Levine, S.: Meta-learning to guide segmentation. In: ICLR (2019)
39. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: ICML (2016)
40. Shankar Nagaraja, N., Schmidt, F.R., Brox, T.: Video segmentation with just a few strokes. In: ICCV (2015)
41. Siam, M., et al.: Video segmentation using teacher-student adaptation in a human robot interaction (HRI) setting. In: ICRA (2019)
42. Song, H., Wang, W., Zhao, S., Shen, J., Lam, K.-M.: Pyramid dilated deeper ConvLSTM for video salient object detection. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11215, pp. 744–760. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01252-6_44
43. Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: NIPS (2015)
44. Tokmakov, P., Alahari, K., Schmid, C.: Learning motion patterns in videos. In: CVPR (2017)
45. Tokmakov, P., Alahari, K., Schmid, C.: Learning video object segmentation with visual memory. In: ICCV (2017)
46. Tokmakov, P., Schmid, C., Alahari, K.: Learning to segment moving objects. IJCV **127**(3), 282–301 (2019)
47. Tsai, Y.-H., Zhong, G., Yang, M.-H.: Semantic co-segmentation in videos. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 760–775. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_46
48. Ventura, C., Bellver, M., Girbau, A., Salvador, A., Marques, F., Giro-i Nieto, X.: RVOS: end-to-end recurrent network for video object segmentation. In: CVPR (2019)
49. Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., Chen, L.C.: FEELVOS: fast end-to-end embedding learning for video object segmentation. In: CVPR (2019)

50. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: BMVC (2017)
51. Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H.: Fast online object tracking and segmentation: a unifying approach. In: CVPR (2019)
52. Wang, W., Lu, X., Shen, J., Crandall, D.J., Shao, L.: Zero-shot video object segmentation via attentive graph neural networks. In: ICCV (2019)
53. Wang, W., Shen, J., Porikli, F., Yang, R.: Semi-supervised video object segmentation with super-trajectories. IEEE TPAMI **41**(4), 985–998 (2018)
54. Wang, W., Shen, J., Yang, R., Porikli, F.: Saliency-aware video object segmentation. IEEE TPAMI **40**(1), 20–33 (2017)
55. Wang, W., et al.: Learning unsupervised video object segmentation through visual attention. In: CVPR (2019)
56. Wang, Z., Xu, J., Liu, L., Zhu, F., Shao, L.: Ranet: Ranking attention network for fast video object segmentation. In: ICCV (2019)
57. Wen, L., Du, D., Lei, Z., Li, S.Z., Yang, M.H.: JOTS: joint online tracking and segmentation. In: CVPR (2015)
58. Weston, J., Chopra, S., Bordes, A.: Memory networks. ICLR (2015)
59. Wug Oh, S., Lee, J.Y., Sunkavalli, K., Joo Kim, S.: Fast video object segmentation by reference-guided mask propagation. In: CVPR (2018)
60. Xiao, H., Feng, J., Lin, G., Liu, Y., Zhang, M.: MoNet: deep motion exploitation for video object segmentation. In: CVPR (2018)
61. Xiao, H., Kang, B., Liu, Y., Zhang, M., Feng, J.: Online meta adaptation for fast video object segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **42**, 1205–1217 (2019)
62. Xie, G.S., et al.: Attentive region embedding network for zero-shot learning. In: CVPR (2019)
63. Xie, G.S., et al.: Region graph embedding network for zero-shot learning. In: ECCV (2020)
64. Xu, N., et al.: YouTube-VOS: sequence-to-sequence video object segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11209, pp. 603–619. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01228-1_36
65. Yang, C., Zhang, L., Lu, H., Ruan, X., Yang, M.: Saliency detection via graph-based manifold ranking. In: CVPR (2013)
66. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. In: CVPR (2018)
67. Yang, T., Chan, A.B.: Learning dynamic memory networks for object tracking. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11213, pp. 153–169. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01240-3_10
68. Yang, Z., Wang, Q., Bertinetto, L., Bai, S., Hu, W., Torr, P.H.: Anchor diffusion for unsupervised video object segmentation. In: ICCV (2019)
69. Yoon, J.S., Rameau, F., Kim, J., Lee, S., Shin, S., Kweon, I.S.: Pixel-level matching for video object segmentation using convolutional neural networks. In: ICCV (2017)
70. Zeng, X., Liao, R., Gu, L., Xiong, Y., Fidler, S., Urtasun, R.: DMM-Net: differentiable mask-matching network for video object segmentation. In: ICCV (2019)
71. Zhang, D., Javed, O., Shah, M.: Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In: CVPR (2013)

72. Zhou, T., Wang, S., Zhou, Y., Yao, Y., Li, J., Shao, L.: Motion-attentive transition for zero-shot video object segmentation. In: AAAI (2020)
73. Zhuo, T., Cheng, Z., Zhang, P., Wong, Y., Kankanhalli, M.: Unsupervised online video object segmentation with motion property understanding. IEEE TIP **29**, 237–249 (2019)