Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning

Yuhua Chen[1] Jordi Pont-Tuset[1] Alberto Montes[1] Luc Van Gool[1,2]

[1]Computer Vision Lab, ETH Zurich [2]VISICS, ESAT/PSI, KU Leuven

{yuhua.chen,jponttuset,vangool}@vision.ee.ethz.ch, malberto@student.ethz.ch

Abstract

This paper tackles the problem of video object segmentation, given some user annotation which indicates the object of interest. The problem is formulated as pixel-wise retrieval in a learned embedding space: we embed pixels of the same object instance into the vicinity of each other, using a fully convolutional network trained by a modified triplet loss as the embedding model. Then the annotated pixels are set as reference and the rest of the pixels are classified using a nearest-neighbor approach. The proposed method supports different kinds of user input such as segmentation mask in the first frame (semi-supervised scenario), or a sparse set of clicked points (interactive scenario). In the semi-supervised scenario, we achieve results competitive with the state of the art but at a fraction of computation cost (275 milliseconds per frame). In the interactive scenario where the user is able to refine their input iteratively, the proposed method provides instant response to each input, and reaches comparable quality to competing methods with much less interaction.

1. Introduction

Immeasurable amount of multimedia data is recorded and shared in the current era of the Internet. Among it, video is one of the most common and rich modalities, albeit it is also one of the most expensive to process. Algorithms for fast and accurate video processing thus become crucially important for real-world applications. Video object segmentation, i.e. classifying the set of pixels of a video sequence into the object(s) of interest and background, is among the tasks that despite having numerous and attractive applications, cannot currently be performed in a satisfactory quality level and at an acceptable speed. The main objective of this paper is to fill in this gap: we perform video object segmentation at the accuracy level comparable to the state of the art while keeping the processing time at a speed that even allows for real-time human interaction.

Towards this goal, we model the problem in a simple and intuitive, yet powerful and unexplored way: we formulate video object segmentation as pixel-wise retrieval in a learned embedding space. Ideally, in the embedding space, pixels belonging to the same object instance are close together and pixels from other objects are further apart. We build such embedding space by learning a Fully Convolutional Network (FCN) as the embedding model, using a modified triplet loss tailored for video object segmentation, where no clear correspondence between pixels is given. Once the embedding model is learned, the inference at test-time only needs to compute the embedding vectors with a forward pass for each frame, and then perform a per-pixel nearest neighbor search in the embedding space to find the most similar annotated pixel. The object, defined by the user annotation, can therefore be segmented throughout the video sequence.

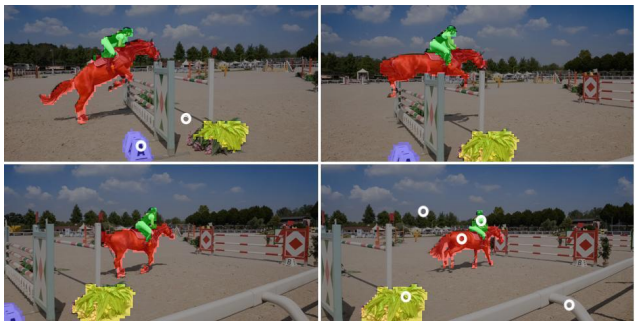

Figure 1. **Interactive segmentation using our method:** The white circles represent the clicks where the user has provided an annotation, the colored masks show the resulting segmentation in a subset of the sequence's frames.

There are several main advantages of our formulation: Firstly, the proposed method is highly efficient as there is no fine-tuning in test time, and it only requires a single forward pass through the embedding network and a nearest-neighbor search to process each frame. Secondly, our method provides the flexibility to support different types of user input (i.e. clicked points, scribbles, segmentation masks, etc.) in an unified framework. Moreover, the embedding process is independent of user input, thus the embedding vectors do not need to be recomputed when the user input changes, which makes our method ideal for the interactive scenario.

We show an example in Figure 1, where the user aims to segment several objects in the video: The user can iteratively refine the segmentation result by gradually adding more clicks on the video, and get feedback immediately after each click.

The proposed method is evaluated on the DAVIS 2016 [26] and DAVIS 2017 [29] datasets, both in the semi-supervised and interactive scenario. In the context of semi-supervised Video Object Segmentation (VOS), where the full annotated mask in the first frame is provided as input, we show that our algorithm presents the best trade-off between speed and accuracy, with 275 milliseconds per frame and $\mathcal{J}\&\mathcal{F}$=77.5% on DAVIS 2016. In contrast, better performing algorithms start at 8 seconds per frame, and similarly fast algorithms reach only 60% accuracy. Where our algorithm shines best is in the field of interactive segmentation, with only 10 clicks on the whole video we can reach an outstanding 74.5% accuracy.

2. Related Work

Semi-Supervised and Unsupervised Video Object Segmentation:
The aim of video object segmentation is to segment a specific object throughout an input video sequence. Driven by the surge of deep learning, many approaches have been developed and performance has improved dramatically. Dependent on the amount of supervision, methods can be roughly categorized into two groups: semi-supervised and unsupervised.

