VideoMatch: Matching Based Video Object Segmentation

Yuan-Ting Hu[1(✉)], Jia-Bin Huang[2], and Alexander G. Schwing[1]

[1] University of Illinois at Urbana-Champaign, Champaign, USA
{ythu2,aschwing}@illinois.edu
[2] Virginia Tech, Blacksburg, USA
jbhuang@vt.edu

Abstract. Video object segmentation is challenging yet important in a wide variety of applications for video analysis. Recent works formulate video object segmentation as a prediction task using deep nets to achieve appealing state-of-the-art performance. Due to the formulation as a prediction task, most of these methods require fine-tuning during test time, such that the deep nets memorize the appearance of the objects of interest in the given video. However, fine-tuning is time-consuming and computationally expensive, hence the algorithms are far from real time. To address this issue, we develop a novel matching based algorithm for video object segmentation. In contrast to memorization based classification techniques, the proposed approach learns to match extracted features to a provided template without memorizing the appearance of the objects. We validate the effectiveness and the robustness of the proposed method on the challenging DAVIS-16, DAVIS-17, Youtube-Objects and JumpCut datasets. Extensive results show that our method achieves comparable performance without fine-tuning and is much more favorable in terms of computational time.

1 Introduction

Video segmentation plays a pivotal role in a wide variety of applications ranging from object identification, video editing to video compression. Despite the fact that delineation and tracking of objects are seemingly trivial for humans in many cases, video object segmentation remains challenging for algorithms due to occlusions, fast motion, motion blur, and significant appearance variation over time.

Research efforts developing effective techniques for video object segmentation continue to grow, partly because of the recent release of high-quality datasets, *e.g.*, the DAVIS dataset [40,42]. Two of the main setups for video object segmentation are the unsupervised and the semi-supervised setting [40,42]. Both

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01237-3_4) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11212, pp. 56–73, 2018.
https://doi.org/10.1007/978-3-030-01237-3_4

cases are analogous in that the semantic class of the objects to be segmented during testing are not known ahead of time. Both cases differ in the supervisory signal that is available at test time. While no supervisory signal is available during testing in the unsupervised setting, the ground truth segmentation mask of the first frame is assumed to be known in the semi-supervised case. With video editing applications in mind, here, we focus on the *semi-supervised setting*, *i.e.*, our goal is to delineate in all frames of the video the object of interest which is specified in the first frame of the video.

Taking advantage of the provided groundtruth for the first frame, existing semi-supervised video object segmentation techniques follow deep learning based methods [4,5,25,26,47,48,53,59] and fine-tune a pre-trained classifier on the given ground truth in the first frame during online testing [4,5,23,25,26,53]. This online fine-tuning of a classifier during testing has been shown to improve accuracy significantly. However, fine-tuning during testing is necessary for each object of interest given in the first frame, takes a significant amount of time, and requires specialized hardware in the form of a very recent GPU due to the memory needs of back-propagation for fine-tuning.

In contrast, in this paper, we propose a novel end-to-end trainable approach for fast semi-supervised video object segmentation that does not require any fine-tuning. Our approach is based on the intuition that features of the foreground and background in any frame should match features of the foreground and background in the first frame. To ensure that the proposed approach can cope with appearance and geometry changes, we use a deep net to learn the features that should match and adapt the sets of features as inference progresses.

Our method yields competitive results while saving computational time and memory when compared to the current state-of-the-art approaches. On the recently released DAVIS-16 dataset [40], our algorithm achieves 81.03% in IoU (intersection over union) while reducing the running time by one order of magnitude compared to the state-of-the-art, requiring on average only 0.32 s per frame.

2 Related Work

Video object segmentation has been extensively studied in the past [5,15,22,25,29–31,36,39,44,49,50,55]. In the following, we first discuss the related literature, (1) focusing on semi-supervised video object segmentation, and (2) discussing unsupervised video object segmentation. Subsequently, we examine the relationship of our work and the tracking and matching literature.

Semi-supervised Video Object Segmentation: Semi-supervised video object segmentation assumes that the groundtruth of the first frame is available during testing. Many approaches in this category employ fine-tuning during testing in order to achieve better performance [4,5,8,19,23,25,26,32,53]. It has been shown that fine-tuning on the first frame significantly improves accuracy. However, the fine-tuning step is computationally demanding, adding more than 700 s per video to test time [5].

Additional cues such as optical flow [8,25,26,32], semantic segmentation [4, 26] and re-identification modules [32] can be integrated into the framework to further improve the accuracy. Since fine-tuning is still required, those cues increase the computational needs.

Among the semi-supervised video object segmentation methods, the approach by Yoon *et al.* [59] is most related to our approach. Yoon *et al.* [59] also address video object segmentation by pixel matching. Their approach concatenates the features extracted from the template and the input images, and uses fully connected layers to simulate matching between the two images. Importantly, the approach still requires fine-tuning. In addition, the fully connected layers restrict the method to process frames at a specific, pre-defined spatial resolution.

Concurrent to our work, several recent methods (all developed independently) have been proposed to improve the speed of video object segmentation through part-based tracking [9], pixel-wise metric learning [7], or network modulation [38,56]. We refer the readers to these works for a more complete picture.

Unsupervised Video Object Segmentation: Neither groundtruth nor user annotation is available in the unsupervised video object segmentation setting. Therefore, the unsupervised setup requires algorithms to automatically discover the salient objects in video. Different methods such as motion analysis [39], trajectory clustering [37], and saliency-based spatio-temporal propagation [12, 20] have been proposed to identify the foreground objects. More recently, deep net based approaches have been discussed [22,47,48].

Object Tracking: Semi-supervised video object segmentation and object tracking [28,58] are related to our approach as they both keep track of the objects through the entire video. However, the two tasks differ in the format of the output. The output of video object segmentation is a pixel-level segmentation mask while the output of object tracking is a bounding box that delineates the position and scale of the object. From the tracking literature, work by Bertinetto *et al.* [3] is in a spirit similar to our proposed approach as they formulate tracking by matching. However, due to the difference in the output, Bertinetto *et al.* [3] calculated correlation by convolving the whole patch with the given template, while we propose a soft matching for pixel-wise segmentation.

Matching: Image matching [18,33] has been extensively studied over the last few decades. With the success of deep learning, research focus moved from matching using handcrafted features [35] to deep features [57]. Correlation between the extracted feature maps is typically computed to find correspondences [45], to estimate optical flow fields [10] and geometric transformations [46]. Since the objective of matching is to find point-to-point correspondences, the result will be noisy if the matching algorithm is directly applied to segmentation. To deal with the noisy prediction, we proposed a soft matching mechanism which estimates the similarity score between different segments as discussed next.

