Fast Video Object Segmentation Using the Global Context Module

Yu Li[1(✉)], Zhuoran Shen[2], and Ying Shan[1]

[1] Applied Research Center (ARC), Tencent PCG, Shenzhen, China
`ianyli@tencent.com`
[2] The University of Hong Kong, Pokfulam, Hong Kong
`cmsflash@connect.hku.hk`

Abstract. We developed a real-time, high-quality semi-supervised video object segmentation algorithm. Its accuracy is on par with the most accurate, time-consuming online-learning model, while its speed is similar to the fastest template-matching method with sub-optimal accuracy. The core component of the model is a novel global context module that effectively summarizes and propagates information through the entire video. Compared to previous approaches that only use one frame or a few frames to guide the segmentation of the current frame, the global context module uses all past frames. Unlike the previous state-of-the-art space-time memory network that caches a memory at each spatio-temporal position, the global context module uses a fixed-size feature representation. Therefore, it uses constant memory regardless of the video length and costs substantially less memory and computation. With the novel module, our model achieves top performance on standard benchmarks at a real-time speed.

Keywords: Video object segmentation · Global context module

1 Introduction

Video object segmentation [1,21,31,37] aims to segment a foreground object from the background on all frames in a video. The task has numerous applications in computer vision. An important one is intelligent video editing. As videos become the most popular form of media on mass content platforms, video content creation is getting increasing levels of attention. Object segmentation on each frame with image segmentation tools is time-consuming and has poor temporal consistency. Semi-supervised video object segmentation tries to solve the problem by segmenting the object in the whole video given only a fine object

---

Y. Li and Z. Shen—contributed equally. This work was done during Zhuoran's internship at Tencent.

---

mask on the first frame. This problem is challenging since object appearance might vary drastically over time in a video due to pose changes, motion, and occlusions, *etc.*
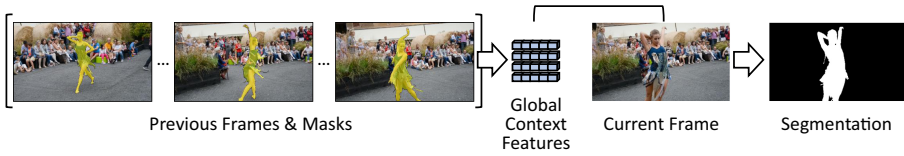

Previous Frames & Masks   Global Context Features   Current Frame   Segmentation

Fig. 1. Our video object segmentation method creates and maintains fixed-size global context features for previous frames and their object masks. These global context features can guide the segmentation on the incoming frame. The global context module is efficient in both memory and computation and achieves high segmentation accuracy. See more details below.

With the success in many computer vision tasks, deep learning techniques are widely used in video object segmentation recently. The essence is to learn an invariant representation that accounts for the variation of object appearance across frames. Some early works [4, 25, 28, 35] in semi-supervised video object segmentation use the first frame to train a model using various data augmentation strategies, which are commonly referred to as online learning-based methods. These methods usually obtain accurate segmentation that are robust to occlusions. However, online learning incurs huge computational costs that lead to several seconds of processing per frame. Another direction is propagation-based methods, *e.g.* [6], which rely on the segmentation of the previous frame to infer for the current frame. These methods are simple and fast, but usually have suboptimal segmentation accuracies. These methods cannot handle occlusions and may suffer from error drifts. Some later works take the advantages of both directions, use both the first frame and the previous frame [26, 34, 39, 42], and achieve both high accuracy and fast processing speed.

A recent work [27] makes a further step to use all previous frames with the corresponding object segmentation results to infer the object mask on the current frame. It proposes a novel space-time memory (STM) module that stores the segmentation information at each processed frame, *i.e.*, this memory module saves information at all spatio-temporal locations on all previous frames. When working on a new frame, a read operation is used to retrieve relevant information from the memory by performing a dense feature correlation operation in both temporal and spatial dimensions. By using this memory that saves all guidance information, the method is robust against drastic object appearance changes and occlusions. It produces promising results and achieves state-of-the-art performance on multiple benchmark datasets.

While STM achieved state-of-the-art performance by making full use of the information from previous frames, leveraging the space-time memory is costly, especially on long videos. As the space-time memory module keeps creating new

memories to save new information and put it together with old memories, the computational cost and memory usage in the feature correlation step increase linearly with the number of frames. This makes the method slower and slower while processing and may easily cause GPU memory overflow. To resolve this issue, the authors propose to reduce the memory saving frequency, *e.g.* saving a new memory every 5 frames. However, the linear complexity with time still exists, and such reduction in memorization frequency defeats the original purpose to utilize information from every previous frame.