Semi-supervised video object segmentation methods take the segmentation mask in the first frame as input. MaskTrack [25] propagates the segmentation from the previous frame to the current one, with optical flow as input. OSVOS [3] learns the appearance of the first frame by a FCN, and then segments the remaining frames in parallel. Follow-up works extend the idea with various techniques, such as online adaptation [39], semantic instance segmentation [2, 22]. Other recent techniques obtain segmentation and flow simultaneously [8, 38], train a trident network to improve upon the errors of optical flow propagation [18], or use a CNN in the bilateral space [17].

Unsupervised video object segmentation, on the other hand, uses only video as input. These methods typically aim to segment the most salient object from cues such as motion and appearance. The current leading technique [19] use region augmentation and reduction to refine object proposals to estimate the primary object in a video. [16] proposes to combine motion and appearance cues with a two-stream network. Similarly, [37] learns a two-stream network to encode spatial and temporal features, and a memory module to capture the evolution over time.

In this work, we focus on improving the efficiency of video object segmentation to make it suitable for real-world applications where rapid inference is needed. We do so by, in contrast to previous techniques using deep learning, not performing test-time network fine-tuning and not relying on optical flow or previous frames as input.

Interactive Video Object Segmentation:
Interactive Video Object Segmentation relies on iterative user interaction to segment the object of interest. Many techniques have been proposed for the task. Video Cutout [40] solves a min-cut labeling problem over a hierarchical mean-shift segmentation of the set of video frames, from user-generated foreground and background scribbles. The pre-processing plus post-processing time is in the order of an hour, while the time between interactions is in the order of tens of seconds. A more local strategy is LIVEcut [30], where the user iteratively corrects the propagated mask frame to frame and the algorithm learns from it. The interaction response time is reduced significantly (seconds per interaction), but the overall processing time is comparable. TouchCut [41] simplifies the interaction to a single point in the first frame, and then propagates the results using optical flow. *Click carving* [15] uses point clicks on the boundary of the objects to fit object proposals to them. *A few strokes* [23] are used to segment videos based on point trajectories, where the interaction time is around tens of seconds per video. A click-and-drag technique [28] is used to label per-frame regions in a hierarchy and then propagated and corrected.

In contrast to most previous approaches, our method response time is almost immediate, and the pre-processing time is 275 milliseconds per frame, making it suitable to real-world use.

Deep Metric Learning:
Metric learning is a classical topic and has been widely studied in the learning community [43, 4]. Following the recent success of deep learning, deep metric learning has gained increasing popularity [36], and has become the cornerstone of many computer vision tasks such as person re-identification [7, 44], face recognition [33], or unsupervised representation learning [42]. The key idea of deep metric learning is usually to transform the raw features by a network and then compare the samples in the embedding space directly. Usually metric learning is performed to learn the similarity between images or patches, and methods based on pixel-wise metric learning are limited. Recently, [11] exploits metric learning at the pixel level for the task of instance segmentation.

In this work, we learn an embedding where pixels of the same instance are aimed to be close to each other, and we formulate video object segmentation as a pixel-wise retrieval problem. The formulation is inspired also by works in image retrieval [35, 31].

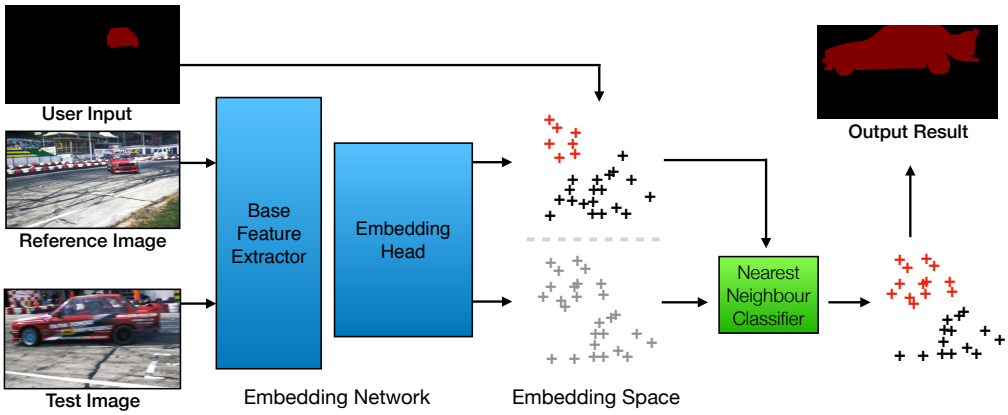

Figure 2. **Overview of the proposed approach:** Here we assume the user input is provided in the form of full segmentation mask for the reference frame, but interactions of other kind are supported as well.

3. Proposed Method

3.1. Overview

In this work, we formulate video object segmentation as a pixel-wise retrieval problem, that is, for each pixel in the video, we look for the most similar reference pixel in the embedding space and assign the same label to it. The proposed method is sketched in Figure 2. Our method consists of two stages when processing a new video: we first embed each pixel into a d-dimensional embedding space using the proposed embedding network. Then the second step is to perform per-pixel retrieval in this space to transfer labels to each pixel according to its nearest reference pixel.