3 Matching Based Video Object Segmentation

In the following, we describe details of the proposed algorithm for video object segmentation. We first formally define the problem setting and provide an overview of our approach in Sect. 3.1. We then detail the new proposed soft matching mechanism in Sect. 3.2. Subsequently, we show in Sect. 3.3 how our model accommodates appearance changes of objects over time during online testing without the need for finetuning. Finally, we demonstrate how to easily extend our method to instance-level video object segmentation in Sect. 3.4.

3.1 Overview

Given a sequence of T video frames $\{I_1, \ldots, I_T\}$ and the groundtruth segmentation $y_1^* \in \{1, \ldots, N\}^{W \times H}$ of the first frame I_1, the task of semi-supervised video object segmentation is to predict the segmentation masks of the subsequent video frames $I_2, \ldots, I_T$, denoted as $y_2, \ldots, y_T \in \{1, \ldots, N\}^{W \times H}$. Hereby, N is the number of objects of interest in the given video. We denote width and height of the frames as W and H. We start by discussing the single instance case ($N = 1$) and explain how to extend the proposed method to $N > 1$ in Sect. 3.4. Importantly, we emphasize that semi-supervised video object segmentation requires object independent formulations since we do not know ahead of time the semantic class of the object to be segmented.

As the object category and appearance are unknown before test time, a network detecting objectness is usually trained offline. During test time a natural way is to use the given groundtruth for the first frame, *i.e.* y_1^*, as training data to fine-tune the pretrained objectness network [4,5,23,25,26,53]. Fine-tuning encourages the network to memorize appearance of the object of interest. In previous works on instance-level segmentation, memorization is achieved by fine-tuning a pretrained network N times, *i.e.*, to obtain one fine-tuned network for each object. As discussed before, although this fine-tuning step is the key to improving performance, it introduces a significant amount of processing time overhead and consumes more memory during testing even when there is only one object of interest in the video.

Our idea for efficient video object segmentation is to develop a network which is general enough such that the fine-tuning step can be omitted. To this end, we propose to match features obtained from the test frame I_t to features of the groundtruth foreground and background in the first frame I_1 (template). We designed an end-to-end trainable deep neural net, not only to extract features from video frames, but also to match two sets of features.

To achieve this goal, as shown in Fig. 1, we use a Siamese architecture that employs a convolutional neural network to compute the two feature maps. We use $\mathbf{x}_1 \in \mathbb{R}^{h \times w \times c}$ and $\mathbf{x}_t \in \mathbb{R}^{h \times w \times c}$ to refer to feature tensors extracted from the first frame (template) I_1 and the test frame I_t, respectively. The feature tensors $\mathbf{x}_1$ and $\mathbf{x}_t$ are of size $h \times w \times c$, where c is the number of the feature channels and w, h are the width and height of the feature maps, proportional to the $W \times H$

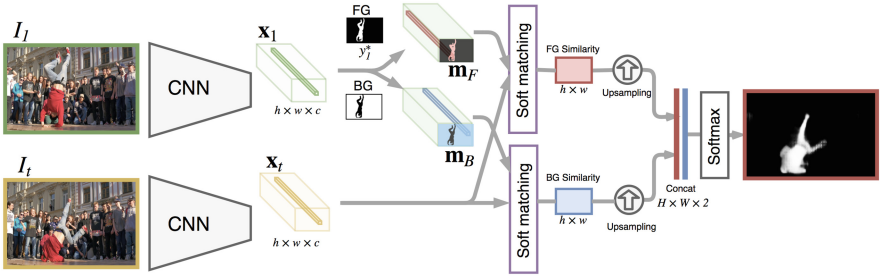


Fig. 1. Overview of the proposed video object segmentation algorithm. We use the provided ground truth mask of the first frame to obtain the set of foreground and background features ($\mathbf{m}_F$ and $\mathbf{m}_B$). After extracting the feature tensor $\mathbf{x}_t$ from the current frame, we use the proposed soft matching layer to produce FG and BG similarity. We then concatenates the two similarity scores and generate the final prediction via softmax.

sized video frame. The ratio between W and w depends on the downsampling rate of the convolutional neural net.

Next we define a set of features for the foreground and the background. We refer to those sets via $\mathbf{m}_F$ and $\mathbf{m}_B$ respectively. To formally define those sets of features, let $\mathbf{x}_t^i$ denote the c-dimensional vector representing the feature at pixel location i in the downsampled image. Given the groundtruth template y_1^* for the first frame, we collect the foreground features $\mathbf{m}_F$ and background features $\mathbf{m}_B$ for this first frame via

$$\mathbf{m}_F = \{\mathbf{x}_1^i : i \in g(y_1^*)\} \qquad \text{and} \qquad \mathbf{m}_B = \{\mathbf{x}_1^i : i \notin g(y_1^*)\}.$$

Hereby $g(y_1^*)$ is the set of pixels that belongs to foreground as indicated by the ground truth mask y_1^* downsampled to size $w \times h$.

After having extracted the foreground ($\mathbf{m}_F$) and background ($\mathbf{m}_B$) features from the template and after having computed features $\mathbf{x}_t \in \mathbb{R}^{h \times w \times c}$ from frame I_t using the same deep net, we match $\mathbf{x}_t^i$ $\forall i \in \{1, \ldots, wh\}$ to features collected in both sets $\mathbf{m}_F$ and $\mathbf{m}_B$ via a soft matching layer. The result of the soft matching layer for each pixel i is its foreground and background matching scores. Subsequently, the foreground and background matching scores are upsampled and normalized into a predicted foreground probability y_t via the softmax operation. We visualize this process in Fig. 1 and describe the proposed soft matching layer subsequently in greater detail.

