One-Shot Video Object Segmentation

S. Caelles[1,*] K.-K. Maninis[1,*] J. Pont-Tuset[1] L. Leal-Taixé[2] D. Cremers[2] L. Van Gool[1]
[1]ETH Zürich [2]TU München

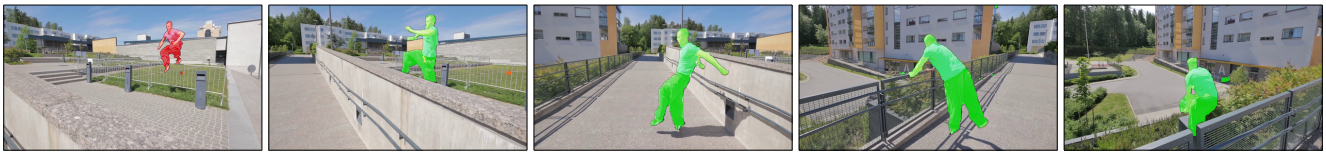


Figure 1. Example result of our technique: The segmentation of the first frame (red) is used to learn the model of the specific object to track, which is segmented in the rest of the frames independently (green). One every 20 frames shown of 90 in total.

Abstract

This paper tackles the task of semi-supervised video object segmentation, i.e.*, the separation of an object from the background in a video, given the mask of the first frame. We present One-Shot Video Object Segmentation (OSVOS), based on a fully-convolutional neural network architecture that is able to successively transfer generic semantic information, learned on ImageNet, to the task of foreground segmentation, and finally to learning the appearance of a single annotated object of the test sequence (hence one-shot). Although all frames are processed independently, the results are temporally coherent and stable. We perform experiments on two annotated video segmentation databases, which show that OSVOS is fast and improves the state of the art by a significant margin (79.8% vs 68.0%).*

1. Introduction

From Pre-Trained Networks...

Convolutional Neural Networks (CNNs) are revolutionizing many fields of computer vision. For instance, they have dramatically boosted the performance for problems like image classification [24, 47, 19] and object detection [15, 14, 26]. Image segmentation has also been taken over by CNNs recently [29, 23, 51, 3, 4], with deep architectures pre-trained on the weakly related task of image classification on ImageNet [44]. One of the major downsides of deep network approaches is their hunger for training data. Yet, with various pre-trained network architectures one may ask how much training data do we really need for the specific problem at hand? This paper investigates segmenting an object along an entire video, when we only have one single labeled training example, e.g. the first frame.

*First two authors contributed equally

...to One-Shot Video Object Segmentation

This paper presents *One-Shot Video Object Segmentation* (OSVOS), a CNN architecture to tackle the problem of semi-supervised video object segmentation, that is, the classification of all pixels of a video sequence into background and foreground, given the manual annotation of one (or more) of its frames. Figure 1 shows an example result of OSVOS, where the input is the segmentation of the first frame (in red), and the output is the mask of the object in the 90 frames of the sequence (in green).

The first contribution of the paper is to adapt the CNN to a particular object instance given a single annotated image (hence *one-shot*). To do so, we adapt a CNN pre-trained on image recognition [44] to video object segmentation. This is achieved by training it on a set of videos with manually segmented objects. Finally, it is fine-tuned *at test time* on a specific object that is manually segmented in a single frame. Figure 2 shows the overview of the method. Our proposal tallies with the observation that leveraging these different levels of information to perform object segmentation would stand to reason: from generic semantic information of a large amount of categories, passing through the knowledge of the *usual* shapes of objects, down to the specific properties of a particular object we are interested in segmenting.

The second contribution of this paper is that OSVOS processes each frame of a video independently, obtaining temporal consistency as a by-product rather than as the result of an explicitly imposed, expensive constraint. In other words, we cast video object segmentation as a per-frame segmentation problem given the *model* of the object from one (or various) manually-segmented frames. This stands in contrast to the dominant approach where temporal consistency plays the central role, assuming that objects do not change too much between one frame and the next. Such methods adapt their single-frame models smoothly throughout

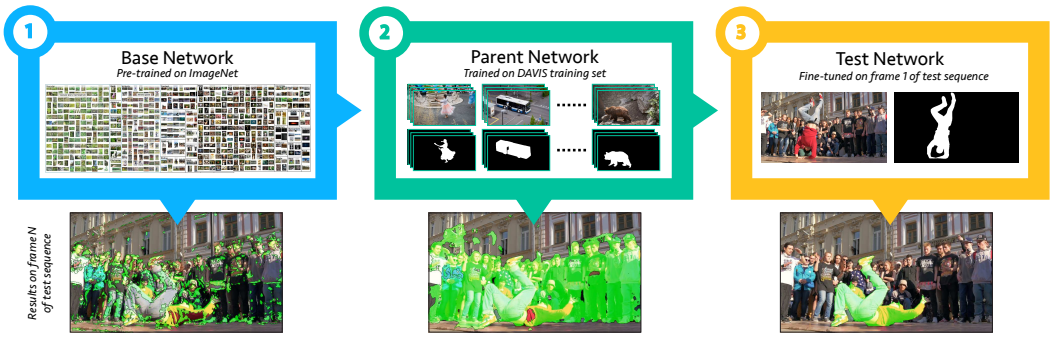


Figure 2. **Overview of OSVOS**: (1) We start with a pre-trained base CNN for image labeling on ImageNet; its results in terms of segmentation, although conform with some image features, are not useful. (2) We then train a *parent network* on the training set of DAVIS; the segmentation results improve but are not focused on an specific object yet. (3) By fine-tuning on a segmentation example for the specific target object in a single frame, the network rapidly focuses on that target.

the video, looking for targets whose shape and appearance vary *gradually* in consecutive frames, but fail when those constraints do not apply, unable to recover from relatively common situations such as occlusions and abrupt motion.