A key aspect of our approach, which allows for a fast user interaction, is our way of incorporating the user input. Alternative approaches have been exploited to inject user input into deep learning systems:

User input to fine-tune the model: The first way is to fine-tune the network to the specific object based on the user input. For example, techniques such as OSVOS [3] or MaskTrack [25] fine-tune the network at test time based on the user input. When processing a new video, they require many iterations of training to adapt the model to the specific target object. This approach can be time-consuming (seconds per sequence) and therefore impractical for real-time applications, especially with a human in the loop.

User input as the network input: Another way of injecting user interaction is to use it as an additional input to the network. In this way, no training is performed at test time. Such methods typically either directly concatenate the user input with the image [45], or use a sub-network to encode the user input [34, 46]. A drawback of these methods is that the network has to be recomputed once the user input changes. This can still be a considerable amount of time, especially for video, considering the large number of frames.

In contrast to previous methods, in this work user input is disentangled from the network computation, thus the forward pass of the network needs to be computed only once. The only computation after user input is then a nearest-neighbor search, which is very fast and enables rapid response to the user input.

3.2. Segmentation as Pixel-wise Retrieval

For clarity, here we assume a single-object segmentation scenario, and the segmentation mask of first frame is used as user input. The discussion is, however, applicable for multiple objects and for other types of inputs as well.

The task of semi-supervised video object segmentation is defined as follows: segmenting an object in a video given the object mask of the first frame. Formally, let us denote the *i*-th pixel in the *j*-th frame of the input video as $x_{j,i}$. The user provides the annotation for the first frame: $(x_{1,i}, l_{1,i})$, where $l \in \{0, 1\}$, and $l_{1,i} = 0, 1$ indicates $x_{1,i}$ belongs to background and foreground, respectively. We refer to these annotated pixels as reference pixels. The goal is then to infer the labels of all the unlabeled pixels in other frames $l_{j,i}$ with $j > 1$.

Embedding Model:
We build an embedding model f and each pixel $x_{j,i}$ is represented as a d-dimensional embedding vector $e_{j,i} = f(x_{j,i})$. Ideally, pixels belonging to the same object are close to each other in the embedding space, and pixels belonging to different objects are distant to each other. In more detail, our embedding model is build on DeepLab-v2 [5] based on the ResNet101 [14] backbone architecture. First, we pre-train the network for semantic segmentation on COCO [20] using the same procedure presented in [5] and then we remove the final classification layer and replace it with a new convolutional layer with d output channels. We fine-tune the network to learn the embedding for video object segmentation, which will be detailed in Section 3.3. To avoid confusion,

we refer to the the original DeepLab-v2 architecture as *base feature extractor* and to the two convolutional layers as *embedding head*. The resulting network is fully convolutional, thus the embedding vector of all pixels in a frame can be obtained in a single forward pass. For an image of size $h \times w$ pixels the output is a tensor $[h/8, w/8, d]$, where d is the dimension of the embedding space. We use $d = 128$ unless otherwise specified. The tensor is 8 times smaller due to that the network has a stride length of 8 pixels.

Since an FCN is deployed as the embedding model, spatial and temporal information are not kept due to the translation invariance nature of the convolution operation. However, such information is obviously important for video and should not be ignored when performing segmentation. We circumvent this problem with a simple approach: we add the spatial coordinates and frame number as additional inputs to the embedding head, thus making it aware of spatial and temporal information. Formally, the embedding function can be represented as $e_{j,i} = f(x_{j,i}, i, j)$, where i and j refer to the ith pixel in frame j. This way, spatial information i and temporal information j can also be encoded in the embedding vector $e_{j,i}$.

Retrieval with Online Adaptation:
During inference, video object segmentation is simply performed by retrieving the closer reference pixels in the embedded space. We deploy a k-Nearest Neighbors (kNN) classifier which finds the set of reference pixels whose feature vector e_i^j is closer to the feature vector of the pixels to be segmented. In the experiments, we set $k = 5$ for the semi-supervised case, and $k = 1$ for the interactive segmentation case. Then, the identity of the pixel is computed by a majority voting of the set of closer reference pixels. Since our embedding model operates with a stride of 8, we upsample our results to the original image resolution by the bilateral solver [1].

A major challenge for semi-supervised video object segmentation is that the appearance changes as the video progresses. The appearance change causes severe difficulty for a fixed model learned in the first frame. As observed in [39, 6], such appearance shift usually leads to a decrease in performance for FCNs. To cope with this issue, OnAVOS[39] proposes to update the model using later frames where their prediction is very confident. In order to update their model online, however, they have to run a few iterations of the fine-tuning algortihm using highly confident samples, which makes their method even slower than the original OSVOS.

This issue can also be understood as the sample distribution shifts in the embedding space over time. In this work, we can easily update the model online to capture the appearance change, a process that is nearly effortless. In particular we initialize the pool of reference samples with the samples that the user have annotated. As the video progresses, we gradually add samples with high confidence to the pool of reference samples. We add the samples into our reference pool if all of its $k = 5$ near neighbors agree with the label.