3.2 Soft Matching Layer

A schematic illustrating the details of the proposed soft matching layer is given in Fig. 2. The developed soft matching layer, $\text{SML}(\mathbf{x}_t, \mathbf{m})$, takes two sets of features as inputs, *i.e.*, $\mathbf{x}_t$ and $\mathbf{m}$ ($\mathbf{m}$ refers to either $\mathbf{m}_F$ or $\mathbf{m}_B$) and computes a matching score matrix $S_t \in \mathbb{R}^{h \times w}$ which measures the compatibility of the frame

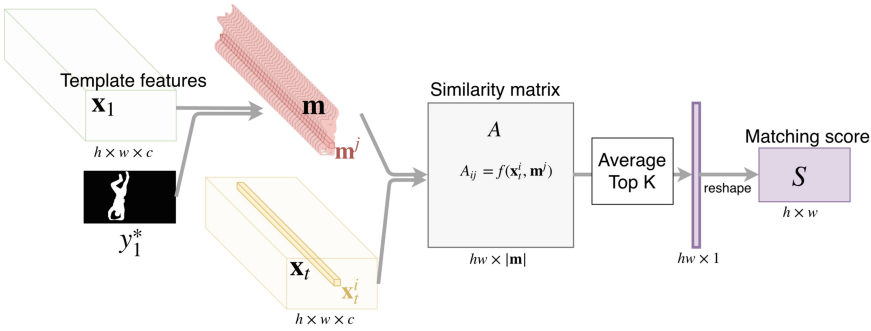


Fig. 2. Illustration of the proposed soft matching layer. We first take two sets of features and compute pairwise similarity between all pairs of features. We then produce the final matching score by computing the average of top K similarity scores.

I_t (represented by its features $\mathbf{x}_t$) with either foreground ($\mathbf{m}_F$) or background ($\mathbf{m}_B$) pixels of the template I_1 for every pixel $i \in \{1, \ldots, hw\}$. The entry S_t^i represents the similarity of the feature at pixel location i with respect to a subset of features in $\mathbf{m}$.

More formally, our developed soft matching layer first computes the pairwise similarity score matrix $A \in [-1, 1]^{(hw)\times|\mathbf{m}|}$ where the ij-th entry of A is calculated via

$$A_{ij} = f(\mathbf{x}_t^i, \mathbf{m}^j).$$

Hereby, f is a scoring function measuring the similarity between the two feature vectors $\mathbf{x}_t^i$ and $\mathbf{m}^j$. We use the cosine similarity, *i.e.*, $f(\mathbf{x}_t^i, \mathbf{m}^j) = \frac{\mathbf{x}_t^j \cdot \mathbf{m}^j}{\|\mathbf{x}_t^j\| \|\mathbf{m}^j\|}$, but any other distance metric is equally applicable once adequately normalized.

Given the similarity score matrix A, we compute the matching score matrix S_t of size $h \times w$, respectively its i-th entry ($i \in \{1, \ldots, hw\}$) via

$$S_t^i = \frac{1}{K} \sum_{j \in \mathrm{Top}(A_i, K)} A_{ij},$$

where the set $\mathrm{Top}(A_i, K)$ contains the indices with the top K similarity scores in the i-th row of the similarity score matrix A. K is set to 20 in all our experiments.

Intuitively, we use the average similarity of the top K matches because we assume a pixel to match to a number of pixels in a region as opposed to only one pixel, which will be too noisy, or to all pixels, which will be too strict in general as the foreground or background may be rather diverse. Consequently, we expect a particular pixel to match to one of the foreground or background regions rather than requiring a pixel only to match locally or to all regions. Again, an illustration of the soft matching layer, $\mathrm{SML}(\mathbf{x}_t, \mathbf{m})$, is presented in Fig. 2.

(a) FG pred. $y_{t,\text{init}}$ (b) FG pred. y_{t-1} (c) Extruded pred. $\hat{y}_{t-1}$ (d) Output pred. y_t

Fig. 3. Example of the proposed outlier removal process. We first extrude the prediction from the previous frame (b) to obtain an extruded prediction (c). We then produce the prediction at the current frame by finding the intersection between (a) and (c).

3.3 Outlier Removal and Online Update

Outlier Removal. To obtain the final prediction y_t for frame $t \in \{2, \ldots, T\}$ we convert the foreground and background matching score matrices into an initial foreground probability prediction $y_{t,\text{init}}$ via upsampling and via a subsequent weighted softmax operation. Finally, we obtain the prediction y_t by comparing the initial prediction $y_{t,\text{init}}$ with y_{t-1} to remove outliers. More specifically, we first extrude the prediction y_{t-1} of the previous frame to find pixels whose distance to the segmentation is less than a threshold d_c. We then compute y_t from $y_{t,\text{init}}$ by removing all initial foreground predictions that don't overlap with the extruded prediction $\hat{y}_{t-1}$. Note that the hat symbol '$\hat{\cdot}$' refers to the extrusion operation. This process assumes that the change of the object of interest is bounded from above. In Fig. 3, we visualize one example of the current foreground prediction $y_{t,\text{init}}$, previous foreground prediction y_{t-1}, the extruded prediction $\hat{y}_{t-1}$, and the final foreground prediction y_t.

Online Update. Obviously, we expect the appearance of the object of interest to change over time in a given video. In order to accommodate the appearance change, we repeatedly adjust the foreground and background model during testing. Inspired by [53], we update the foreground and background sets of features, *i.e.*, $\mathbf{m}_F$ and $\mathbf{m}_B$, by appending additional features after we predicted the segmentation for each frame. We find the additional features by comparing the initial prediction mask $y_{t,\text{init}}$ for $t \in \{2, \ldots, T\}$ with the extruded prediction $\hat{y}_{t-1}$ of the previous frame.

Specifically, we update the background model $\mathbf{m}_B$ at time t via

$$\mathbf{m}_B \leftarrow \mathbf{m}_B \cup \{\mathbf{x}_t^i : i \in \mathbf{b}_t\},$$

where the index set

$$\mathbf{b}_t = \{i : i \in g(y_{t,\text{init}}), i \notin g(\hat{y}_{t-1})\} = \{i : i \in g(y_{t,\text{init}}) \setminus g(y_t)\}$$

subsumes the set of pixels that are predicted as foreground initially, *i.e.*, in $y_{t,\text{init}}$, yet don't belong to the set of foreground pixels in the extruded previous prediction $\hat{y}_{t-1}$. Note that this is equivalent to the set of pixels which are predicted as foreground initially, *i.e.*, $y_{t,\text{init}}$, but are not part of the final prediction y_t.

Taking Fig. 3 as an example, $\mathbf{b}_t$ contains the indices of pixels being foreground in Fig. 3(a) but not in Fig. 3(b).

Intuitively, we find the possible outliers in the current predictions if a pixel is predicted as foreground at time t but does not appear to be foreground or is near to the foreground mask at time $t-1$.

