Meta-Learning Deep Visual Words for Fast Video Object Segmentation

Harkirat Singh Behl Mohammad Najafi Anurag Arnab Philip H.S. Torr
University of Oxford
{harkirat, monaj, aarnab, phst}@robots.ox.ac.uk

Figure 1: **Video object segmentation using a dictionary of deep visual words.** Our proposed method represents an object as a set of cluster centroids in a learned embedding space, or "visual words", which correspond to object parts in image space (bottom row). This representation allows more robust and efficient matching as shown by our results (top row). The visual words are learned in an unsupervised manner, using meta-learning to ensure the training and inference procedures are identical. The t-SNE plot [29] on the right shows how different object parts cluster in different regions of the embedding space, and thus how our representation captures the multi-modal distribution of pixels constituting an object.

Abstract

Accurate video object segmentation methods finetune a model using the first annotated frame, and/or use additional inputs such as optical flow and complex post-processing. In contrast, we develop a fast algorithm that requires no finetuning, auxiliary inputs or post-processing, and segments a variable number of objects in a single forward-pass. We represent an object with clusters, or "visual words", in the embedding space, which correspond to object parts in the image space. This allows us to robustly match to the reference objects throughout the video, because although the global appearance of an object changes as it undergoes occlusions and deformations, the appearance of more local parts may stay consistent. We learn these visual words in an unsupervised manner, using meta-learning to ensure that our training objective matches our inference procedure. We achieve comparable accuracy to finetuning based methods, and state-of-the-art in terms of speed/accuracy trade-offs on four video segmentation datasets.

1. Introduction

Masses of video data is produced every second – examples include uploads to the internet and streams recorded by driverless cars and surveillance systems. Fast and accurate algorithms to understand video data are thus of paramount importance. This paper addresses the task of video segmentation, and follows the protocol defined in the DAVIS datasets [4, 36]. Here the ground-truth mask of one or more objects are provided in the first frame, which must then be tracked at a pixel-level throughout the rest of the video.

Accurate approaches to video segmentation trained a fully convolutional network (FCN) [27] for foreground/background segmentation on existing datasets, and then adapted it to the testing video by finetuning the network on the first, fully-annotated frame [3, 28, 44, 21, 40, 9]. Although these methods produce accurate results (and can be improved further by using optic flow [42, 2, 8, 16, 47] or post-processing with DenseCRF [22, 3, 2, 7]), they are extremely time consuming, taking between 700s to 3h to finetune per DAVIS video [3, 21], rendering them unsuitable for real-life applications.

This paper, in contrast, considers the more challenging scenario where the network is not finetuned at all, and uses no optical flow or additional post-processing, in order to develop a fast algorithm. Our approach is inspired by metric-learning approaches which embed pixels from the same object close to each other in a learned embedding space, and pixels from different objects far apart. Chen *et al*. [6] used this idea to formulate video segmentation as a pixelwise retrieval task, where each pixel of the ground-truth mask was embedded in the first frame to form an index, and pixels in subsequent frames were classified with nearest neighbours.

Contrastingly, in the related context of few-shot learning, Prototypical networks [41] represent each class with the mean of their embeddings and classify subsequent queries with a softmax over distances to each prototype.

Prototypical networks, although simple and fast, do not have sufficient capacity to model complex, multi-modal data distributions such as an object in a video that undergoes deformations, occlusions and viewpoint changes. Nearest neighbour approaches [6, 24, 33, 17] have greater modelling capacity, but are more computationally expensive as the time and memory cost to perform a lookup grows linearly with the size of the index. For pixel-level tasks, they also store many redundant pixels with similar appearance in the index. Furthermore, they are more prone to overfitting and noise, which becomes more prevalent during the "online adaptation" of video segmentation models to account for variations throughout the video [6, 17, 44, 9].