In this context, *motion* estimation has emerged as a key ingredient for state-of-the-art video segmentation algorithms [49, 42, 17]. Exploiting it is not a trivial task however, as one *e.g.* has to compute temporal matches in the form of optical flow or dense trajectories [5], which can be an even harder problem.

We argue that temporal consistency was needed in the past, as one had to overcome major drawbacks of the then inaccurate shape or appearance models. On the other hand, in this paper deep learning will be shown to provide a sufficiently accurate model of the target object to produce temporally stable results even when processing each frame independently. This has some natural advantages: OSVOS is able to segment objects through occlusions, it is not limited to certain ranges of motion, it does not need to process frames sequentially, and errors are not temporally propagated. In practice, this allows OSVOS to handle *e.g.* interlaced videos of surveillance scenarios, where cameras can go blind for a while before coming back on again.

Our third contribution is that OSVOS can work at various points of the trade-off between speed and accuracy. In this sense, it can be adapted in two ways. First, given one annotated frame, the user can choose the level of fine-tuning of OSVOS, giving him/her the freedom between a faster method or more accurate results. Experimentally, we show that OSVOS can run at 181 ms per frame and 71.5% accuracy, and up to 79.7% when processing each frame in 7.83 s. Second, the user can annotate more frames, those on which the current segmentation is less satisfying, upon which OSVOS will refine the result. We show in the experiments that the results indeed improve gradually with more supervision, reaching an outstanding level of 84.6% with two annotated frames per sequence, and 86.9% with four, from 79.8% from one annotation.

Technically, we adopt the architecture of Fully Convolutional Networks (FCN) [12, 27], suitable for dense predictions. FCNs have recently become popular due to their performance both in terms of accuracy and computational efficiency [27, 8, 9]. Arguably, the Achilles' heel of FCNs when it comes to segmentation is the coarse scale of the deeper layers, which leads to inaccurately localized predictions. To overcome this, a large variety of works from different fields use skip connections of larger feature maps [27, 18, 51, 30], or learnable filters to improve upscaling [34, 52]. To the best of our knowledge, this work is the first to use FCNs for the task of video segmentation.

We perform experiments on two video object segmentation datasets (DAVIS [37] and Youtube-Objects [41, 20]) and show that OSVOS significantly improves the state of the art 79.8% vs 68.0%. Our technique is able to process a frame of DAVIS (480×854 pixels) in 102 ms. By increasing the level of supervision, OSVOS can further improve its results to 86.9% with just four annotated frames per sequence, thus providing a vastly accelerated rotoscoping tool.

All resources of this paper, including training and testing code, pre-computed results, and pre-trained models are publicly available at `www.vision.ee.ethz.ch/~cvlsegmentation/osvos/`.

2. Related Work

Video Object Segmentation and Tracking: Most of the current literature on semi-supervised video object segmentation enforces temporal consistency in video sequences to propagate the initial mask into the following frames. First of all, in order to reduce the computational complexity some works make use of superpixels [6, 17], patches [42, 11], or even object proposals [38]. Märki *et al.* [33] cast the problem into a bilateral space in order to solve it more efficiently. After that, an optimization using one of the previous aggregations of pixels is usually performed; which can consider the full video sequence [38, 33], a subset of

frames [17], or only the results in frame n to obtain the mask in $n+1$ [42, 6, 11]. As part of their pipeline, some of the methods include the computation of optical flow [17, 42], which considerably reduces speed. Concurrent works have also used deep learning to address Video Object Segmentation. MaskTrack [22] learns to refine the detected masks frame by frame, by using the detections of the previous frame, along with Optical Flow and post-processing with CRFs. In [21], the authors combine training of a CNN with ideas of bilateral filtering. Different from those approaches, OSVOS is a simpler pipeline which segments each frame independently, and produces more accurate results, while also being significantly faster.

In the case of visual tracking (bounding boxes instead of segmentation) Nam and Han [32] use a CNN to learn a representation of the object to be tracked, but only to look for the most similar window in frame $n+1$ given the object in frame n. In contrast, our CNN learns a single model from frame 1 and segments the rest of the frames from this model.

FCNs for Segmentation: Segmentation research has closely followed the innovative ideas of CNNs in the last few years. The advances observed in image recognition [24, 47, 19] have been beneficial to segmentation in many forms (semantic [27, 34], instance- level [14, 39, 8], biomedical [43], generic [29], etc.). Many of the current best performing methods have in common a deep architecture, usually pre-trained on ImageNet, trainable end-to-end. The idea of dense predictions with CNNs was pioneered by [12] and formulated by [27] in the form of Fully Convolutional Networks (FCNs) for semantic segmentation. The authors noticed that by changing the last fully connected layers to 1×1 convolutions it is possible to train on images of arbitrary size, by predicting correspondingly-sized outputs. Their approach boosts efficiency over patch-based approaches where one needs to perform redundant computations in overlapping patches. More importantly, by removing the parameter-intensive fully connected layers, the number of trainable parameters drops significantly, facilitating training with relatively few labeled data.

In most CNN architectures [24, 47, 19], activations of the intermediate layers gradually decrease in size, because of spatial pooling operations or convolutions with a stride. Making dense predictions from downsampled activations results in coarsely localized outputs [27]. Deconvolutional layers that learn how to upsample are used in [34, 52]. In [39], activations from shallow layers are gradually injected into the prediction to favor localization. However, these architectures come with many more trainable parameters and their use is limited to cases with sufficient data.