Generalization to different user input modes and multiple objects:
So far we focused on single-object scenarios where user interaction is provided as the full object mask in the first frame. However, multiple object might be present in the video, and the user input might be in an arbitrary form other than the full mask of the first frame. Our method can be straightforwardly applicable to such cases.

In a general case, the input from user can be represented as a set of pixels and its corresponding label: $\{x_{i,j}, l_{i,j}\}$ without need for all inputs to be on the first frame ($j = 1$) or the samples to be exhaustive (covering all pixels of one frame). Please note that the latter is in contrast to the majority of semi-supervised video object segmentation techniques, which assume a full annotated frame to segment the object from the video.

In our case, the input $x_{i,j}$ can be in the form of clicked points, drawn scribbles, or others possibilities. The label $l_{i,j}$ can also be an integer $l_i^j \in \{1...K\}$ representing an identifier of an object within a set of K objects, thus generalizing our algorithm to multiple-object video segmentation.

3.3. Training

The basic idea of metric learning is to *pull* similar samples close together and *push* dissimilar points far apart in the embedding space. A proper training loss and sampling strategy are usually of critical importance to learn a robust embedding. Below we present our training loss and sampling strategy specifically designed for video object segmentation.

Training loss:
In the metric learning literature, contrastive loss [9, 13], triplet loss [4], and their variants are widely used for metric learning. We argue, however, and verify in our experiments, that the standard losses are not suitable for the task at hand, i.e. video object segmentation, arguably due to the intra-object variation present in a video. In other words, the triplet loss is designed for the situation where the identity of the sample is clear, which is not the case for video object segmentation as an object can be composed of several parts, and each part might have very different appearance. Pulling these samples close to each other, therefore, is an extra constraint that can be harmful for learning a robust metric. We illustrate this effect with an example in Figure 3.

Keeping this in mind, we modify the standard triplet loss to adapt it to our application. Formally, let us refer to anchor sample as x^a. $x^p \in \mathcal{P}$ is a positive sample from a positive

Measure		OnAVOS	OSVOS	MSK	PML (Ours)	SFL	CTN	VPN	OFL	BVS	FCP	JMP	HVS	SEA
$\mathcal{J}\&\mathcal{F}$	Mean $\mathcal{M}\uparrow$	**85.5**	80.2	77.5	77.4	76.0	71.4	67.8	65.7	59.4	53.8	55.1	53.8	49.2
$\mathcal{J}$	Mean $\mathcal{M}\uparrow$	**86.1**	79.8	79.7	75.5	76.1	73.5	70.2	68.0	60.0	58.4	57.0	54.6	50.4
	Recall $\mathcal{O}\uparrow$	**96.1**	93.6	93.1	89.6	90.6	87.4	82.3	75.6	66.9	71.5	62.6	61.4	53.1
	Decay $\mathcal{D}\downarrow$	5.2	14.9	8.9	8.5	12.1	15.6	12.4	26.4	28.9	**-2.0**	39.4	23.6	36.4
$\mathcal{F}$	Mean $\mathcal{M}\uparrow$	**84.9**	80.6	75.4	79.3	76.0	69.3	65.5	63.4	58.8	49.2	53.1	52.9	48.0
	Recall $\mathcal{O}\uparrow$	89.7	92.6	87.1	**93.4**	85.5	79.6	69.0	70.4	67.9	49.5	54.2	61.0	46.3
	Decay $\mathcal{D}\downarrow$	5.8	15.0	9.0	7.8	10.4	12.9	14.4	27.2	21.3	**-1.1**	38.4	22.7	34.5
$\mathcal{T}$	Mean $\mathcal{M}\downarrow$	19.0	37.8	21.8	47.0	18.9	22.0	32.4	22.2	34.7	30.6	15.9	36.0	**15.4**

Table 1. **Evaluation results on DAVIS 2016 validation set set:** We compare the proposed method with an exhaustive set of very recent techniques.

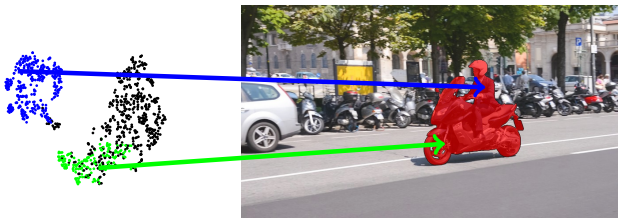

Figure 3. **Illustration of pixel-wise feature distribution:** Green denotes pixels from motorbike, blue represents person, and black background. The object of interest in this video and the annotation is the human and the motorbike. However, features from motorbike and person lie in two clusters in the feature space. Pulling these two cluster close might be harmful for the metric learning. Visualization is done by t-SNE [21].

sample pool $\mathcal{P}$. Similarly, x^n denotes a negative sample and $\mathcal{N}$ denotes the negative pool. The standard triplet loss pushes the negative points further away than the distance between anchor and positive points. Since we do not want to pull every pair of positive points close (different parts of an object that look different), we modify the loss to only push the smallest negative points further than the smallest positive points, the loss can thus be represented as:

$$\sum_{x^a \in \mathcal{A}} \{ \min_{x^p \in \mathcal{P}} \|f(x^a) - f(x^p)\|_2^2 - \min_{x^n \in \mathcal{N}} \|f(x^a) - f(x^n)\|_2^2 + \alpha \} \quad (1)$$

where α is the slack variable to control the margin between positive and negative samples, as in the standard formulation, and we denote the set of anchors as $\mathcal{A}$.