Our flexible approach interpolates the spectrum of metric learning approaches by representing an object with a fixed number of cluster centroids in the embedding space. We denote this as a dictionary of visual words, because each cluster centroid in the embedding space corresponds to a part of the object in the image space as shown in Fig. 1, even though these words are formed in an unsupervised manner.

The use of visual words enables more robust matching, because even though an object as a whole may be subject to occlusions, deformations, viewpoint changes, or disappear and reappear from the same video, the appearance of some of its more local parts may stay consistent.

These visual words are learned without any explicit supervision by clustering our embedding space, and using meta-learning to ensure that our training objective matches our inference procedure. This is in contrast to related metric-learning based approaches [6, 33, 24] which are trained with surrogate, and sometimes unstable, losses. Furthermore, as our method requires only a single forward-pass to segment a variable number of objects per video, it naturally scales to the multi-object setting. Related methods [45], in contrast, segment each object independently before combining results and are thus slower for multiple objects.

The advantages of our simple and intuitive approach is reflected by its performance on multiple single- and multi-object video segmentation datasets (DAVIS 2016 [35], DAVIS 2017 [36], SegTrack v2 [23], YouTube-Objects [37, 18]) where we achieve comparable accuracy to finetuning-based methods (whilst being 1 to 2 orders of magnitude faster), and lie on the Pareto front as no other published methods are both faster and more accurate.

2. Related Work

The most accurate video segmentation methods using the DAVIS protocol [35, 36] currently finetune models on the first frame of the video [28, 3, 44, 21] and/or use optical flow [16, 47, 8, 42] or DenseCRF [3, 2, 7] post-processing to improve performance. Our proposed approach does not involve finetuning, or additional information such as optical flow, and still achieves comparable performance whilst being one to two orders of magnitude faster.

Fast approaches to video segmentation, that do not finetune on the first frame, or use optical flow, can broadly be divided into methods performing mask propagation or metric learning. Mask propagation methods, such as [34, 45, 19], use the segmentation mask from the one frame to guide the network to predict the mask in the next frame (*i.e*. pixel-level tracking). These methods use the prior that objects move smoothly and slowly over time, and thus struggle when there are temporal discontinuities like occlusion or rapid motion. Moreover, errors accumulate over time as the model "drifts", particularly if the algorithm loses track of the object. Li *et al*. [25] addressed this issue using re-identification modules which traverse the video back-and-forth to recover any potential missed object prediction. However, this method is not causal as it looks at future frames. Oh *et al*. [45] do not only use the previous frame, but also the first reference frame, to guide the tracking. However, this does not completely alleviate the problem of model drift, as if the model loses track of the object, the object's appearance may have changed so much from the first frame that the reference frame is not effective in recovering it. Moreover, since these methods match the entire object as a whole, they struggle with occlusions. This is in contrast to our approach which represents objects by their constituent parts to be more robust to appearance changes. Finally, [45] is designed for tracking a single object, and thus handling multiple objects require processing each object individually before heurstically merging results. Our method in comparison segments multiple objects in a single-forward pass.

Our work is more similar to methods using pixel-to-pixel matching or metric learning [6, 17, 33, 40, 24]. Chen *et al*. [6] formulated video segmentation as a pixel-level retrieval problem, where embeddings from the first reference frame are used to form an index for a nearest neighbour classifier. Note that the method of [6] is trained with a variant of the triplet loss, which though common for metric learning, does not optimise explicitly for the nearest neighbour search at inference time. The triplet loss is also difficult to train with, as it is very sensitive to triplet selection [39, 14]. Siamese networks have also been employed in a similar manner [17, 33, 40], where one branch computes embeddings from the annotated first frame which are used to match to the embeddings computed by the other branch on the current frame. When classifying query images, these methods all effectively search all the pixels from the reference frame. This approach is not only expensive in terms of time and memory (as it retains redundant embeddings of similar pixels), but is also more susceptible to noise. This

is an issue during the "online adaptation" [6, 9, 17, 44] of the model which may introduce incorrectly labelled embeddings. Our method retains only cluster centroids in the embedding space (which correspond to exemplars of object parts), which enables faster and more robust matching.