Following the ideas of FCNs, Xie and Tu [51] separately supervised the intermediate layers of a deep network for contour detection. The duality between multiscale contours and hierarchical segmentation [1, 40] was further studied by Maninis *et al*. [29] by bringing CNNs to the field of generic image segmentation. In this work we explore how to train an FCN for accurately localized dense prediction based on very limited annotation: a single segmented frame.

3. One-Shot Deep Learning

Let us assume that one would like to segment an object in a video, for which the only available piece of information is its foreground/background segmentation in one frame. Intuitively, one could analyze the entity, create a *model*, and search for it in the rest of the frames. For humans, this very limited amount of information is more than enough, and changes in appearance, shape, occlusions, etc. do not pose a significant challenge, because we leverage strong priors: first "It is an object," and then "It is *this particular* object." Our method is inspired by this gradual refinement.

We train a Fully Convolutional Neural Network (FCN) for the binary classification task of separating the foreground object from the background. We use two successive training steps: First we train on a large variety of objects, offline, to construct a model that is able to discriminate the general notion of a foreground object, *i.e*., "It is an object." Then, at test time, we fine-tune the network for a small number of iterations on the particular instance that we aim to segment, *i.e*., "It is *this particular* object." The overview of our method is illustrated in Figure 2.

3.1. End-to-end trainable foreground FCN

Ideally, we would like our CNN architecture to satisfy the following criteria:

1. Accurately localized segmentation output, as discussed in Section 2.
2. Relatively small number of parameters to train from a limited amount of annotation data.
3. Relatively fast testing times.

We draw inspiration from the CNN architecture of [30], originally used for biomedical image segmentation. It is based on the VGG [47] network, modified for accurately localized dense prediction (Point 1). The fully-connected layers needed for classification are removed (Point 2), and efficient image-to-image inference is performed (Point 3). The VGG architecture consists of groups of convolutional plus Rectified Linear Units (ReLU) layers grouped into 5 stages. Between the stages, pooling operations downscale the feature maps as we go deeper into the network. We connect convolutional layers to form separate skip paths from the last layer of each stage (before pooling). Upscaling operations take place wherever necessary, and feature maps from the separate paths are concatenated to construct a volume with information from different levels of detail. We linearly fuse the feature maps to a single output which has the same dimensions as the image, and we assign a loss

function to it. The proposed architecture is shown in Figure 4 (1), foreground branch.

The pixel-wise cross-entropy loss for binary classification (we keep the notation of Xie and Tu [51]) is in this case defined as:

$$\mathcal{L}(\mathbf{W}) = -\sum_j y_j \log P(y_j{=}1|X;\mathbf{W}) + (1-y_j)\log(1-P(y_j{=}1|X;\mathbf{W}))$$
$$= -\sum_{j\in Y_+} \log P(y_j{=}1|X;\mathbf{W}) - \sum_{j\in Y_-} \log P(y_j{=}0|X;\mathbf{W})$$

where $\mathbf{W}$ are the standard trainable parameters of a CNN, X is the input image, $y_j \in 0, 1, j = 1, .., |X|$ is the pixel-wise binary label of X, and Y_+ and Y_- are the positive and negative labeled pixels. $P(\cdot)$ is obtained by applying a sigmoid to the activation of the final layer.

In order to handle the imbalance between the two binary classes, Xie and Tu [51] proposed a modified version of the cost function, originally used for contour detection (we drop $\mathbf{W}$ for the sake of readability):

$$\mathcal{L}_{mod} = -\beta \sum_{j\in Y_+} \log P(y_j{=}1|X) - (1-\beta) \sum_{j\in Y_-} \log P(y_j{=}0|X) \quad (1)$$

where $\beta = |Y_-|/|Y|$. Equation 1 allows training for imbalanced binary tasks [23, 51, 29, 30].

3.2. Training details

Offline training: The base CNN of our architecture [47] is pre-trained on ImageNet for image labeling, which has proven to be a very good initialization to other tasks [27, 51, 23, 29, 18, 52]. Without further training, the network is not capable of performing segmentation, as illustrated in Figure 2 (1). We refer to this network as the "*base network*."

We therefore further train the network on the binary masks of the training set of DAVIS, to learn a generic notion of how to segment objects from their background, their usual shapes, etc. We use Stochastic Gradient Descent (SGD) with momentum 0.9 for 50000 iterations. We augment the data by mirroring and zooming in. The learning rate is set to 10^{-8}, and is gradually decreased. After offline training, the network learns to segment foreground objects from the background, as illustrated in Figure 2 (2). We refer to this network as the "*parent network*."

Online training/testing: With the parent network available, we can proceed to our main task ("*test network*" in Figure 2 (3)): Segmenting a particular entity in a video, given the image and the segmentation of the first frame. We proceed by further training (fine-tuning) the parent network for the particular image/ground-truth pair, and then testing on the entire sequence, using the new weights. The timing of our method is therefore affected by two times: the fine-tuning time (once per annotated mask) and the segmentation of all frames (once per frame). In the former we have a trade-off between quality and time: the more iterations we allow the technique to learn, the better results but the longer the user will have to wait for results. The latter does not depend on the training time: OSVOS is able to segment each 480p frame (480×854) in 102 ms.

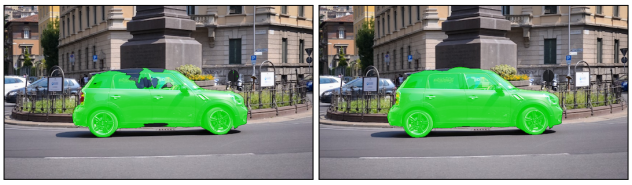


Figure 3. **Qualitative evolution of the fine tuning**: Results at 10 seconds and 1 minute per sequence.