Beyond adjusting the background model we also update the foreground model $\mathbf{m}_F$ via

$$\mathbf{m}_F \leftarrow \mathbf{m}_F \cup \{\mathbf{x}_t^i : i \in g(\breve{y}_t), y_t^i > c, i \notin \mathbf{b}_t\},$$

where $g(\breve{y}_t)$ is the set of foreground pixels in the eroded current segmentation prediction y_t and c is a constant threshold. Intuitively, we add the features of pixels that are not only predicted as foreground with high confidence (larger than c_1) but are also far from the boundary. In addition, we exclude those pixels in $\mathbf{b}_t$ to avoid conflicts between the foreground and background features.

Since our method just appends additional representations to the foreground and background features $\mathbf{m}_F$ and $\mathbf{m}_B$, the parameters of the employed network remain fixed, and the online update step is fast. Compared to [53], where each online update requires fine-tuning the network on the tested images, our approach is more efficient. Note that we designed a careful process to select features which are added in order to avoid the situation that the sizes of $\mathbf{m}_F$ and $\mathbf{m}_B$ grow intractably large, which will slow down the computation when computing the matching scores. It is obviously possible to keep track of how frequently features appear in the Top$-K$ set and remove those that don't contribute much. In practice, we didn't find this to be necessary for the employed datasets.

3.4 Instance-Level Video Object Segmentation

Next, we explain how the proposed method can be generalized for instance-level video object segmentation, where one or more objects of interest are presented in the first frame of the video. We consider the case where the ground truth segmentation mask contains a single or multiple objects, *i.e.*, $y_1^* \in \{1, \ldots, N\}^{H \times W}$, where $N \geq 1$. We construct the foreground and background features for every object, *i.e.*, we find the foreground features $\mathbf{m}_{F,k}$ and the background features $\mathbf{m}_{B,k}$ of the object $k \in \{1, \ldots, N\}$, where

$$\mathbf{m}_{F,k} = \{\mathbf{x}_1^i : i \in g(\delta(y_1^* = k))\} \qquad \text{and} \qquad \mathbf{m}_{B,k} = \{\mathbf{x}_1^i : i \notin g(\delta(y_1^* = k))\}.$$

Hereby, $\delta(\cdot) : \{1, \ldots, N\}^{H \times W} \rightarrow \{0, 1\}^{H \times W}$ is the indicator function which provides a binary output indicating the regions in y_1^* that belong to the k-th object. We then compute $y_{t,k}$, the foreground probability map of the frame t w.r.t. the k-th object by considering $\mathbf{x}_t$, $\mathbf{m}_{F,k}$ and $\mathbf{m}_{B,k}$ using the soft matching layer described above. After having computed k probability maps, we fuse them to obtain the final output prediction. The prediction y_t is computed by finding the index of the object that has maximum probability $y_{t,k}^i$ among all $k \in \{1, \ldots, N\}$ for all pixels i. If for all k, $y_{t,k}^i$ is less than a threshold c_2, the pixel i will be classified as background.

4 Experimental Results

In the following we first provide implementation details before evaluating the proposed approach on a variety of datasets using a variety of metrics.

4.1 Implementation Details, Training and Evaluation

To obtain the features $\mathbf{x}$, we found ResNet-101 [17] as the backbone with dilated convolutions [6] to perform well. More specifically, we use the representation from the top convolutional layer in the network as $\mathbf{x}_t$. The feature maps have spatial resolution 8 times smaller than the input image. In the experiments, we set $K = 20$, $d_c = 100$, $c_1 = 0.95$ and $c_2 = 0.4$. We initialized the parameters using the model pretrained on Pascal VOC [11,16] for semantic image segmentation. We trained the entire network end-to-end using the Adam optimizer [27]. We set the initial learning rate to 10^{-5} and gradually decreases over time. The weight decay factor is 0.0005.

To training our matching network, we use any two randomly chosen frames in a video sequence as training pairs. Importantly, the two frames are not required to be consecutive in time which provides an abundance of training data. We augmented the training data by random flipping, cropping and scaling between a factor of 0.5 to 1.5. We use Tensorflow to implement the algorithm. Training takes around 4 h for 1000 iterations on an Nvidia Titan X. At test time, a forward pass with an input image of size 480×854 takes around 0.17 s.

Training: We trained the proposed network using the 30 video sequences available in the DAVIS-16 training set [40] for 1000 iterations and evaluated on the DAVIS-16 validation set. Similarly, we used the 60 sequences in the DAVIS-17 training set [42] for training when testing on the DAVIS-17 validation set. Although the model is trained on DAVIS, we found it to generalize well to other datasets. Therefore, we use the model trained on the DAVIS-17 training set for evaluation on both the JumpCut [13] and the YouTube-Objects [43] datasets.

Evaluation: We validate the effectiveness of our method on the DAVIS-16 [40] validation, the DAVIS-17 [42] validation, the JumpCut [13] and the YouTube-Objects [43] datasets. For the YouTube-Objects dataset, we use the subset with groundtruth segmentation masks provided by [21], containing 126 video sequences. All of the datasets provide pixel-level groundtruth segmentation. More specifically, binary (foreground-background) ground truth is provided in the DAVIS-16, JumpCut, and YouTube-Objects datasets, while there is instance-level segmentation groundtruth available for the DAVIS-17 dataset. Challenges such as occlusion, fast motion, and appearance change are presented in the four datasets. Thus, these four datasets serve as a good test bed to evaluate different video object segmentation techniques.

4.2 Evaluation Metrics

Jaccard Index (mIoU): Jaccard index is a common evaluation metric to evaluate the segmentation quality. It is calculated as the intersection over union

(IoU) of the predicted and groundtruth masks. We compute the mean of the IoU across all the frames in a sequence and thus also refer to this metric as mIoU.

Contour Accuracy (F) [40]**:** To measure the quality of the predicted mask, we assess the contour accuracy by computing a bipartite matching between the contour points of the predicted segmentation and the contour points of the groundtruth segmentation. Based on the matching result we calculate the contour accuracy via the F-1 score.

Error Rate [13]**:** Following the evaluation protocol in [13], we compute the error rate on the JumpCut dataset. We select key frames $i = \{0, 16, ..., 96\}$ in each sequence and for the i-th keyframe, we compute the error in the predicted segmentation of the $i + d$-th frames, given the groundtruth segmentation mask of the i-th frame. Intuitively, we measure the transfer (or matching) error of methods with respect to a certain transfer distance d. The error is equal to the number of false positive and false negative pixels (the mislabeled pixels) divided by the number of all positive pixels (all foreground pixels) in the predicted segmentation of the $i + d$-th frame. We use $d = 16$ in the experiments and compute the average of the errors to obtain the error rate.