The utility of object parts for more robust matching for video segmentation has been identified before by [7]. However, Cheng *et al*. [7] use handcrafted heuristics to form object parts based on bounding boxes in the image space. In contrast, we cluster our embedding space in an unsupervised manner to obtain visual words which resemble object parts (as pixels with similar appearance cluster together). Moreover, the method of Cheng *et al*. [7] – which tracks bounding boxes of object parts and then merges foreground segmentations within these boxes – consists of two separately trained modules (using different datasets), whereas our method is trained via meta-learning with a single objective function that matches our inference procedure.

Finally, we note that meta-learning has not been explored much in the context of video segmentation. Yang *et al*. [48] used meta-learning to adapt the weights of the final layer of a segmentation network at test time. This is in contrast to our method which adaptively computes the initial dictionary of visual words from the first labelled frame in the video.

Note that our method can be viewed as a generalisation of Protoypical networks [41] and Matching networks [43] for few-shot classification. Prototypical networks represents the training data from each class as a single prototypical vector. Matching networks, on the opposite end of the spectrum, consider all training data samples of a particular class to make a classification regardless of how redundant or noisy these samples may be. Our method interpolates these two methods by representing an object class via a fixed number of cluster centroids, which correspond to exemplars of object parts in the case of video segmentation.

3. Proposed Approach

We first describe the formulation of video object segmentation as a meta-learning problem in Sec. 3.1. This allows us to train our model in the same way that it will be tested, unlike other metric-learning based approaches to video segmentation [6, 17]. We then describe our model and meta-training strategy in Sec. 3.2 and 3.3. Finally Sec. 3.4 describes how we can easily adapt our model to account for changes in the object's appearance through the video in.

3.1. Video Object Segmentation as Meta-Learning

Meta-learning, or learning to learn, is often defined as learning from a number of tasks in the training set, to become better at learning a new task in the test set [38, 32, 15, 11]. In the context of video object segmentation, the task is to learn from the ground-truth masks of the objects in the first frame of the video (support set) to segment and track them in rest of the video (query set). Our meta-learning objective is to learn model parameters θ on a variety of tasks (videos), which are sampled from the distribution $p(\mathcal{T})$ of training tasks (*i.e*. meta-training set), such that the learned model performs well on a new unseen task (test video). Denoting the loss of the model on the n^{th} task, $\mathcal{T}_n$, as $\mathcal{L}_{\mathcal{T}_n}(\theta)$, the meta-training objective is thus

$$\theta^* = \arg\min_{\theta} \sum_{\mathcal{T}_n \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_n}(\theta). \quad (1)$$

Figure 2: **Formulation of video object segmentation as a meta learning problem.** Each video presents a new task: Learn from the ground truth object masks on the first frame (support set), to segment them on the remaining frames in the video (query set).

Figure 2 illustrates our meta-learning setup for video segmentation using this definition. The support set $\mathcal{S}$ is the set of all labeled pixels in the first frame, $\mathcal{S} = \{x_i, y_i\}_{i=1}^{N}$. Here x_i represents the pixel i in the first frame, $y_i \in \mathcal{C} = \{1, ..., C\}$ is the ground truth class label of pixel x_i, N is the number of labeled pixels in the frame, and C is the number of object classes that need to be tracked and segmented in the video. Similarly the query set is defined by $\mathcal{Q} = \{x_j, y_j\}_{j=1}^{N \times F}$, where F is the number of frames in the video after the first frame, and j indexes pixels across all F frames. The output of each task $\mathcal{T}$ is the set of predicted class labels for the pixels in $\mathcal{Q}$, $\hat{\mathcal{Y}} = \{\hat{y}_j\}_{j=1}^{N \times F}$.