Regarding the fine-tuning time, we present two different modes: One can either need to fine-tune online, by segmenting a frame and waiting for the results in the entire sequence, or offline, having access to the object to segment beforehand. Especially in the former mode, there is the need to control the amount of time dedicated to training: the more time allocated for fine-tuning, the more the user waits and the better the results are. In order to explore this trade-off, in our experiments we train for a period between 10 seconds and 10 minutes per sequence. Figure 3 shows a qualitative example of the evolution of the results' quality depending on the time allowed for fine-tuning.

In the experiments section, Figure 8 quantifies this evolution. Ablation analysis shows that both offline and online training are crucial for good performance: If we perform our online training directly from the ImageNet model, the performance drops significantly. Only dropping the online training for a specific object also yields a significantly worse performance, as already transpired from Figure 2 (2).

3.3. Contour snapping

In the field of image classification [24, 47, 19], where our base network was designed and trained, spatial invariance is a design choice: no matter where an object appears in the image, the classification result should be the same. This is in contrast to the accurate localization of the object contours that we expect in (video) object segmentation. Despite the use of skip connections [27, 18, 51, 30] to minimize the loss of spatial accuracy, we observe that OSVOS's segmentations have some room for improvement in terms of contour localization. We propose two different strategies to improve the results in this regard.

First, we propose the use of the Fast Bilateral Solver (FBS) [2] to snap the background prediction to the image edges. It performs a Gaussian smoothing in the five-dimensional *color-location* space, which results in a smoothing of the input signal (foreground segmentation) that preserves the edges of the image. It is useful in practice because it is fast (≈60 ms per frame), and it is differentiable so it can be included in an end-to-end trainable deep learn-

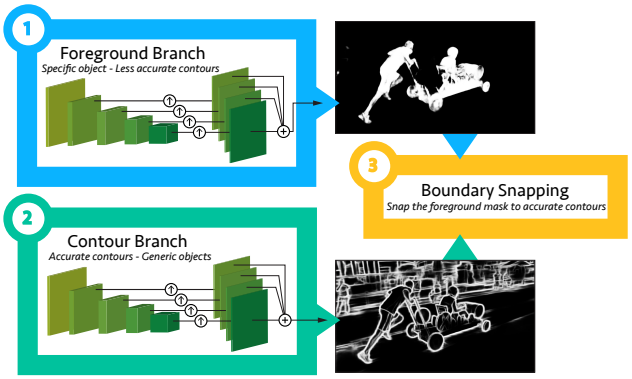


Figure 4. **Two-stream FCN architecture**: The main foreground branch (1) is complemented by a contour branch (2) which improves the localization of the boundaries (3).

ing architecture. The drawback of this approach, though, is that it preserves naive image *gradients*, *i.e.* pixels with high Euclidean differences in the color channels.

To overcome this limitation, our second approach snaps the results to *learned* contours instead of simple image gradients. To this end, we propose a complementary CNN in a second branch, that is trained to detect object contours. The proposed architecture is presented in Figure 4: (1) shows the main foreground branch, where the foreground pixels are estimated; (2) shows the contour branch, which detects all contours in the scene (not only those of the foreground object). This allows us to train offline, without the need to fine-tune on a specific example online. We used the exact same architecture in the two branches, but training for different losses. We noticed that jointly training a network with shared layers for both tasks rather degrades the obtained results thus we kept the computations for the two objectives uncorrelated. This allows us to train the contour branch only offline and thus it does not affect the online timing. Since there is need for high recall in the contours, we train on the PASCAL-Context [31] database, which provides contour annotations for the full scene of an image. Finally, in the boundary snapping step (Figure 4 (3), we compute superpixels that align to the computed contours (2) by means of an Ultrametric Contour Map (UCM) [1, 40], which we *threshold* at a low value. We then take a foreground mask (1) and we select superpixels via majority voting (those that overlap with the foreground mask over 50%) to form the final foreground segmentation.

In this second case, we trade accuracy for speed, since the snapping process takes longer (400 ms instead of 60 ms per frame), but we achieve more accurate results. Both refinement processes result in a further boost in performance, and are fully modular, meaning that depending on the requirements one can choose not to use them, sacrificing accuracy for execution time, since both modules come with a small, yet avoidable computational overhead.

4. Experimental Validation

Databases, state-of-the-art, and measures: The main part of our experiments is done on the recently-released DAVIS database [37], which consists of 50 full-HD video sequences with all of their frames segmented with pixel-level accuracy. We use three measures: region similarity in terms of intersection over union ($\mathcal{J}$), contour accuracy ($\mathcal{F}$), and temporal instability of the masks ($\mathcal{T}$). All evaluation results are computed on the validation set of DAVIS.

We compare to a large set of state-of-the-art methods, including two very recent semi-supervised techniques, OFL [49], BVS [33], as well as the methods originally compared on the DAVIS benchmark: FCP [38], JMP [11], HVS [17], SEA [42], and TSP [6]. We also add the unsupervised techniques: FST [36], SAL [46], KEY [25], MSG [5], TRC [13], CVOS [48], and NLC [10]. We add two informative bounds: the quality that an oracle would reach by selecting the best segmented object proposal out of two state-of-the-art techniques (COB [29] and MCG [40]), and by selecting the best superpixels from COB (COB|SP).