For each anchor sample x^a we have two pools of samples: one pool of positive samples $\mathcal{P}$, whose labels are consistent with the anchor and another pool of negative examples $\mathcal{N}$, whose labels are different from the anchor sample. We take the closest sample to the anchor in each pool, and we compare the positive distance and negative distance. Intuitively, the loss *pushes* only the closest negative away, while keeping the closest positive closer.

Training Strategy:
During training, we have fully annotated videos available (object segmentation on each frame). To form a valid triplet to train from, to be used in the aforementioned loss, we need to sample an anchor point x^a, a positive sample pool $\mathcal{P}$ and a negative sample pool $\mathcal{N}$. For this purpose, three frames are randomly sampled from the training video: from one we sample anchor points and the pixels from the other two frames are joined together. From those, the pixels that have the same label than the anchor form the positive pool $\mathcal{P}$, and the rest form the negative pool $\mathcal{N}$. Note that the pools are sampled from two different frames to have temporal variety, which is needed for the embedding head to learn to weight the temporal information from the feature vector. Also, we do not use pixels from the the anchor frame in the pools to avoid too easy samples.

In each iteration, a forward pass is performed on three randomly selected frames with one frame as the anchor. Then the anchor frame is used to sample 256 anchor samples, and the positive and negative pools are all foreground and background pixels in the other two frames. We compute the loss according to Equation 1 and the network is trained in an end to end manner.

4. Experimental Validation

We evaluate the proposed method mainly on DAVIS 2016 [26], a dataset containing 50 full high-definition videos annotated with pixel-level accurate object masks (one per sequence) densely on all the frames. We train our model on the 30 training videos and report the results on the validation set, consisting of 20 videos. We perform experiments with multiple objects in DAVIS 2017 [29], an extension of the former to 150 sequences and multiple objects.

4.1. Semi-supervised VOS on DAVIS

We first consider the semi-supervised scenario defined in DAVIS 2016, where the methods are given the full segmentation of the object in the first frame and the goal is to segment the rest of the frames.

We compare against an exhaustive set of very recent techniques: OnAVOS [39], OSVOS [3], MSK [25], SFL [8], CTN [18], VPN [17], OFL [38], BVS [24],

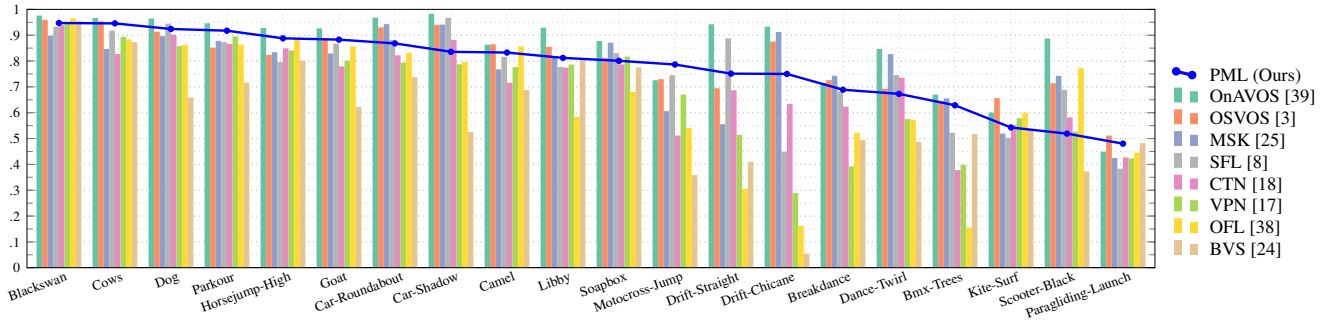

Figure 4. : Per-sequence results of mean region similarity ($\mathcal{J}$) and contour accuracy ($\mathcal{F}$). The rest of the state-of-the-art techniques are shown using bars, ours is shown using a line. Sequences are sorted by our performance.

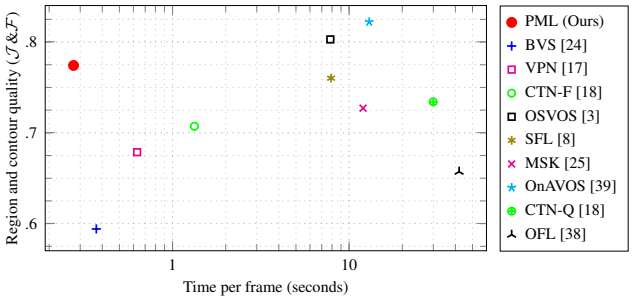

Figure 5. **Quality versus timing in DAVIS 2016**: $\mathcal{J}$&$\mathcal{F}$ of all techniques with respect to their time to process one frame. The timing is taken from each paper. OnAVOS and MSK do not report their timings with the post-processing steps that lead to the most accurate results, so we compare to the version with reported times.