In this paper, building upon the idea of STM to use information in all past frames, we develop a compact global representation that summarizes the object segmentation information and guides the segmentation of the next frame. This representation automatically updates when the system moves forward by a frame. The core component of it is a novel global context module (illustrated in Fig. 1). By keeping only a fixed-size set of features, our memory and computational complexities for inference are light and do not increase with time, in comparison to the linear complexities of the STM module. We show that using this highly efficient global context module, our method is about three times faster than STM and do not need to worry about the memory usage. The performance of our method on the single object segmentation benchmark DAVIS 2016 in terms of segmentation accuracy is on par with the state-of-the-art. The results of our method on more challenging multiple object segmentation benchmarks DAVIS 2017 and YouTube-VOS are highly competitive.

The contribution of our paper can be summarized as:

– We propose a novel global context module that reliably maintains segmentation information of the entire video to guide the segmentation of incoming frames.
– We implement the global context module in a light-weight way that is efficient in both memory usage and computational cost.
– Experiments on DAVIS and YouTube-VOS benchmarks show that our proposed global context module can achieve top accuracy in video object segmentation and is highly efficient that runs in real time.

2   Related Works

Online-Learning Methods. Online-learning methods usually fine-tune a general object segmentation network on the object mask of the start frame to teach the network to identify the appearance of the target object in the remaining video frames [4]. They use online adaptation [35], instance segmentation information [25], data augmentation techniques [17], or an integration of multiple techniques [24]. Some methods report that online learning boosts the performance of their model [20,39]. While online learning can achieve high-quality segmentation and is robust against occlusions, it is computationally expensive as it requires fine-tuning for each video. This huge computational burden makes it impractical for most applications.

Offline-Learning Methods Offline-learning methods use strategies like mask propagation, feature matching, tracking, or a hybrid strategy. Propagation-based methods rely on the segmentation STM mask of the previous frame. It usually takes the previous mask as an input and learns a network to refine the mask to align it with the object in current frame. A representative work is MaskTrack [28]. This strategy of using the previous frame is also used in [2,42]. Many works [7,15,28] also use optical flow [9,14] in the propagation process. Matching-based methods [5,13,26,32,34,39] treat the first frame (or intermediate frames) as a template and match the pixel-level feature embeddings in the new frame with the templates. A more common setup is to use both the first frame and the previous frame [26,34,39], which covers both long-term and short-term object appearance information.

Space-Time Memory Network. The STM network [27] is a special feature matching-based method which performs dense feature matching across the entire spatio-temporal volume of the video. This is achieved by a novel space-time memory mechanism that stores the features at the each spatio-temporal location. The space-time memory module mainly contains two components and two operations. The two components are a key map and a value map where the keys encode the visual semantic embeddings for robust matching against appearance variations, and the values store detailed features for guiding the segmentation. The memory write operation simply concatenates the key and value maps generated on past frames and their object segmentation masks. When processing a new frame, the memory read operation uses the keys to match and find the relevant locations in the spatio-temporal volume in the video. Then the features stored in the value maps at those locations are retrieved to predict the current object mask.

3   Our Method

3.1   Global Context Module

The space-time memory network [27] achieves great success in video object segmentation with the space-time memory module. The STM module is an effective module that queries features from a set of spatio-temporal features encoding previous frames and their object masks to guide the processing of the current frame. In this way, the current frame is able to use the features from semantically related regions and the corresponding segmentation information in past frames to generate its features. However, the STM module has a drawback in efficiency in that it stores a pair of key and value vectors for each location of each frame in the memory. These feature vectors are simply concatenated over time when the system moves forward and their sizes keep increasing. This means its resource usage is highly sensitive to the spatial resolution and temporal length of the video. Consequently, the module is limited to have memories with low spatial resolutions, short temporal spans, or reduced memory saving frequency in practice. To remedy this drawback, we propose the global context (GC) module. Unlike the STM
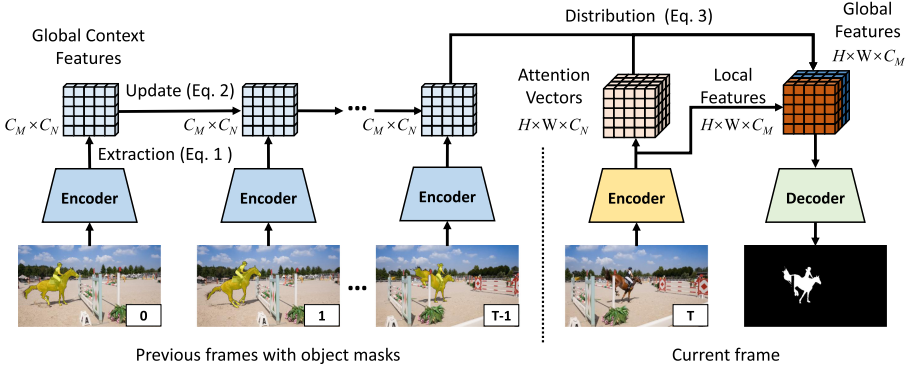
Fig. 2. This is an overview of our pipeline. Our network encodes past frames and their masks to a fixed-size set of global context features. These vectors are updated by a simple rule when the system moves to the next frame. At the current frame, a set of attention vectors are generated from the encoder to retrieve relevant information in the global context to form the global features. Local features are also generated from the encoder output. The global and local features are then concatenated and passed to a decoder to produce the segmentation result for this frame. Note that there are two types of encoders, the blue one for past frames and masks (four input channels), and the orange one for the current frame (three input channels).