For completeness, we also experiment on Youtube-objects [41], manually segmented by Jain and Grauman [20]. We compare to OFL [49], BVS [33], LTV [35], HBT [16], AFS [50], SCF [20], and JFS [45] and take the pre-computed evaluation results from previous work.

Ablation Study on DAVIS: To analyze and quantify the importance and need of each of the proposed blocks of our algorithm, Table 1 shows the evaluation of OSVOS compared to ablated versions without each of its building blocks. Each column shows: the original method without boundary snapping (-BS), without pre-training the parent network on DAVIS (-PN), or without performing the one-shot learning on the specific sequence (-OS). In smaller and italic font we show the loss (in blue) or gain (in red) on each metric with respect to our final approach.

We can see that both the pre-training of the parent network and the one-shot learning play an important role (we lose 15.2 and 27.3 points in $\mathcal{J}$ without them, respectively). Removing both, *i.e.*, using the Imagenet raw CNN, the results in terms of segmentation ($\mathcal{J} = 17.6\%$) are completely *random*. The boundary snapping adds 2.4 points of im-

	Measure	Ours	-BS		-PN-BS		-OS-BS		-PN-OS-BS	
$\mathcal{J}$	Mean $\mathcal{M}\uparrow$	**79.8**	77.4	*2.4*	64.6	*15.2*	52.5	*27.3*	17.6	*62.2*
	Recall $\mathcal{O}\uparrow$	**93.6**	91.0	*2.6*	70.5	*23.2*	57.7	*35.9*	2.3	*91.3*
	Decay $\mathcal{D}\downarrow$	14.9	17.4	*2.5*	27.8	*13.0*	**–1.9**	*16.7*	1.8	*13.1*
$\mathcal{F}$	Mean $\mathcal{M}\uparrow$	**80.6**	78.1	*2.5*	66.7	*13.9*	47.7	*32.9*	20.3	*60.4*
	Recall $\mathcal{O}\uparrow$	**92.6**	92.0	*0.6*	74.4	*18.3*	47.9	*44.7*	2.4	*90.2*
	Decay $\mathcal{D}\downarrow$	15.0	19.4	*4.5*	26.4	*11.4*	**0.6**	*14.3*	2.4	*12.6*
$\mathcal{T}$	Mean $\mathcal{M}\downarrow$	37.6	**33.5**	*4.0*	60.9	*23.3*	53.8	*16.2*	46.0	*8.4*

Table 1. **Ablation study on DAVIS**: Comparison of OSVOS against downgraded versions without some of its components.

		Semi-Supervised								Unsupervised							Bounds		
	Measure	Ours	OFL	BVS	FCP	JMP	HVS	SEA	TSP	FST	NLC	MSG	KEY	CVOS	TRC	SAL	COB|SP	COB	MCG
$\mathcal{J}$	Mean $\mathcal{M}\uparrow$	**79.8**	68.0	60.0	58.4	57.0	54.6	50.4	31.9	**55.8**	55.1	53.3	49.8	48.2	47.3	39.3	**86.5**	79.3	70.7
	Recall $\mathcal{O}\uparrow$	**93.6**	75.6	66.9	71.5	62.6	61.4	53.1	30.0	**64.9**	55.8	61.6	59.1	54.0	49.3	30.0	**96.5**	94.4	91.7
	Decay $\mathcal{D}\downarrow$	14.9	26.4	28.9	**−2.0**	39.4	23.6	36.4	38.1	**0.0**	12.6	2.4	14.1	10.5	8.3	6.9	2.8	3.2	**1.3**
$\mathcal{F}$	Mean $\mathcal{M}\uparrow$	**80.6**	63.4	58.8	49.2	53.1	52.9	48.0	29.7	51.1	**52.3**	50.8	42.7	44.7	44.1	34.4	**87.1**	75.7	62.9
	Recall $\mathcal{O}\uparrow$	**92.6**	70.4	67.9	49.5	54.2	61.0	46.3	23.0	51.6	51.9	**60.0**	37.5	52.6	43.6	15.4	**92.4**	88.5	76.7
	Decay $\mathcal{D}\downarrow$	15.0	27.2	21.3	**−1.1**	38.4	22.7	34.5	35.7	**2.9**	11.4	5.1	10.6	11.7	12.9	4.3	2.3	3.9	**1.9**
$\mathcal{T}$	Mean $\mathcal{M}\downarrow$	37.6	21.7	34.5	29.6	15.3	35.0	**14.9**	41.2	34.3	41.4	29.1	25.2	**24.4**	37.6	64.1	**27.4**	44.1	69.8

Table 2. **DAVIS Validation**: OSVOS versus the state of the art, and practical bounds.

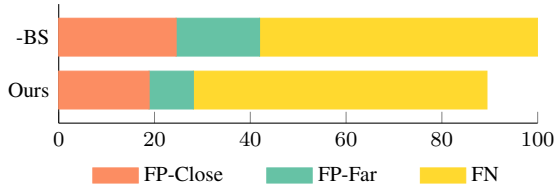


Figure 5. **Error analysis of our method**: Errors divided into False Positives (FP-Close and FP-Far) and False Negatives (FN). Values are total error pixels relative to the error in the -BS case.