Next, we describe our model for estimating the outputs of each task, *i.e*. the object label for every pixel in the query frames of the video.

3.2. Model

In order to predict the label for each pixel in the query set $\mathcal{Q}$, we need to learn a representation for each object using information from the support set $\mathcal{S}$. We represent each object in the video using a dictionary of deep visual words. Each pixel in the query set is then labelled based on the deep visual word that it is assigned to.

Learning visual words is a challenging task, as we do not have any ground truth information of the object parts

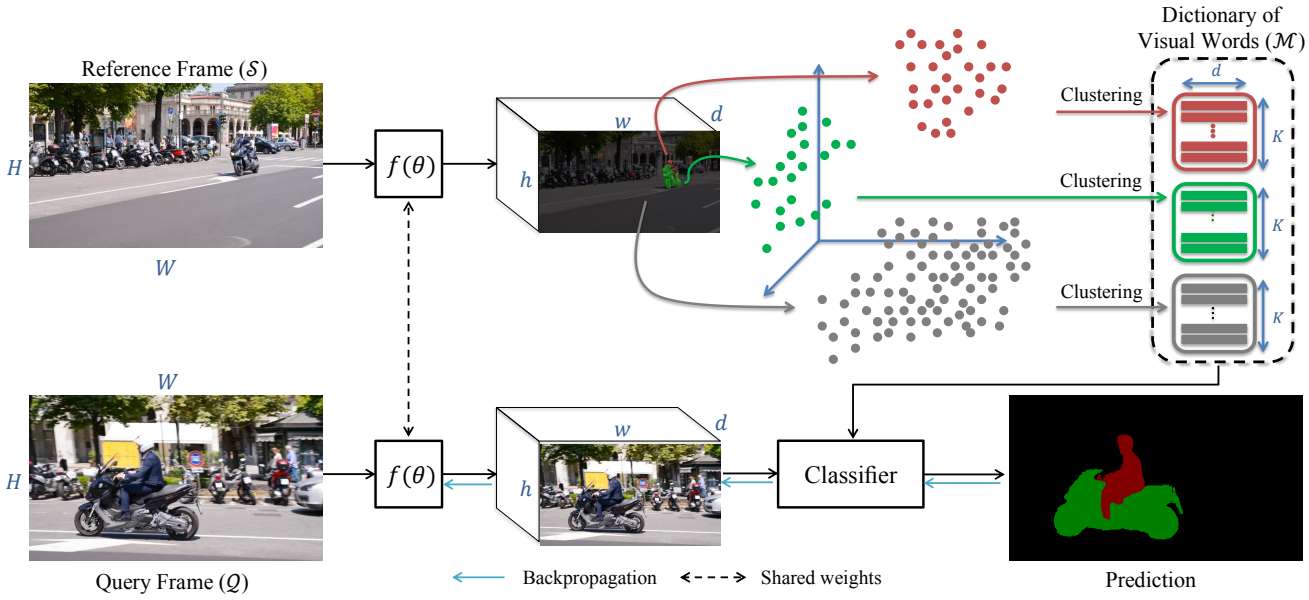

Figure 3: **Overview of the proposed method.** The first frame of the video (reference frame), which forms the support set $\mathcal{S}$ in our meta-learning setup, passes through a deep segmentation network $f(\theta)$ to compute a $d = 128$ dimensional embedding for each pixel. A dictionary of deep visual words are then learned by clustering these embeddings for each object in the reference frame (Eq. 2). Pixels of the query frame are classified as one of the objects based to their similarities to the visual words (Eq. 3 and Eq. 4). The model is meta-trained by alternately learning the visual words given model parameters θ, and learning model parameters given the visual words.

that they correspond to. Consequently, as summarised in Fig. 3, we use a meta-training algorithm, where we alternate between the unsupervised learning of deep visual words (Sec. 3.2.1) and supervised learning of pixel classification given these visual words (Sec. 3.2.2). Our model thus learns to learn a better classifier, by optimising the visual words that it will produce at test-time.