4.3 Quantitative Results

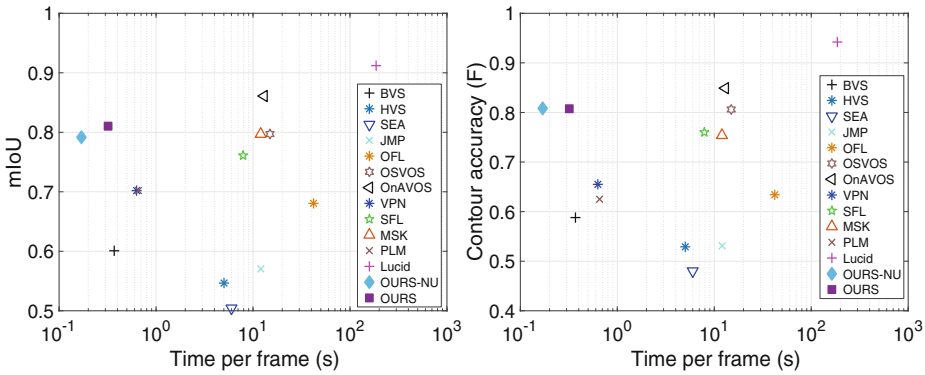
We carefully evaluated the proposed approach and compared the proposed method with a wide variety of video object segmentation methods *i.e.*, MSK [25], SFL [8], OSVOS [5], OnAVOS [53], PLM [59], MaskRNN [19], Lucid [26], SEA [1], HVS [15], JMP [13], FCP [41], BVS [34], OFL [50], CTN [24], VPN [23], SVC [54], JFS [36], LTV [37], HBT [14], AFS [51], SCF [21], RB [2] and DA [60]. Note that MSK, OSVOS, SFL, OnAVOS, PLM, MaskRNN, Lucid employ fine-tuning during testing.

We present the quantitative results on four datasets: DAVIS-16 [40], YouTube-Objects [43], JumpCut [13] and DAVIS-17 [42]. Our method outperforms state-of-the-art methods by 0.4% in mIoU and by 0.71 in error rate on Youtube-Objects and JumpCut datasets, respectively. On DAVIS-16 and DAVIS-17 datasets, our approach performs on par with state-of-the-art techniques while not using fine-tuning. The quantitative results are summarized in Tables 1, 2, 3, 4 and Fig. 4. The best method is highlighted in bold and the second-best method is underlined. Details are described in the following.

Evaluation on the DAVIS-16 Dataset: In Table 1, we compare our method with deep net baselines that do not require fine-tuning as well, such as VPN [23] and CTN [24]. We also compare to OSVOS [5], MSK [25], OnAVOS [53] and SFL [8], disabling their fine-tuning step. We use the super-script '$^-$' to denote methods with a disabled fine-tuning step. In Table 1, we report the mean IoU and the average running time per frame for each method tested on the DAVIS-16 dataset. Our method achieves the best mIoU, outperforming the baselines by more than 6% while running efficiently. Our method without the outlier

Table 1. Comparisons with deep net methods *without* fine-tuning (VPN and CTN) or with fine-tuning step disabled (denoted with $^-$) on DAVIS-16 validation set. OURS-NU: our method without online update and outlier removal.

	OURS	OURS-NU	OSVOS$^-$	MSK$^-$	OnAVOS$^-$	SFL$^-$	VPN	CTN
mIoU	**0.810**	0.792	0.525	0.699	0.736	0.674	0.702	0.735
Speed (s)	0.32	0.17	**0.12**	0.15	3.55	0.3	0.63	29.95



(a) mIoU vs. speed (b) F vs. speed

Fig. 4. Performance comparison on the DAVIS-16 validation set. The x axis denotes the average running time per frame in seconds (log scale) and the y axis is (a) mIoU (Jaccard index) and (b) F score (contour accuracy).

removal (denoted as OURS-NU in Table 1) runs 2 times faster while achieving competitive performance.

In Fig. 4, we compare our method which does not require fine-tuning with baselines that may or may not need fine-tuning. We report the mIoU vs average computational time per frame in Fig. 4(a) and the contour accuracy vs running time per frame in Fig. 4(b). Note that the average running time per frame also includes the fine-tuning step for those methods requiring fine-tuning. Since the network employed in our method is general enough to learn how to match we observe competitive performance at a fraction of the time required by other techniques. Note that the time axis scaling is logarithmic.

Evaluation on the YouTube-Objects Dataset: We present the evaluation results on the YouTube-Objects dataset [21,43] in Table 2. Our method outperforms the baselines despite the fact that our network is not fine-tuned, but other baselines such as OnAVOS and MSK and OSVOS are. Thus, our method is more favorable both in terms of computational time and in terms of accuracy.

Evaluation on the JumpCut Dataset: We present the evaluation results on the JumpCut dataset [13] in Table 3. We follow the evaluation in [13] and compute the error rates of different methods. The transfer distance d is equal to 16. In this experiment we don't apply the outlier removal described in Sect. 3.3 to

Table 2. Evaluation on the Youtube-Object dataset [21,43] using Jaccard index (mIoU).

Sequence	OURS	OnAVOS	MSK	OSVOS	OFL	JFS	BVS	SCF	AFS	FST	HBT	LTV
Fine-tuned?	-	Yes	Yes	Yes	-	-	-	-	-	-	-	-
Aeroplane	0.880	0.902	0.816	0.882	0.899	0.89	0.868	0.863	0.799	0.709	0.736	0.137
Bird	0.873	0.879	0.829	0.857	0.842	0.816	0.809	0.81	0.784	0.706	0.561	0.122
Boat	0.805	0.816	0.747	0.775	0.74	0.742	0.651	0.686	0.601	0.425	0.578	0.108
Car	0.779	0.738	0.670	0.796	0.809	0.709	0.687	0.694	0.644	0.652	0.339	0.237
Cat	0.788	0.759	0.696	0.708	0.683	0.677	0.559	0.589	0.504	0.521	0.305	0.186
Cow	0.771	0.787	0.750	0.778	0.798	0.791	0.699	0.686	0.657	0.445	0.418	0.163
Dog	0.803	0.809	0.752	0.813	0.766	0.703	0.685	0.618	0.542	0.653	0.368	0.18
Horse	0.688	0.742	0.649	0.728	0.726	0.678	0.589	0.54	0.508	0.535	0.443	0.115
Motorbike	0.774	0.663	0.498	0.735	0.737	0.615	0.605	0.609	0.583	0.442	0.489	0.106
Train	0.811	0.838	0.777	0.757	0.763	0.782	0.652	0.663	0.624	0.296	0.392	0.196
Average	**0.797**	0.793	0.718	0.783	0.776	0.74	0.68	0.676	0.625	0.538	0.463	0.155

Table 3. Error rates on the JumpCut dataset [13]. The transfer distance d is 16.