FCP [27], JMP [10], HVS [12], and SEA [32]; using the pre-computed results available on the DAVIS website and the metrics proposed in DAVIS ($\mathcal{J}$ Jaccard index or IoU, $\mathcal{F}$ boundary accuracy, $\mathcal{T}$ temporal stability). Readers are referred to each paper for more details.

Table 1 shows the comparison to the rest of the state of the art, i.e. at the best-performing regime (and slowest) of all techniques. In global terms ($\mathcal{J}$&$\mathcal{F}$), PML (Ours) is comparable to MSK and only behind OSVOS and OnAVOS, which are significantly slower, as we will show in the next experiment. Our technique is especially competitive in terms of boundary accuracy ($\mathcal{F}$), despite there is no refinement or smoothing step explicitly tackling this feature as in other methods.

To analyze the trade off between quality and performance, Figure 5 plots the quality of each technique with respect to their mean time to process one frame (in 480p resolution). Our technique presents a significantly better trade off than the rest of techniques. Compared to the fastest one (BVS), we perform +18 points better while still being 100 milliseconds faster. Compared to the technique with more accurate results (OnAVOS), we lose 5 points but we process each frame 43× faster.

Figure 4 breaks the performance into each of the 20 sequences of DAVIS validation. We can observe that we are close to the best performance in the majority of the sequences, we obtain the best result in some of them, and our worst performance is 0.5, which shows the robustness of the embedding over various challenges and scenarios.

Figure 6 displays the qualitative results of our technique on a homogeneous set of sequences, from the ones in which we perform the best to those more challenging. Please note that in sequences Bmx-Trees (last row) and Libby (third row), our method is very robust to heavy occlusions, which is logical since we do not perform any type of temporally-neighboring propagation. Results also show that our method is robust to drastic changes in foreground scale and appearance (Motocross-Jump - fourth row) and to background appearance changes (Parkour - second row). Sequences Motocross-Jump, and BMX-Trees (fourth, and last row) show a typical failure mode (which is also observed in other techniques such as OSVOS) in which foreground objects that were not seen in the first frames are classified as foreground when they appear.

	Contrastive Loss	Triplet Loss	Proposed Loss
Mean $\mathcal{J}$	66.1	69.5	**75.5**
Mean $\mathcal{F}$	68.5	73.5	**79.3**
Mean $\mathcal{J}$&$\mathcal{F}$	67.3	71.5	**77.4**

Table 2. Ablation Study on Different Losses

4.2. Ablation Study

In this section we analyze the relative importance of each proposed component, by evaluating ablated versions of our method.

Training Losses for Metric Learning:
As discussed in Section 3.3, our embedding model is optimized using a modified version of the triplet loss. To verify the design, we compare our model with two others trained with the original contrastive loss and triplet loss, respectively; while keeping the other settings unchanged. First, we briefly describe the different losses tested:

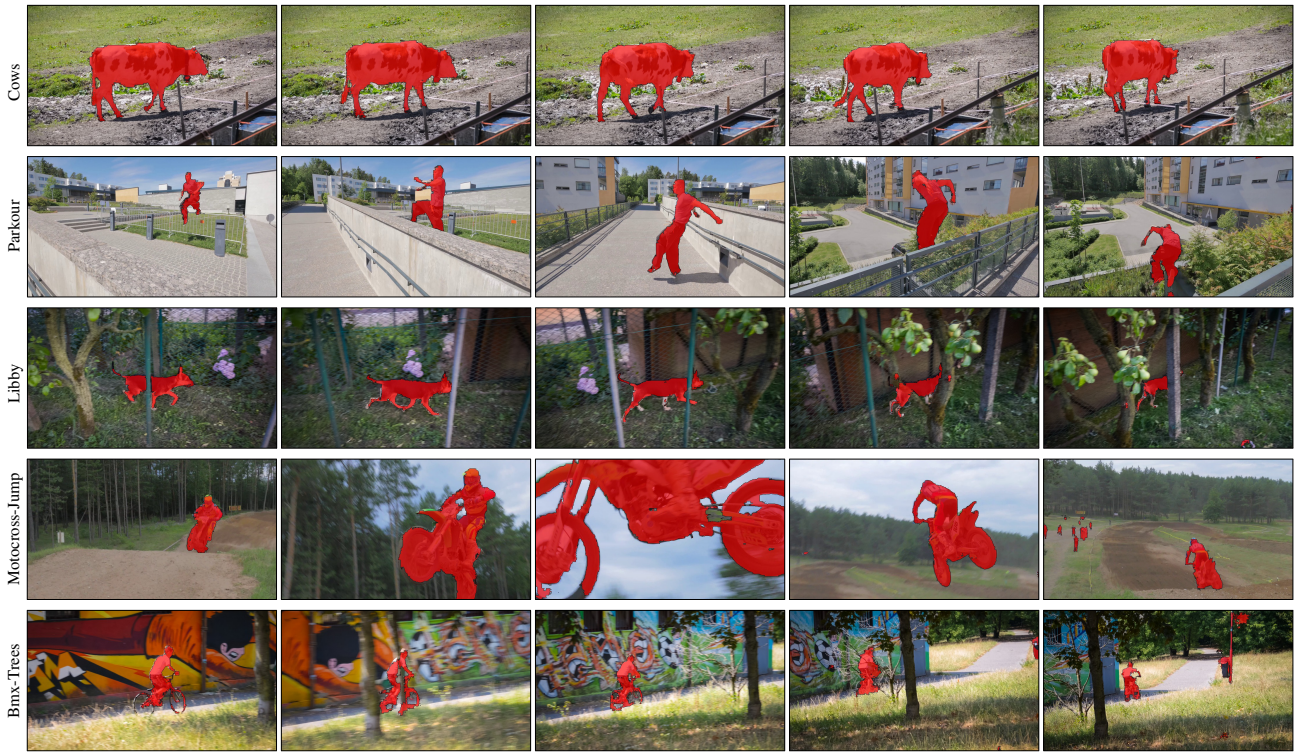

Figure 6. **Qualitative results**: Homogeneous sample of DAVIS sequences with our result overlaid.

The **contrastive loss** operates on pairs of samples and can be written as:

$$L_{contra} = \sum_{i}^{N} \left\{ (y) d^2 + (1-y) \max(\alpha - d, 0)^2 \right\}$$

where y is the label of the pair ($y = 0$ indicates that the pairs have different identities and $y = 1$ otherwise), $d = \|x_i - x_j\|$ is the distance between two points, and α is a slack variable to avoid negative points being overly penalized. The loss minimizes the distance between samples if $y = 1$, and maximizes it if $y = 0$.

The **triplet loss** shares a similar spirit with contrastive loss, but using three samples as a unit. Each triplet is composed of three samples: one as anchor x_a, one positive x_p, and one negative x_n. The positive (negative) sample has the same (different) label than the anchor. The loss is then defined as:

$$L = \sum_{i}^{N} \left\{ \|f(x_a) - f(x_p)\|_2^2 - \|f(x_a) - f(x_n)\|_2^2 + \alpha \right\}$$

where again α is a slack variable to control the margin.

Table 2 compares our embedding model with the models trained with the alternative losses. The results clearly show that the proposed loss achieves better performance than the alternatives.

Spat.-Temp.	Online Adapt.	Mean $\mathcal{J}$	Mean $\mathcal{F}$	Mean $\mathcal{J}\&\mathcal{F}$
		72.0	73.6	72.8
	✓	73.2	75.0	74.1
✓		74.3	78.1	76.2
✓	✓	**75.5**	**79.3**	**77.4**

Table 3. Ablation study on online adaptation and spatio-temporal embedding

Spatio-Temporal-Aware Embedding and Online Adaptation:
We proceed with our ablation analysis by studying the individual impact of two major sub-components: online adaptation and spatial and temporal awareness, as presented in Section 3.2.

Table 3 presents our ablation study on each component: online adaptation provides a slight boost of $+1.2\%$ in $\mathcal{J}$. Bringing in spatial and temporal information gives $+2.3\%$ improvement in $\mathcal{J}$ and $+4.5\%$ in $\mathcal{F}$ which validates the importance of spatial and temporal information for video object segmentation. Combining both results gives the best performance of 75.5% in overlap, which is overall $+3.5\%$ higher at nearly no extra cost.

4.3. Interactive Video Object Segmentation

Getting dense annotations in the first frame is a laborious and expensive process. It is therefore highly desirable

that a system can interact with users in a more realistic way and reach the target quality with as little effort as possible. Our system allows users to interact with the system in real time, and see the result immediately after their input. In this section we consider the scenario of interactive video object segmentation, where the users are allowed to annotate any frame. The process is iterated and the user decides how to annotate based on the result up to the given point.

For the sake of simplicity, we limit the interaction to clicks: users can click the object of interest or the background. This way, the amount of interaction can easily be quantified as number of clicks. Please note though, that other types of interactions such as scribbles are also naturally supported by our system, although more difficult to evaluate in this experiment.

We first simulate the user behavior by a *robot*. The robot randomly selects one pixel from the foreground and one pixel from the background as the first annotations, thus the nearest neighbor search can be performed. After having the initial result, the robot iteratively refines the segmentation result by randomly selecting from the pixels where the predicted label is wrong, and correcting its label based on the ground-truth.

The left side of Figure 7 (—) shows the evolution of the quality of the result as more clicks are provided. We achieve an overlap of $\mathcal{J} = 80\%$ with only 0.55 clicks per frame, and the performance goes up to $\mathcal{J} = 83\%$ with 2 clicks per frame. Our method achieves the same result as when providing the full mask on the first frame ($\mathcal{J} = 75.5\%$) using only 0.15 clicks per frame. Due to the randomness of our experiment, each experiment is repeated for 5 times and we report the average overlap. We find the variance to be only 0.1 at 1 click per frame, which suggests that our method is reasonably robust to the selection of points.

To verify that the simulated clicks are realistic, we carry out a user study on *real* users, where we ask them to click freely until they are happy with the segmentation. The results are shown as points (+) in Figure 7. We can see that the real-user results are slightly better than the simulated ones, which we attribute to the fact that a real user can choose which point to click based on a global view (for instance, select the worst frame) instead of the random sampling that the robot performs.