module, the global context module keeps only a small set of fixed-size global context features yet retains almost the same representational power compared to the STM module. Figure 2 shows the overall pipeline of our proposed method using the global context module. The main structure is an encoder-decoder and the global context module is built on top of the encoder output, similarly to [27]. There are mainly two operations in our pipeline, namely 1) context extraction and update on a processed frame and its object mask, and 2) context distribution to the current frame under processing. There are two types of encoders used which produce features of $H \times W$ resolution and C channels. One takes a color image and an object mask (ground truth mask for the start frame and segmentation results for intermediate frames) to encode the frame and segmentation information into the feature space. Another encoder encodes the current frame to a feature embedding. We distribute the global context features stored in the global context module and combine the distributed context features with local features. Then, a decoder is used to produce the final object segmentation mask on this frame.

Context Extraction and Update. When extracting the global context features, the global context module first generates the keys and the values, following the same procedure as in the STM module. The keys and the values have size $H \times W \times C_N$ and $H \times W \times C_M$ respectively, where C_N and C_M are the numbers of channels used. Unlike the STM module that directly stores the keys and
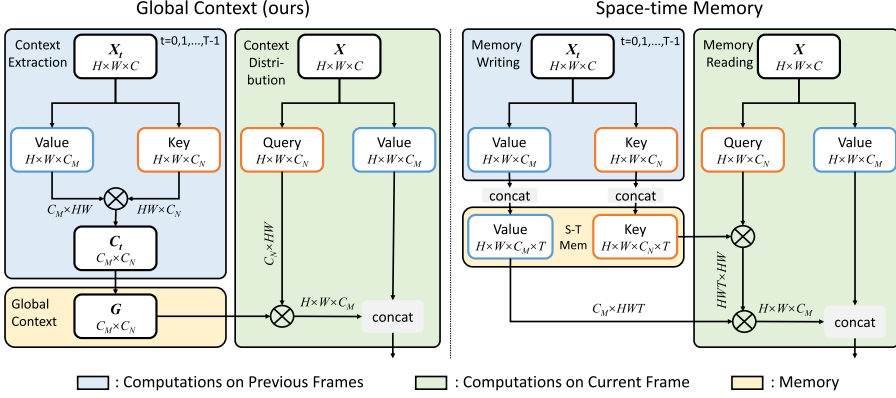
Fig. 3. The detailed implementation of the global context module with a comparison to the space-time memory module in [27].

values, the global context module puts the keys and the values through a further step called global summarization.

The global summarization step treats the keys not as $H \times W$ vectors of size C_M each for a location, but as C_M 1-channel feature maps each as an unnormalized weighting over all locations. For each such weighting, the global context module computes a weighted sum of the values where each value is weighted by the scalar at the corresponding location in the weighting. Each such weighted sum is called a global context vector. The global context module organizes all C_M global context vectors into a global context matrix as the output of global summarization. This step can be efficiently implemented as a matrix product between the transpose of the key matrix and the value matrix. Following is an equation describing the context extraction process of the global context module:

$$\boldsymbol{C}_t = k(\boldsymbol{X}_t)^{\mathsf{T}} v(\boldsymbol{X}_t), \tag{1}$$

where t is the index of the current frame, $\boldsymbol{C}_t$ is the context matrix of this frame, $\boldsymbol{X}_t$ is the input to the module (output of the encoder), and k, v are the functions that generate the keys and values. Having obtained $\boldsymbol{C}_t$, the rule for updating the global context feature is simply

$$\boldsymbol{G}_t = \frac{t-1}{t}\boldsymbol{G}_{t-1} + \frac{1}{t}\boldsymbol{C}_t, \tag{2}$$

where $\boldsymbol{G}_t$ is the global context for the first t frames with $\boldsymbol{G}_0$ being a zero matrix. The weight coefficients before the sum make that each $\boldsymbol{C}_p$ for $1 \le p \le t$ contributes equally to $\boldsymbol{G}_t$.