Attr	Ours		OFL		BVS		FCP		JMP		HVS		SEA	
AC	**80.6**	*−1.2*	56.6	*17.6*	48.6	*17.6*	52.8	*8.6*	52.4	*7.0*	41.4	*20.4*	43.2	*11.1*
DB	**74.3**	*6.5*	44.3	*27.9*	31.9	*33.0*	53.4	*5.9*	40.7	*19.1*	42.9	*13.9*	31.1	*22.7*
FM	**76.5**	*5.1*	49.6	*28.2*	44.8	*23.3*	50.7	*11.9*	45.2	*18.0*	34.5	*31.0*	30.9	*30.1*
MB	**73.7**	*11.0*	55.5	*22.8*	53.7	*11.5*	50.9	*13.6*	50.9	*11.1*	42.3	*22.5*	39.3	*20.3*
OCC	**77.2**	*3.7*	67.3	*1.0*	67.3	*−10.4*	49.2	*13.2*	45.1	*16.9*	48.7	*8.5*	38.2	*17.5*

Table 3. **Attribute-based performance**: Quality of the techniques on sequences with a certain attribute and the gain with respect to this quality in the sequences without it (in italics and smaller font). See DAVIS [37] for the meaning of the acronyms.

provement, and is faster than conventional methods such as adding a CRF on top of the segmentation [7].

Figure 5 further analyzes the type of errors that OSVOS produces (with and without boundary snapping), by dividing them into False Positives (FP) and False Negatives (FN). FP are further divided into close and far, setting the division at 20 pixels from the object. We can observe that the majority of the errors come from false negatives. Boundary snapping mainly reduces the false positives, both the ones close to the boundaries (more accurate contours) and the spurious detections far from the object, because they do not align with the trained generic contours.

Comparison to the State of the Art on DAVIS: Table 2 compares OSVOS to the rest of the state of the art. In terms of region similarity $\mathcal{J}$, OSVOS is 11.8 points above the second best technique and 19.8 above the third best. In terms of contour accuracy $\mathcal{F}$, OSVOS is 17.2 and 21.8 points above them. Our results are better than those obtained by an oracle selecting the best object proposal from the state-of-the-art object proposals COB. Even if the oracle would select the best set of superpixels to form each mask (COB|SP), OSVOS would be only 6.7 points below.

Table 3 shows an evaluation with respect to different attributes annotated in the DAVIS dataset, by comparing the performance of the methods on the sequences with a given attribute (challenge) versus the performance on those without it. OSVOS has the best performance on all attributes, and it has a significant resilience to these challenges (smallest decrease of performance when the attribute is present - numbers in italics).

Figure 6 shows the results per sequence compared to the state of the art. OSVOS has the best performance in the majority of sequences and is very close to the best in the rest. The results are especially impressive in sequences such as Drift-Chicane or Bmx-Trees, where the majority of techniques fail. Figure 7 shows the qualitative results on these two sequences. In the first row, the problem is especially challenging because of the smoke and the small initial size of the car. In the second row, OSVOS' *worse* sequence, despite vastly outperforming the rest of techniques. In this case, OSVOS loses track of the biker when he is occluded, but recovers when he is visible again. The rest of techniques lose the object because of the heavy occlusions.

Number of training images (parent network): To evaluate how much annotated data are needed to retrain a parent network, Table 4 shows the performance of OSVOS (-BS) when using a subset of the DAVIS train set. We randomly selected a fixed percentage of the annotated frames in each video. We conclude that by using only ~200 annotated frames, we are able to reach almost the same performance than when using the full DAVIS train split, thus not requiring full video annotations for the training procedure.

Training data	100	200	600	1000	2079
Quality ($\mathcal{J}$)	74.6	76.9	77.2	77.3	77.4

Table 4. **Amount of training data**: Region similarity ($\mathcal{J}$) as a function of the number of training images. Full DAVIS is 2079.

Timing: The computational efficiency of video object segmentation is crucial for the algorithms to be usable in practice. OSVOS can adapt to different timing requirements, providing progressively better results the more time

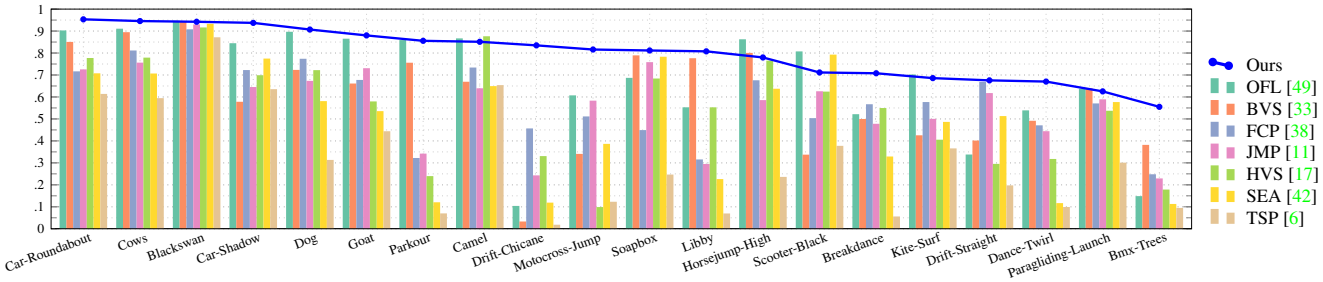


Figure 6. **DAVIS Validation**: Per-sequence results of region similarity ($\mathcal{J}$).

Figure 7. **Qualitative results**: First row, an especially difficult sequence which OSVOS segments well. Second row, OSVOS' worst result.

we can afford, by letting the fine-tuning algorithm at test time do more or fewer iterations. To show this behavior, Figure 8 shows the quality of the result with respect to the time it takes to process each 480p frame. As introduced before, OSVOS' time can be divided into the fine-tuning time plus the time to process each frame independently. The first mode we evaluate is -OS-BS (•), in which we do not fine-tune to the particular sequence, and thus use the parent network directly. In this case, the quality is not very good (although comparable to some previous techniques), but we only need to do a forward pass of the CNN for each frame.