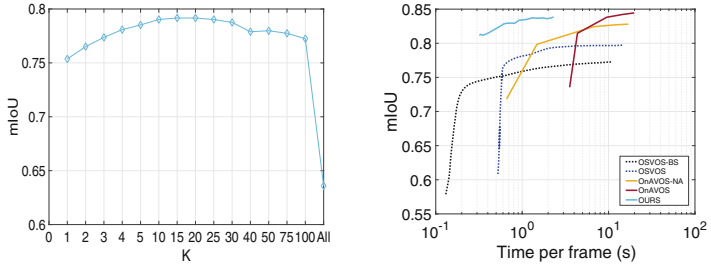
		RB	DA	SEA	JMP	SVC	PLM	OURS			RB	DA	SEA	JMP	SVC	PLM	OURS
	Fine-tuned?	-	-	-	-	-	Yes	-			-	-	-	-	-	Yes	-
ANIMAL	bear	4.58	4.48	4.21	4	**2.11**	3.45	5.14	SNAPCUT	animation	11.9	6.38	6.78	4.55	**3.35**	5.86	6.15
	giraffe	22	11.2	17.4	**7.4**	9.67	17.4	11.96		fish	51.8	21.7	25.7	17.5	7.67	**7.42**	12.21
	goat	13.1	13.3	8.22	**4.14**	4.97	15.2	4.73		horse	8.39	45.1	37.8	6.8	**4.84**	7.94	8.25
	pig	9.22	9.85	10.3	3.43	**3.24**	5.15	5.12		Avg.	24.03	24.39	23.43	9.62	**5.29**	7.07	8.87
	Avg.	12.23	9.71	10.03	**4.74**	5.00	10.30	6.74	FAST	bball	18.4	8.47	8.89	**3.9**	4.16	8.04	6.19
HUMAN	couple	17.5	16	23.4	**5.13**	8.49	9.14	11.77		cheetah	31.5	16.6	7.68	8.16	**7.1**	11.8	7.61
	park	11.8	6.54	6.91	5.39	**5.33**	10.2	11.42		dance	56.1	50.8	43	18.7	26.5	**14.7**	17.31
	station	8.85	20.9	21.3	9.01	8.42	**4.68**	9.98		hiphop	67.5	51.1	33.7	14.2	21.9	13.6	**10.49**
	Avg.	12.72	14.48	17.20	**6.51**	7.41	8.01	11.06		kongfu	40.2	40.8	17.9	8	**3.77**	6.25	4.05
STATIC	car	**1.76**	5.93	5.08	2.26	2.57	2.18	1.86		skater	38.7	40.8	29.6	22.8	21.4	**12.6**	13.57
	cup	5.45	12.9	9.31	**2.15**	2.4	6.04	5.38		supertramp	129	60.5	57.4	42.9	27.1	**20.7**	22.12
	pot	2.43	5.03	2.98	2.95	**1.79**	2.66	5.55		tricking	79.4	70.9	35.8	21.3	21.2	15.7	**8.32**
	toy	**1.28**	3.19	2.16	1.3	1.49	2.25	2.81		Avg.	57.60	42.50	29.25	17.50	16.64	12.92	**11.21**
	Avg.	2.73	6.76	4.88	2.17	**2.06**	3.28	3.90									
Average		28.68	23.75	18.89	9.82	9.07	9.23	**8.73**									

restrict mask transfer between non-successive frames. Again, our method outperforms the baselines on this dataset with an average error rate that is 0.34 lower than the best competing baseline SVC [54].

Evaluation on the DAVIS-17 Dataset: We show the experiments on instance-level video object segmentation using the DAVIS-17 validation set. The results are shown in Table 4. Our method performs reasonably well when compared to methods without finetuning, *i.e.*, OSVOS$^-$, OnAVOS$^-$, MaskRNN$^-$ and OFL. We further finetune our method (denoted as OURS-FT), and the performance is competitive among the baselines while the computational time is much faster. Note that OnAVOS$^+$ [52] in Table 4 is OnAVOS with upsampling layers on top and model ensembles.

Table 4. Evaluation on the DAVIS-17 validation set.

	OURS	OFL	OSVOS$^-$	OnAVOS$^-$	MaskRNN$^-$	OSVOS	OnAVOS	MaskRNN	OnAVOS$^+$	OURS-FT
Fine-tuned?	-	-	-	-	-	Yes	Yes	Yes	Yes	Yes
mIoU	0.565	0.549	0.366	0.395	0.455	0.521	0.610	0.605	**0.645**	0.614
Speed (s)	0.35	130	**0.13**	3.78	0.6	5	13	9	30	2.62



(a) Effect of K in Top K (b) Effect of fine-tuning

Fig. 5. Sensitivity analysis and finetuning. (a) The effect of K when computing the Top K similarity scores in the soft matching layer. (b) The effect of fine-tuning of our approach compared with other baselines. Both results are shown using the DAVIS-16 validation dataset.

4.4 Ablation Study

We study the important components of the proposed method. Subsequently, we discuss the effect of outlier removal and online update, the effect of K, the effect of foreground and background matching, the effect of fine-tuning and the memory consumption of the proposed approach.

Table 5. Ablation study of the three modules in our approach: (1) outlier removal, (2) online background update, and (3) online foreground update, assessed on the DAVIS-16 validation set.

Outlier removal	BG update	FG update	mIoU
-	-	-	0.792
✓	-	-	0.805
✓	✓	-	0.809
✓	✓	✓	0.810

Effect of K: We study the effect of K in the proposed soft matching layer where we compute the average similarity scores of top K matchings. We present the performance on DAVIS-16 with different settings of K in Fig. 5(a). We varied K to be between 1 and 100. The performance when K is equal to 1 ('hard matching') is 0.753 while the performance increases when K is larger than 1 ('soft

matching') until K is equal to 20. When K is larger than 20, the performance keeps decreasing and the performance of computing the average similarity scores among all matchings is 0.636. Intuitively, a point is a good match to a region if the feature of the point is similar to a reasonable amount of pixels in that region, which motivates the proposed soft matching layer.