Context Distribution. At the current frame, our pipeline distributes relevant information stored in the global context module to each pixel. In the distribution process, the global context module first produces the query keys in the same way

as the spatce-time memory module. However, the global context module uses the queries differently. At each location, it interprets the query key as a set of weights over the global context features in the memory and computes the weighted sum of the features as the distributed context information. In contrast, the memory read process of the STM module is much more complex. The STM module uses the query to compute a similarity score with the key at each location of each frame in the memory. After that, the module computes the weighted sum of the values in the memory weighted by these similarity scores. Following is an formula that expresses this context distribution process:

$$\boldsymbol{D}_t = q(\boldsymbol{X}_t)\boldsymbol{G}_{t-1}, \tag{3}$$

where $\boldsymbol{D}_t$ is the distributed global features for frame t, and q is the function generating the queries.

Surprisingly, the simple context distribution process of the global context module and the much more complex process of the space-time memory module accomplish the same goal, which is to summarize semantic regions across all past frames that are of interest to the querying location in the current frame. The space-time memory module achieves this by first identifying such regions via query-key matching and then summarizing their values through a weighted sum. The global context module achieves the same much more efficiently since the global context vectors are already global summarizations of regions with similar semantics across all past frames. Therefore, the querying location only needs to determine on an appropriate weighting over the global context vectors to produce a vector that summarizes all the regions of interest to itself. For example, if a pixel is interested in persons, it could place large weights on global context vectors that summarize faces and bodies. Another pixel might be interested in the foreground and could put large weights on vectors that summarize various foreground object categories. We provide a mathematical proof that the global context and space-time memory modules have identical modeling power in the supplementary materials.

Comparison with the Space-Time Memory Module. We have plotted the detailed implementation of our global context module in Fig. 3 and compare with the space-time memory module used in [27]. There are a few places our global context has advantages in efficacy over space-time memory.

The global summarization process is the first way the global context module gains efficiency advantage over the space-time memory module. The global context matrix has size $C_M \times C_N$, which tends to be much smaller than the $H \times W \times C_N$ key and $H \times W \times C_M$ value matrices that the space-time memory module produces. The second way the global context module improves efficiency is that it adds the extracted context matrix to the stored context matrix, thereby keeping constant memory usage however many frames are processed. In contrast, the space-time memory module concatenates the obtained key and value matrices to the original key and value matrices in the memory, thus having a linear growth of memory with the number of frames.

For the computation on the current frame, i.e. the context distribution step, our global context module only needs to perform a light weight matrix product of size $C_M \times C_N$ and $C_N \times HW$ with $C_M C_N HW$ multiplications involved. In contrast, the last step of memory read for the STM module calculate a matrix product of size $HWT \times HW$ and $HW \times C_M$ ($C_M H^2 W^2 T$ multiplications), which has much larger memory usage and computational cost than the global context module and has linear complexities with respect to time T. To get a more intuitive comparison of the two, we calculate the computation and resources needed in this step when the input to the encoder is of size 384×384, $C_M = 512$, and $C_N = 128$ (the default setting in STM). The detailed numbers are listed in Table 1. It is noticeable that our global context module has great advantages over space-time memory in terms of both computation and memory cost, especially when the number of processed frames t becomes large along the processing.

Table 1. The complexity comparison of the memory read operation in space-time memory [27] and context distribution in our global context module. The memory usage is calculated using the float32 data type.

	t	FLOPS	Memory		t	FLOPS	Memory
STM	0	0.2 G	4 MB	GC (ours)	Any	**0.04 G**	**1 MB**
	10	2.1 G	40 MB				
	100	21.2 G	394 MB				

Table 2. Study on the size of global context feature ($C_M \times C_N$). This result is on DAVIS 2016 test set and the $\mathcal{J}\&\mathcal{F}$ is a segmentation accuracy metric (details in Sect. 4).

$C_M \times C_N$	$\mathcal{J}\&\mathcal{F}$	#Params	Time (s)	$C_M \times C_N$	$\mathcal{J}\&\mathcal{F}$	#Params	Time (s)
512×128	**86.6**	38 M	0.040	512×512	86.1	46 M	0.046

3.2   Implementation

Our encoder and decoder design is the same as STM [27]. We use ResNet-50 [12] as the backbone for both the context encoder and current encoder where the context encoder takes four-channel inputs and the current encoder takes three-channel inputs. The feature map at res4 is used to generate the key and value maps. After the context distribution operation, the features are compressed to 256 channels and fed into the decoder. The decoder takes this low-resolution input feature map and gradually upscales it by a scale factor of two each time. A skip connection is used to retrieve and fuse the feature map at the same resolution in the current encoder with the bilinearly upsampled feature map from

the previous stage of the decoder. The key and value generation (i.e. k, q, v in Eq. (1) and (3)) are implemented using 3×3 convolutions. In our implementation, we set $C_M = 512$ and $C_N = 128$. We have tested with larger feature sizes which introduce more complexities, but we do not observe accuracy gain in segmentation (see Table 2).