On average, the user did 0.17 clicks per frame to achieve an overall result of $\mathcal{J} = 77.7\%$. This equals to 11 clicks per video, which takes around 24 seconds. In contrast, a user takes 79 seconds to segment an object at the MS COCO quality [20], so the full mask of the first frame at the quality of DAVIS can safely be estimated to take over 3 minutes. The quality achieved in these 24 seconds is comparable with most state-of-the-art semi-supervised methods, but at a fraction of the annotation and running cost.

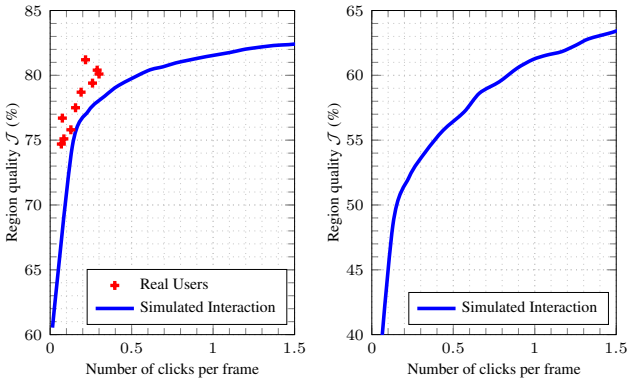

Figure 7. **Interactive Segmentation Results**: Achieved quality with respect to the number of clicks provided in the single-object (left) on DAVIS 2016 and multiple-object (right) scenarios on DAVIS 2017.

4.4. Extension to Multiple Objects

As discussed in Section 3.2, our method can naturally extend to the segmentation of multiple objects. To validate the effectiveness of our method in such scenario, we carry out experiments on DAVIS 2017 [29], where each video has multiple objects, usually interacting with and occluding each other.

We summarize our results in the right side of Figure 7: our method generalizes well to multiple objects and the results are comparable with most state-of-the-art methods. For instance, OSVOS achieves 57% in $\mathcal{J}$. We match their results by only 0.5 clicks per frame, which leads to a fraction of the processing time of the former.

5. Conclusions

This work presents a conceptually simple yet highly effective method for video object segmentation. The problem is casted as a pixel-wise retrieval in an embedding space learned via a modification of the triplet loss specifically designed for video object segmentation. This way, the annotated pixels on the video (via scribbles, segmentation on the first mask, clicks, etc.) are the reference samples, and the rest of pixels are classified via a simple and fast nearest-neighbor approach. We obtain results comparable to the state of the art in the semi-supervised scenario, but significantly faster. Since the computed embedding vectors do not depend on the user input, the method is especially well suited for interactive segmentation: the response to the input feedback can be provided almost instantly. In this setup, we reach the same quality than in the semi-supervised case with only 0.15 clicks per frame. The method also naturally generalizes to the multiple objects scenario.

Acknowledgements This project is supported by armasuisse.

References

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

[2] S. Caelles, Y. Chen, J. Pont-Tuset, and L. Van Gool. Semantically-guided video object segmentation. *arXiv:1704.01926*, 2017.

[3] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017.

[4] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

[6] Y. Chen, W. Li, and L. Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *CVPR*, 2018.

[7] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, 2016.

[8] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, 2017.

[9] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[10] 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.

[11] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv:1703.10277*, 2017.

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

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

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

[15] S. D. Jain and K. Grauman. Click Carving: Segmenting Objects in Video with Point Clicks. In *HCOMP*, 2016.

[16] S. D. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv:1701.05384*, 2017.

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

[18] W.-D. Jang and C.-S. Kim. Online video object segmentation via convolutional trident network. In *CVPR*, 2017.

[19] Y. J. Koh and C.-S. Kim. Primary object segmentation in videos based on region augmentation and reduction. In *CVPR*, 2017.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[21] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[22] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *arXiv:1709.06031*, 2017.

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

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

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

[26] 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.

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

[28] J. Pont-Tuset, M. Farré, and A. Smolic. Semi-automatic video object segmentation by advanced manipulation of segmentation hierarchies. In *International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2015.

[29] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[30] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009.

[31] D. Qin, Y. Chen, M. Guillaumin, and L. Van Gool. Learning to rank bag-of-word histograms for large-scale object retrieval. In *BMVC*, 2014.

[32] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, 2014.

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[34] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots. One-shot learning for semantic segmentation. *arXiv:1709.03410*, 2017.

[35] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *T-PAMI*, 22(12):1349–1380, 2000.

[36] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016.

[37] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. *arXiv:1704.05737*, 2017.

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

[39] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017.

[40] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics*, 24(3):585, 2005.

[41] T. Wang, B. Han, and J. Collomosse. TouchCut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding*, 120:14–30, 2014.

[42] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *CVPR*, 2015.

[43] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[44] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *CVPR*, 2016.

[45] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang. Deep interactive object selection. In *CVPR*, 2016.

[46] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. *arXiv:1802.01218*, 2018.