Outlier Removal and Online Update: In Table 5, we study the effects of outlier removal, online background feature update and foreground feature update. We found that our method with neither outlier removal nor online update performs competitively, achieving 0.792 on DAVIS-16. Removing of outliers improves the performance by 0.013. If we incorporate the online background feature update, the performance improves by 0.004 and having the foreground feature updated as well further improves the performance, achieving 0.810 in mIoU on the DAVIS-16 dataset.

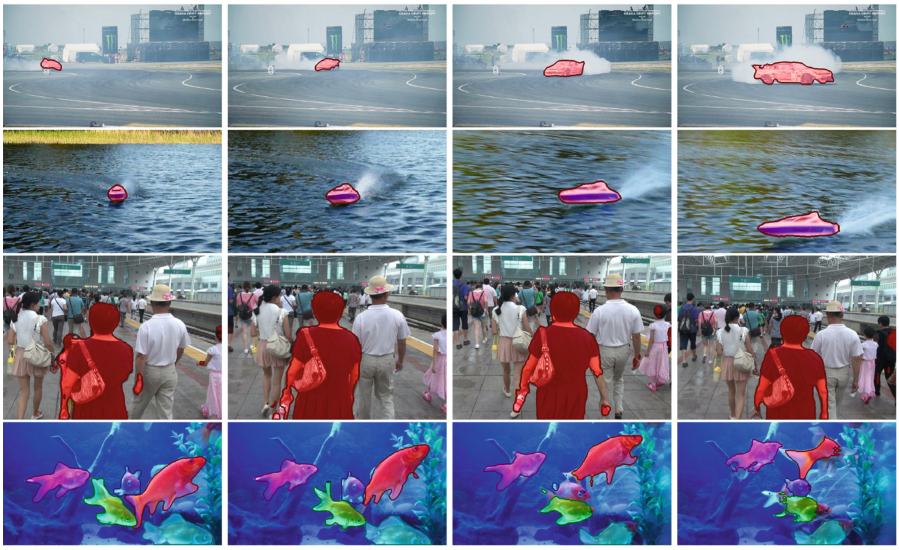


Fig. 6. Visual results of our approach. Testing videos are from DAVIS-16 (1^{st} row), Youtube-Objects (2^{nd} row), JumpCut (3^{rd} row), and DAVIS-17 datasets (4^{th} row).

Fig. 7. Failure cases of our approach. For each case, we show the results of our approach at the beginning and toward the end of the video sequence.

Matching Foreground and Background: As shown in Fig. 1, we match the input image with not only the foreground region but also the background region in the template and thus we have two soft matching layers for computing the foreground similarity and the background similarity. We found that having both foreground and background models is important for good performance. Specifically, the performance of matching only the foreground, *i.e.*, only having one soft matching layer to compute foreground similarity, is only 0.527 in mIoU on DAVIS-16 while having both foreground and background similarity computed achieves 0.792.

Online Fine-Tuning: We would like to point out that the network in our method can be fune-tuned during testing when observing the groundtruth mask of the first frame. We show the trade-off between fine-tuning time and performance on DAVIS-16 in Fig. 5(b). Specifically, we show the average running time per frame taking the fine-tuning step into account, and compare with OSVOS, OSVOS-BS (OSVOS without the post-processing step), OnAVOS and OnAVOS-NA (OnAVOS without test time augmentation). We report the results of OnAVOS and OnAVOS-NA without a CRF as post-processing. Note that the time axis scaling is again logarithmic. The bottom left point of each curve denotes performance without fine-tuning. Clearly, the performance of our approach outperforms other baselines if fine-tuning is prohibited. After fine-tuning, our method can be further improved and still runs efficiently, taking 2.5 s per frame while other baselines require more than 10 s to achieve their peak performance. Note that we don't have any post-processing step to refine the segmentation mask in our method while still achieving competitive results.

4.5 Qualitative Results

In Fig. 6, we show visual results of our method on DAVIS-16 (1st row), Youtube-Objects (2nd row), JumpCut (3rd row), and DAVIS-17 datasets (4th row). We observe our method can accurately segment the foreground objects with challenges such as fast motion, cluttered background and appearance change. We also observe the proposed method produce accurate instance level segmentation on DAVIS-17 datasets.

We show the failure cases of our method in Fig. 7. Possible reasons for our method to fail include tiny objects and similar appearance of different instances.

5 Conclusion

We present an efficient video object segmentation algorithm base on a novel soft matching layer. The method generalizes well and does not require online fine-tuning while maintaining good accuracy. Our method achieves state-of-the-art on the Youtube-Objects and JumpCut datasets and is competitive on DAVIS-16 and DAVIS-17, while its computational time is at least one order of magnitude faster than current state-of-the-art.

Acknowledgments. This material is based upon work supported in part by the National Science Foundation under Grant No. 1718221, 1755785, Samsung, and 3M. We thank NVIDIA for providing the GPUs used for this research.

References

1. Avinash Ramakanth, S., Venkatesh Babu, R.: SeamSeg: video object segmentation using patch seams. In: Proceedings of CVPR (2014)
2. Bai, X., Wang, J., Simons, D., Sapiro, G.: Video snapcut: robust video object cutout using localized classifiers. In: SIGGRAPH (2009)
3. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: Proceedings of CVPR (2017)
4. Caelles, S., Chen, Y., Pont-Tuset, J., Van Gool, L.: Semantically-guided video object segmentation (2017). arXiv preprint: arXiv:1704.01926
5. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: Proceedings of CVPR (2017)
6. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. PAMI **40**, 834–848 (2018)
7. Chen, Y., Pont-Tuset, J., Montes, A., Van Gool, L.: Blazingly fast video object segmentation with pixel-wise metric learning. In: Proceedings of CVPR (2018)
8. Cheng, J., Tsai, Y.H., Wang, S., Yang, M.H.: SegFlow: joint learning for video object segmentation and optical flow. In: Proceedings of ICCV (2017)
9. Cheng, J., Tsai, Y.H., Hung, W.C., Wang, S., Yang, M.H.: Fast and accurate online video object segmentation via tracking parts. In: Proceedings of CVPR (2018)
10. Dosovitskiy, A., et al.: Flownet: learning optical flow with convolutional networks. In: Proceedings of ICCV (2015)
11. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge: a retrospective. IJCV **111**, 98–136 (2015)
12. Faktor, A., Irani, M.: Video segmentation by non-local consensus voting. In: BMVC (2014)
13. Fan, Q., Zhong, F., Lischinski, D., Cohen-Or, D., Chen, B.: JumpCut: non-successive mask transfer and interpolation for video cutout. In: SIGGRAPH (2015)
14. Godec, M., Roth, P.M., Bischof, H.: Hough-based tracking of non-rigid objects. In: Proceedings of ICCV (2011)
15. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: Proceedings of CVPR (2010)
16. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: Proceedings of ICCV (2011)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR (2016)
18. Hu, Y.T., Lin, Y.Y., Chen, H.Y., Hsu, K.J., Chen, B.Y.: Matching images with multiple descriptors: an unsupervised approach for locally adaptive descriptor selection. TIP **24**, 5995–6010 (2015)
19. Hu, Y.T., Huang, J.B., Schwing, A.: MaskRNN: instance level video object segmentation. In: NIPS (2017)