3.3   Training

Our training process mainly contains two stages. We first pre-train the network using simulated videos generated from static images. After that we fine-tune this pre-trained model on the video object segmentation datasets. We minimize the cross-entropy loss and train our model using the Adam [18] optimizer with a learning rate of 10^{-5}.

Pre-training on Images. We follow the successful practice in [26,27,39] that pre-trains the network using simulated video clips with frames generated by applying random transformation to static images. We use the images from the MSRA10K [8], ECSSD [41], and HKU-IS [19] datasets for the saliency detection task [3]. We found these datasets cover more object categories than those semantic segmentation or instance segmentation datasets [10,11,23]. This is more suitable for our purpose to build a general video object segmentation model. There are in total about 15000 images with object masks. A synthetic clip containing three frames is then generated using image transformations. We use random rotation $[-30°, 30°]$, scaling $[-0.75, 1.25]$, thin plate spline (TPS) warping (as in [28]), and random cropping for the video data simulation. We use 4 GPUs and set the batch size to be 8. We run the training for 100 epochs, and it takes one day to finish the pre-training.

Fine-Tuning on Videos. After training on the synthetic video data, we fine-tune our pre-trained model on video object segmentation datasets [29,30] at the 480p resolution. The learning rate is set to 10^{-6} and we run this fine-tuning for 30 epochs. Each training sample contains several temporally ordered frames sampled from the training video. We use random rotation, scaling, and random frame sampling interval in $[1,3]$ to gain more robustness to the appearance changes over a long time. We have tested different clip lengths (e.g. 3, 6, 9 frames) but did not observe performance gains on lengths greater than three. Therefore, we stick to three-frame clips. In the training, the network infers the object mask on the second frame and back propagates the error. Then, the soft mask from the network output is fed to the encoder to infer the mask on the third frame without thresholding as in [27].

4   Experimental Results

We evaluate our method and compare it with others on the DAVIS [29,30] and YouTube-VOS [40] benchmarks. The object segmentation mask evaluation

metrics used in our experiments are the average region similarity ($\mathcal{J}$ mean), the average contour accuracy ($\mathcal{F}$ mean), and the average of the two ($\mathcal{J}\&\mathcal{F}$ mean). DAVIS 2016 [29] is for single object segmentation. DAVIS 2017 [30] and YouTube-VOS [40] contain multiple object segmentation tasks.

Table 3. Quantitative comparison on DAVIS 2016 validation set. The results are sorted for online (OL) and non-online methods respectively according to $\mathcal{J}\&\mathcal{F}$ mean. The highest scores in each category are highlighted in bold.

Method	OL	Time (s)	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$ Mean	$\mathcal{F}$ Mean
OSVOS [4]	✓	7	80.2	79.8	80.6
Lucid [17]	✓	–	83.0	83.9	82.0
CINM [2]	✓	>30	84.2	83.4	85.0
OnAVOS [35]	✓	13	85.5	86.1	84.9
OSVOS-S [25]	✓	4.5	86.6	85.6	87.5
PReMVOS [24]	✓	>30	**86.8**	84.9	**88.6**
DyeNet [20]	✓	2.32	–	**86.2**	–
SiamMask [38]		0.03	70.0	71.7	67.8
OSMN [42]		0.13	73.5	74.0	72.9
PML [5]		0.28	77.4	75.5	79.3
VidMatch [13]		0.32	–	81.0	–
FAVOS [6]		1.8	81.0	82.4	79.5
FEELVOS [34]		0.5	81.7	80.3	83.1
RGMP [26]		0.13	81.8	81.5	82.0
AGAME [16]		0.07	81.9	81.5	82.2
RANet [39]		0.13	85.5	85.5	85.4
STM* [27]		0.15	86.5	84.8	**88.1**
GC (ours)		0.04	**86.6**	**87.6**	85.7

4.1 DAVIS 2016 (Single Object)

DAVIS 2016 [29] is a widely used benchmark dataset for single object segmentation in videos. It contains 50 videos among which 30 videos are for training and 20 are for validation. There are in total 3455 frames annotated with a single object mask for each frame. We use the official split for training and validation.