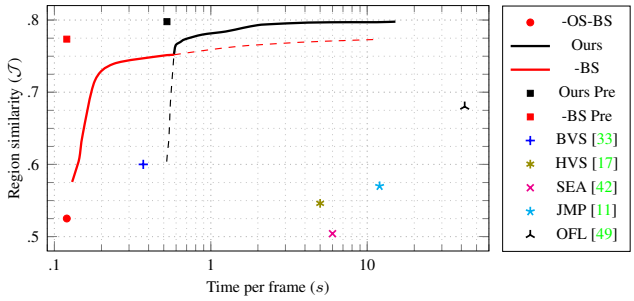


Figure 8. **Quality versus timing**: Region similarity with respect to the processing time per frame.

To take into account the fine-tuning time, we can consider two scenarios. First, in Ours (——) or -BS (——) we average the fine-tuning time (done once per sequence) over the length of that sequence. This way, the curves show the gain in quality with respect to the fine-tuning time, plus the forward pass on each frame. Using the same notation than in the ablation study, the two different curves refer to whether we do not perform boundary snapping (-BS) or we snap to the learned contours (Ours). The better results come at the price of adding the snapping cost so depending on the needed speed, one of the two can be chosen.

Since OSVOS processes frames independently, one could also perform the fine-tuning offline, by training on a picture of the object to be segmented beforehand (*e.g.* take a picture of a racing horse before the race). In this scenario, OSVOS can process each frame by one forward pass of the CNN (Ours Pre ▪, -BS Pre ▪), and so be considerably fast.

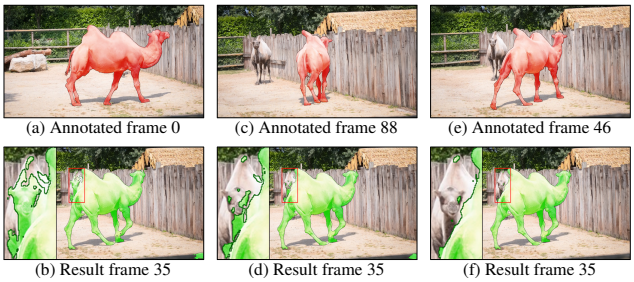
Compared to other techniques, OSVOS is significantly faster and/or more accurate at all regimes, from fast modes: 74.7 versus 60.0 of BVS (+) at 400 ms, and 79.8 versus 68.0 of OFL (٭) at lower speeds.

Refinement of results: Another advantage of our technique is that we can naturally incorporate more supervision in the form of more annotated frames. In a production environment, for instance, one needs a certain quality below which results are not usable. In this scenario, OSVOS can provide the results with one annotated frame, then the operator can decide whether the quality is good enough, and if not, segment another frame. OSVOS can then incorporate that knowledge into further fine-tuning the result.

To model this scenario, we take the results with N manual annotations, select the frame in which OSVOS performs worse, similarly to what an operator would do, *i.e.* select a frame where the result is not satisfactory; and add the ground-truth annotation into the fine-tuning. Table 5 shows the evolution of the quality when more annotations are added (0 means we test the parent network directly, i.e. zero-shot). We can see that the quality significantly increases from one to two annotations and saturates at around

Annotations	0	1	2	3	4	5	All
Quality ($\mathcal{J}$)	58.5	79.8	84.6	85.9	86.9	87.5	88.7

Table 5. **Progressive refinement**: Quality achieved with respect to the number of annotated frames OSVOS trains from.



(a) Annotated frame 0 (c) Annotated frame 88 (e) Annotated frame 46

(b) Result frame 35 (d) Result frame 35 (f) Result frame 35

Figure 9. **Qualitative incremental results**: The segmentation on frame 35 improves after frames 0, 88, and 46 are annotated.

five. As a measure of the upper bound of OSVOS, we fine-tuned on all annotated frames and tested on the same ones (last column), which indeed shows us that five annotated frames almost get the most out of this architecture.

Figure 9 shows a qualitative example of this process, where the user annotates frame 0, where only one camel is visible (a). In frame 35, OSVOS also segments the second camel that appears (b), which has almost the exact same appearance. This can be solved (f) by annotating two more frames, 88 (c) and 46 (e), which allows OSVOS to learn the difference between these two extremely similar objects, even without taking temporal consistency into account.

Evaluation as a tracker: Video object segmentation could also be evaluated as a Visual Object Tracking (VOT) [28] algorithm, by computing the bounding box around each of the segmentations. We compare to the winner of the VOT Challenge 2015 [28]: MDNET [32]. Since we cannot compare in the original dataset of the VOT Challenge (the ground-truth objects are not segmented so we cannot fine-tune on it), we run MDNET on DAVIS. Table 6 shows the percentage of bounding boxes coming from each technique that have an intersection over union with the ground-truth bounding box above different thresholds. The higher the threshold, the more alignment with the ground truth is required. We can see that OSVOS has significant better results as tracker than MDNET at all regimes, with more margin at higher thresholds.

Results on Youtube-Objects: For completeness, we also do experiments on Youtube-objects [41, 20], where we take the pre-computed evaluation from other papers. Table 7 shows that we perform slightly better than the state of the art OFL, which is significantly slower, and despite the fact that the sequences in this database have significant less occlusions and motion than in DAVIS, which favors techniques that enforce temporal consistency.

Overlap	0.5	0.6	0.7	0.8	0.9
Ours	**78.2**	**72.2**	**65.8**	**59.4**	**49.6**
MDNET [32]	66.4	57.8	43.4	29.5	14.7

Table 6. **Evaluation as a tracker**: Percentage of bounding boxes that match with the ground truth at different levels of overlap.