20. Hu, Y.T., Huang, J.B., Schwing, A.: Unsupervised video object segmentation using motion saliency-guided spatio-temporal propagation. In: Ferrari, V., et al. (eds.) ECCV 2018, Part VIII. LNCS, vol. 11205, pp. 813–830. Springer, Cham (2018)
21. Jain, S.D., Grauman, K.: Supervoxel-consistent foreground propagation in video. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part IV. LNCS, vol. 8692, pp. 656–671. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_43
22. Jain, S.D., Xiong, B., Grauman, K.: FusionSeg: learning to combine motion and appearance for fully automatic segmention of generic objects in videos. In: Proceedings of CVPR (2017)
23. Jampani, V., Gadde, R., Gehler, P.V.: Video propagation networks. In: Proceedings of CVPR (2017)
24. Jang, W.D., Kim, C.S.: Online video object segmentation via convolutional trident network. In: Proceedings of CVPR (2017)
25. Khoreva, A., Perazzi, F., Benenson, R., Schiele, B., Sorkine-Hornung, A.: Learning video object segmentation from static images. In: Proceedings of CVPR (2017)
26. Khoreva, A., Benenson, R., Ilg, E., Brox, T., Schiele, B.: Lucid data dreaming for object tracking (2017). arXiv preprint: arXiv:1703.09554
27. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2014)
28. Kristan, M., et al.: A novel performance evaluation methodology for single-target trackers. PAMI **38**, 2137–2155 (2016)
29. Lee, Y.J., Kim, J., Grauman, K.: Key-segments for video object segmentation. In: Proceedings of ICCV (2011)
30. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: spatio-temporal video segmentation with long-range motion cues. In: Proceedings of CVPR (2011)
31. Li, F., Kim, T., Humayun, A., Tsai, D., Rehg, J.M.: Video segmentation by tracking many figure-ground segments. In: Proceedings of ICCV (2013)
32. Li, X., et al.: Video object segmentation with re-identification. In: The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops (2017)
33. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60**, 91–110 (2004)
34. Maerki, N., Perazzi, F., Wang, O., Sorkine-Hornung, A.: Bilateral space video segmentation. In: Proceedings of CVPR (2016)
35. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 1615–1630 (2005)
36. Nagaraja, N., Schmidt, F., Brox, T.: Video segmentation with just a few strokes. In: Proceedings of ICCV (2015)
37. Ochs, P., Malik, J., Brox, T.: Segmentation of moving objects by long term video analysis. PAMI **36**, 1187–1200 (2014)
38. Oh, S.W., Lee, J.Y., Sunkavalli, K., Kim, S.J.: Fast video object segmentation by reference-guided mask propagation. In: Proceedings of CVPR (2018)
39. Papazoglou, A., Ferrari, V.: Fast object segmentation in unconstrained video. In: Proceedings of ICCV (2013)
40. Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L.V., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: Proceedings of CVPR (2016)
41. Perazzi, F., Wang, O., Gross, M., Sorkine-Hornung, A.: Fully connected object proposals for video segmentation. In: Proceedings of ICCV (2015)

42. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L.: The 2017 DAVIS challenge on video object segmentation (2017). arXiv preprint: arXiv:1704.00675
43. Prest, A., Leistner, C., Civera, J., Schmid, C., Ferrari, V.: Learning object class detectors from weakly annotated video. In: Proceedings of CVPR (2012)
44. Price, B.L., Morse, B.S., Cohen, S.: LIVEcut: learning-based interactive video segmentation by evaluation of multiple propagated cues. In: Proceedings of ICCV (2009)
45. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Deepmatching: hierarchical deformable dense matching. IJCV **120**, 300–323 (2016)
46. Rocco, I., Arandjelovic, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: Proceedings of CVPR (2017)
47. Tokmakov, P., Alahari, K., Schmid, C.: Learning motion patterns in videos. In: Proceedings of CVPR (2017)
48. Tokmakov, P., Alahari, K., Schmid, C.: Learning video object segmentation with visual memory. In: Proceedings of ICCV (2017)
49. Tsai, D., Flagg, M., Rehg, J.: Motion coherent tracking with multi-label MRF optimization. In: Proceedings of BMVC (2010)
50. Tsai, Y.H., Yang, M.H., Black, M.J.: Video segmentation via object flow. In: Proceedings of CVPR (2016)
51. Vijayanarasimhan, S., Grauman, K.: Active frame selection for label propagation in videos. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 496–509. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_36
52. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for the 2017 DAVIS challenge on video object segmentation. In: The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops (2017)
53. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: BMVC (2017)
54. Wang, W., Shen, J., Porikli, F.: Selective video object cutout. TIP **26**, 5645–5655 (2017)
55. Xiao, F., Lee, Y.J.: Track and segment: an iterative unsupervised approach for video object proposals. In: Proceedings of CVPR (2016)
56. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. In: Proceedings of CVPR (2018)
57. Yang, T.Y., Hsu, J.H., Lin, Y.Y., Chuang, Y.Y.: DeepCD: learning deep complementary descriptors for patch representations. In: Proceedings of ICCV (2017)
58. Yilmaz, A., Javed, O., Shah, M.: Object tracking: a survey. ACM Comput. Surv. (CSUR) **38**, 13 (2006)
59. 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: Proceedings of ICCV (2017)
60. Zhong, F., Qin, X., Peng, Q., Meng, X.: Discontinuity-aware video object cutout. In: SIGGRAPH (2012)