We list the quantitative results for representative works on DAVIS 2016 validation set in Table 3, including the most recent STM [27] and RANet [39]. To show their best performance, we directly quote the numbers posted on the benchmark website or in the papers. We can see that online-learning methods can get higher scores in most metrics. However, recent works of STM [27] and

RANet [39] demonstrate comparable results without online learning. Overall, the scores of our GC are among the top, including getting the highest $\mathcal{J}$ mean score.
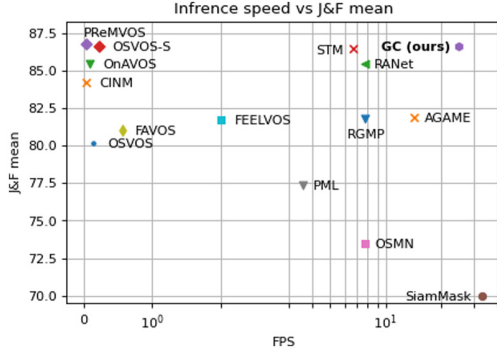

Fig. 4. The speed (FPS) *v.s.* accuracy ($\mathcal{J}\&\mathcal{F}$ mean) comparison on DAVIS 2016 validation set at 480p resolution.

Furthermore, for a more intuitive comparison, we plot the runtime in terms of average FPS and accuracy in terms of $\mathcal{J}\&\mathcal{F}$ mean for different methods in Fig. 4. We test the speed on one Tesla P40. It can be seen that although the online-learning methods (*e.g.* [2,24,25]) can produce highly accurate results, their speeds are extremely slow due to the time consuming online learning process. The methods without online learning (*e.g.* [5,26,42]) are fast but have lower accuracy. The most recent works STM [27] and RANet [39] can get segmentation accuracy comparable to online-learning method while maintaining faster speed (STM [27] 6.7 fps, RANet [39] 8.3 fps for 480p frames[1]). Our GC boosts the speed further (25.0 fps) and still maintains high accuracy. Note that the videos in DAVIS datasets are all short (<100 frames). If running on longer videos, our speed advantage over STM [27] will become more remarkable as STM has a linear time complexity with respect to the video length. While SiamMask [38] is the only method faster than our GC, its accuracy is very unsatisfactory compared to other methods. This demonstrates our GC is both fast and accurate which makes it a practical solution to the video object segmentation problem.

4.2 DAVIS 2017 (Multiple Objects)

DAVIS 2017 is an extension of DAVIS 2016 that contains videos with multiple objects annotated per frame. It has 60 videos for training and 30 videos for testing. We do not use any additional module for multi-object segmentation which [27,39] used, but simply treat each object individually. We still train the

---

[1] RANet [39] reported a faster runtime in the paper using half-precision computation which is disabled in our test for fair comparison.

network to produce a binary mask for the object. In testing, we use the network to get the soft probability map for each object separately and use a softmax operation as post-processing on the maps for all objects in the frame to produce the multi-label segmentation mask.

Table 4 summarizes the performance of existing methods and compare them with ours on DAVIS 2017 dataset. The multi-object scenarios is more challenging than the single object ones due to the interactions and occlusions among multiple objects. It can be seen that again online-learning methods, *e.g.* [2,24], get decent scores in all metrics, but have longer runtime. For non-online methods, STM [27] ranks the highest overall. Our model can get almost identical performance with STM [27] but with faster speed and much less memory consumption.

4.3   YouTube-VOS

YouTube-VOS [40] is a large-scale dataset for multiple object segmentation in videos. Its training set contains 4453 annotated videos and validation set contains 474 videos. Table 5 compares the performance of different methods on this dataset. Note that there are unseen object categories in the validation set. The unseen objects are tested separately to measure the generalization power of each method. It can be seen that STM [27] gets remarkable high scores. Our GC is among the top performance tier. Further, the results of our method do not show large performance difference between seen and unseen objects.

Table 4. The quantitative comparison on DAVIS 2017 validation set. The results are sorted for online (OL) and non-online methods respectively according to $\mathcal{J}\&\mathcal{F}$ Mean. The highest scores in each category are highlighted in bold.

Method	OL	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$ Mean	$\mathcal{F}$ Mean
OSVOS [4]	✓	60.3	56.6	63.9
OnAVOS [35]	✓	65.4	61.6	69.1
OSVOS-S [25]	✓	68.0	64.7	71.3
CINM [2]	✓	70.6	67.2	74.0
PReMVOS [24]	✓	**77.8**	**73.9**	**81.8**
OSMN [42]		54.8	52.5	57.1
SiamMask [38]		56.4	54.3	58.5
FAVOS [6]		58.2	54.6	61.8
VidMatch [13]		–	56.5	–
RANet [39]		65.7	63.2	68.2
RGMP [26]		66.7	64.8	68.6
FEELVOS [34]		69.1	65.9	72.3
AGAME [16]		70.0	67.2	72.7
STM* [27]		**71.6**	69.2	**74.0**
GC (ours)		71.4	**69.3**	73.5

Table 5. The quantitative comparison on YouTube-VOS [40] validation set. The results for other methods are quoted from [22,27].