Category	Ours	OFL	JFS	BVS	SCF	AFS	FST	HBT	LTV
Aeroplane	88.2	**89.9**	89.0	86.8	86.3	79.9	70.9	73.6	13.7
Bird	**85.7**	84.2	81.6	80.9	81.0	78.4	70.6	56.1	12.2
Boat	**77.5**	74.0	74.2	65.1	68.6	60.1	42.5	57.8	10.8
Car	79.6	**80.9**	70.9	68.7	69.4	64.4	65.2	33.9	23.7
Cat	**70.8**	68.3	67.7	55.9	58.9	50.4	52.1	30.5	18.6
Cow	77.8	**79.8**	79.1	69.9	68.6	65.7	44.5	41.8	16.3
Dog	**81.3**	76.6	70.3	68.5	61.8	54.2	65.3	36.8	18.0
Horse	**72.8**	72.6	67.8	58.9	54.0	50.8	53.5	44.3	11.5
Motorbike	73.5	**73.7**	61.5	60.5	60.9	58.3	44.2	48.9	10.6
Train	75.7	76.3	**78.2**	65.2	66.3	62.4	29.6	39.2	19.6
Mean	**78.3**	77.6	74.0	68.0	67.6	62.5	53.8	46.3	15.5

Table 7. **Youtube-Objects evaluation**: Per-category mean intersection over union ($\mathcal{J}$).

5. Conclusions

Deep learning approaches often require a huge amount of training data in order to solve a specific problem such as segmenting an object in a video. Quite in contrast, human observers can solve similar challenges with only a single training example. In this paper, we demonstrate that one can reproduce this capacity of one-shot learning in a machine: Based on a network architecture pre-trained on generic datasets, we propose One-Shot Video Object Segmentation (OSVOS) as a method which fine-tunes it on merely one training sample and subsequently outperforms the state-of-the-art on DAVIS by 11.8 points. Interestingly, our approach does not require explicit modeling of temporal consistency using optical flow algorithms or temporal smoothing and thus does not suffer from error propagation over time (drift). Instead, OSVOS processes each frame of the video independently and gives rise to highly accurate and temporally consistent segmentations. All resources of this paper can be found at www.vision.ee.ethz.ch/~cvlsegmentation/osvos/

Acknowledgements: Research funded by the EU Framework Programme for Research and Innovation Horizon 2020 (Grant No. 645331, EurEyeCase), the Swiss Commission for Technology and Innovation (CTI, Grant No. 19015.1 PFES-ES, NeGeVA), and the ERC Consolidator Grant "3D Reloaded". The authors gratefully acknowledge support by armasuisse and thank NVidia Corporation for donating the GPUs used in this project.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 3, 5

[2] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016. 4

[3] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *ICCV*, 2015. 1

[4] G. Bertasius, J. Shi, and L. Torresani. Semantic segmentation with boundary neural fields. In *CVPR*, 2016. 1

[5] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 2, 5

[6] J. Chang, D. Wei, and J. W. Fisher III. A video representation using temporal superpixels. In *CVPR*, 2013. 2, 3, 5, 7

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 6

[8] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 2, 3

[9] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 2

[10] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014. 5

[11] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. Jumpcut: Non-successive mask transfer and interpolation for video cutout. *ACM Trans. Graph.*, 34(6), 2015. 2, 3, 5, 7

[12] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *TPAMI*, 35(8):1915–1929, 2013. 2, 3

[13] K. Fragkiadaki, G. Zhang, and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *CVPR*, 2012. 5

[14] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 3

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[16] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *CVIU*, 117(10):1245–1256, 2013. 5

[17] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 2, 3, 5, 7

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2, 4

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3, 4

[20] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *ECCV*, 2014. 2, 5, 8

[21] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017. 3

[22] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017. 3

[23] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *ICLR*, 2016. 1, 4

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3, 4

[25] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011. 5

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. SSD: Single shot multibox detector. In *ECCV*, 2016. 1

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3, 4

[28] M. Kristan et al. The visual object tracking VOT2015 challenge results. In *Visual Object Tracking Workshop 2015 at ICCV 2015*, Dec 2015. 8

[29] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. In *ECCV*, 2016. 1, 3, 4, 5

[30] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Deep retinal image understanding. In *MICCAI*, 2016. 2, 3, 4

[31] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014. 5

[32] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 3, 8

[33] N. Nicolas Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *CVPR*, 2016. 2, 5, 7

[34] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 2, 3

[35] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *TPAMI*, 36(6):1187–1200, 2014. 5

[36] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 5

[37] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 2, 5, 6

[38] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, 2015. 2, 5, 7

[39] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 3

[40] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *TPAMI*, 2017. 3, 5

[41] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 2, 5, 8

[42] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, 2014. 2, 3, 5, 7

[43] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1

[45] N. Shankar Nagaraja, F. R. Schmidt, and T. Brox. Video segmentation with just a few strokes. In *ICCV*, 2015. 5

[46] J. Shen, W. Wenguan, and F. Porikli. Saliency-Aware geodesic video object segmentation. In *CVPR*, 2015. 5

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 3, 4

[48] B. Taylor, V. Karasev, and S. Soatto. Causal video object segmentation from persistence of occlusions. In *CVPR*, 2015. 5

[49] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, 2016. 2, 5, 7

[50] S. Vijayanarasimhan and K. Grauman. Active frame selection for label propagation in videos. In *ECCV*, 2012. 5

[51] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 2, 3, 4

[52] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016. 2, 3, 4