	RVOS [33]	OSVOS [4]	S2L(OL) [40]	PreMVOS [24]	AGAME [16]	BoLTVOS [36]	AGSS [22]	STM [27]	**GC (ours)**
Overall	56.8	58.8	64.4	66.9	66.1	71.1	71.3	79.4	73.2
$\mathcal{J}$ seen	63.6	59.8	71.0	71.4	67.8	71.6	71.3	79.7	72.6
$\mathcal{J}$ unseen	45.5	54.2	55.5	56.5	60.8	64.3	65.5	72.8	68.9
$\mathcal{F}$ seen	67.2	60.5	70.0	75.9	–	–	76.2	84.2	75.6
$\mathcal{F}$ unseen	51.0	60.7	61.2	63.7	–	–	73.1	80.9	75.7
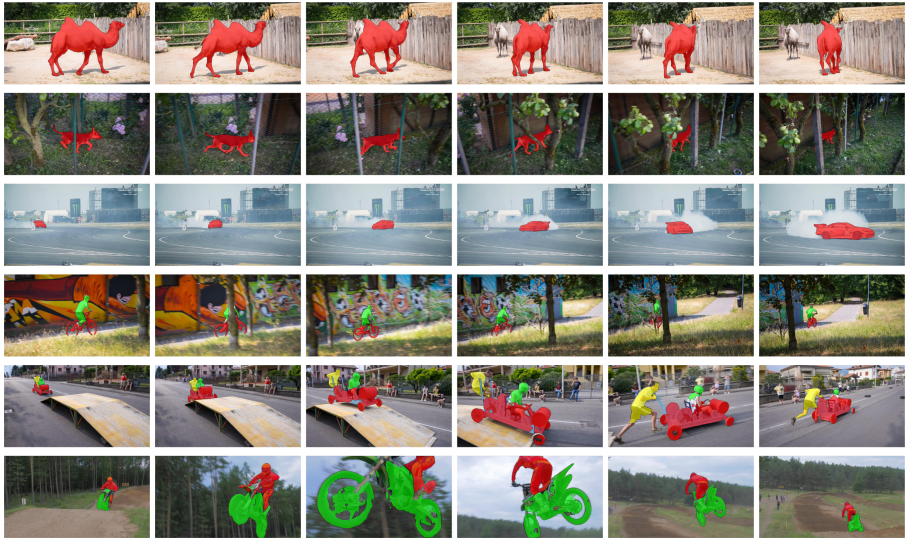
Fig. 5. The visual results of video object segmentation using our global context module.

Notably, the performance gap between STM and our GC does not come from the models. Instead, two external factors are the main cause of the gap:

– an easier testing protocol used by STM;
– a soft-aggregation post-processing module, which is compatible with GC but not implemented due to time constraints.

Table 6 summarized the comparison.

Table 6. Cause of performance gap between STM and our GC on YouTube-VOS.

Model	Soft aggregation	Test input stride	$\mathcal{J}\&\mathcal{F}$ mean
GC		5	73.2
STM		5	73.7
STM	✓	1	79.4

4.4   Qualitative Results

Figure 5 shows visual examples of our segmentation results. As can be seen in the figure, our global context module can effectively handle many challenging cases such as appearance changes (row 1), size changes (row 3, 5, and 6), and occlusions (row 2 and 4).
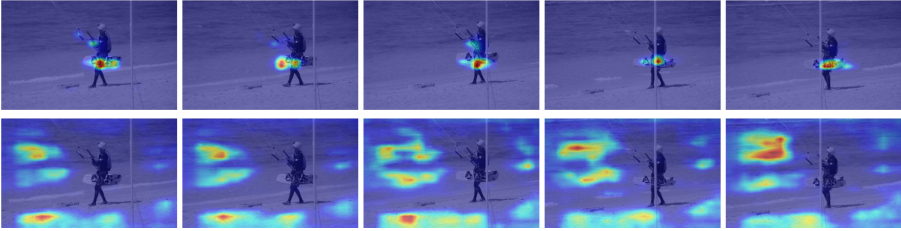

Fig. 6. Visualization of the global context keys.

4.5   Visualization of the Global Context Module

Figure 6 plots visualization of the global context module. As described in Sect. 3.1, each channel of the global context key (the keys in the blue region in Fig. 3 left) is an attention map (or weight map) over all spatial locations. Our global context module aggregates the features at these locations to form one global context feature vector by a weighted sum. After such aggregation on all C_N channels, C_N feature vectors are generated. Figure 6 shows the visualization of two channels in the global context key at evenly sampled time in a whole video sequence. We can clearly see that it can summarize the segmentation information well, where the keys in the upper row capture parts of the foreground object and the keys in the bottom capture the background. In addition, it shows that the global context module can capture the foreground and background information consistently throughout the video.

5   Conclusion

We have presented a practical solution to the problem of semi-supervised video object segmentation. It is achieved by a novel global context module that effectively and efficiently captures the object segmentation information in all processed frames with a set of fixed-size features. The evaluation on multiple benchmark datasets shows that our method gets top performance, especially on the single object DAVIS 2016 dataset, and runs at a much faster speed than all top-performing methods, including the state-of-the-art STM. Our global context module is also efficient in memory usage and will not have memory issues as with STM for longer video sequences. We believe that our global context module has the potential to become a core module in practical video object segmentation tools. In the future, we want to optimize it further to make it suitable

for running on portable devices like tablets and mobile phones. Applying the global context module to other video-related computer vision problems is also of our interest.

References

1. Bai, X., Wang, J., Simons, D.S., Sapiro, G.: Video SnapCut: robust video object cutout using localized classifiers. ACM Trans. Graph. **28**, 70 (2009)
2. 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)
3. Borji, A., Cheng, M.M., Jiang, H., Li, J.: Salient object detection: a benchmark. IEEE Trans. Image Process. **24**(12), 5706–5722 (2015)
4. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: CVPR (2017)
5. Chen, Y., Pont-Tuset, J., Montes, A., Van Gool, L.: Blazingly fast video object segmentation with pixel-wise metric learning. In: CVPR (2018)
6. 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)
7. Cheng, J., Tsai, Y.H., Wang, S., Yang, M.H.: SegFlow: joint learning for video object segmentation and optical flow. In: ICCV (2017)
8. Cheng, M.M., Mitra, N.J., Huang, X., Torr, P.H., Hu, S.M.: Global contrast based salient region detection. TPAMI **37**(3), 569–582 (2014)
9. Dosovitskiy, A., et al.: FlowNet: learning optical flow with convolutional networks. In: ICCV (2015)
10. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. IJCV **88**(2), 303–338 (2010)
11. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV (2011)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
13. 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
14. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: evolution of optical flow estimation with deep networks. In: CVPR (2017)
15. Jang, W.D., Kim, C.S.: Online video object segmentation via convolutional trident network. In: CVPR (2017)
16. 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)
17. Khoreva, A., Benenson, R., Ilg, E., Brox, T., Schiele, B.: Lucid data dreaming for video object segmentation. IJCV **127**(9), 1175–1197 (2019)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2014)
19. Li, G., Yu, Y.: Visual saliency based on multiscale deep features. In: CVPR (2015)
20. Li, X., Loy, C.C.: Video object segmentation with joint re-identification and attention-aware mask propagation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 93–110. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_6
21. Li, Y., Sun, J., Shum, H.Y.: Video object cut and paste. In: SIGGRAPH (2005)

22. Lin, H., Qi, X., Jia, J.: AGSS-VOS: attention guided single-shot video object segmentation. In: ICCV (2019)
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. Luiten, J., Voigtlaender, P., Leibe, B.: PReMVOS: proposal-generation, refinement and merging for video object segmentation. In: ACCV (2018)
25. Maninis, K.K., et al.: Video object segmentation without temporal information. TPAMI **41**(6), 1515–1530 (2018)
26. Oh, S.W., Lee, J.Y., Sunkavalli, K., Kim, S.J.: Fast video object segmentation by reference-guided mask propagation. In: CVPR (2018)
27. Oh, S.W., Lee, J.Y., Xu, N., Kim, S.J.: Video object segmentation using space-time memory networks. In: ICCV (2019)
28. Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., Sorkine-Hornung, A.: Learning video object segmentation from static images. In: CVPR (2017)
29. 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)
30. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 davis challenge on video object segmentation. arXiv:1704.00675 (2017)
31. Price, B.L., Morse, B.S., Cohen, S.: LIVEcut: learning-based interactive video segmentation by evaluation of multiple propagated cues. In: ICCV (2009)
32. Shin Yoon, J., Rameau, F., Kim, J., Lee, S., Shin, S., So Kweon, I.: Pixel-level matching for video object segmentation using convolutional neural networks. In: ICCV (2017)
33. 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)
34. 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)
35. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: BMVC (2017)
36. Voigtlaender, P., Luiten, J., Leibe, B.: BoLTVOS: box-level tracking for video object segmentation. arXiv preprint arXiv:1904.04552 (2019)
37. Wang, J., Bhat, P., Colburn, A., Agrawala, M., Cohen, M.F.: Interactive video cutout. ACM Trans. Graph. **24**, 585–594 (2005)
38. Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P.H.: Fast online object tracking and segmentation: a unifying approach. In: CVPR (2019)
39. Wang, Z., Xu, J., Liu, L., Zhu, F., Shao, L.: RANet: ranking attention network for fast video object segmentation. In: ICCV (2019)
40. 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
41. Yan, Q., Xu, L., Shi, J., Jia, J.: Hierarchical saliency detection. In: CVPR (2013)
42. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. In: CVPR (2018)