3.2.1 Unsupervised Learning of Deep Visual Words

We initially pass the first frame of the video, which is the support set $\mathcal{S}$, through a deep neural network $f(\theta)$ to compute the embedding for each pixel x_i in $\mathcal{S}$, $f_\theta(x_i)$. We then compute a set of deep visual words for all the pixels in each object class. Let $\mathcal{S}_c$ be the set of pixels in $\mathcal{S}$ with class label c. Each set $\mathcal{S}_c$ is partitioned into K clusters $\mathcal{S}_{c1}, ..., \mathcal{S}_{cK}$ using the k-means algorithm [1], with μ_{ck} being the respective centroids of the clusters, using the objective:

$$\mathcal{S}_{c1}, ..., \mathcal{S}_{cK} = \underset{\mathcal{S}_{c1},....,\mathcal{S}_{cK}}{\arg\min} \sum_{k=1}^{K} \sum_{x_i \in \mathcal{S}_{ck}} \|f_\theta(x_i) - \mu_{ck}\|_2^2, \tag{2a}$$

$$\mu_{ck} = \frac{1}{|\mathcal{S}_{ck}|} \sum_{x_i \in \mathcal{S}_{ck}} f_\theta(x_i). \tag{2b}$$

In other words, we represent the distribution of the pixels within each set $\mathcal{S}_c$ in the learned embedding space with a set of deep visual words $\mathcal{M}_c = \{\mu_{c1}, ..., \mu_{cK}\}$. We can, in principle, use any clustering algorithm here and choose k-means as it is computationally efficient and simple.

3.2.2 Supervised Learning for Pixel Classification

Once the deep visual words for each object have been constructed, the probability of assigning a pixel $x_j \in \mathcal{Q}$ to the k^{th} visual word from the c^{th} object class is computed using a non-parametric softmax classifier,

$$p(c_k|x_j) = \frac{\exp\left(s(\mu_{ck}, f_\theta(x_j))\right)}{\sum_{\mu_i \in \mathcal{M}} \exp\left(s(\mu_i, f_\theta(x_j))\right)}, \tag{3}$$

where $\mathcal{M} = \bigcup_{c=1}^{C} \mathcal{M}_c$ is the dictionary of deep visual words for all objects present in the video, and s is the cosine similarity function. We enable our model to account for intra-class variations by encouraging each pixel to resemble only one relevant visual word. As a result, the probability of pixel x_j taking the object class label c is defined as

$$p(\hat{y_j} = c|x_j) = \frac{\max_{k \in \{1,..,K\}} p(c_k|x_j)}{\sum_{c'=1}^{C} \max_{k \in \{1,..,K\}} p(c'_k|x_j)}. \tag{4}$$

This allows our model to learn meaningful visual words that correspond to the diverse object parts that constitute an object. Note from the T-SNE visualisation of our embeddings in Fig. 1 that pixels from different parts of the same object cluster in separate regions of the embedding space.

Finally, our loss function for this pixel classification problem, where $\hat{y}_j$ is the predicted object class label for

pixel x_j is defined as

$$\mathcal{L}_{\mathcal{T}_n} = -\frac{1}{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{Q}|}\log\left[p(\hat{y}_j = y_j|x_j)\right] \quad (5)$$
$$-\frac{1}{|\mathcal{Q}|(C-1)}\sum_{j=1}^{|\mathcal{Q}|}\sum_{c=1,c\neq y_j}^{C}\log\left[1-p(\hat{y}_j = c|x_j)\right].$$

The first term is the standard cross-entropy term that is typically used in image classification and semantic segmentation. The second term further reduces the probability of incorrect classes, by pulling their embeddings further away from the visual word of the correct class. We included this second term as it experimentally improved results.

3.3. Meta-training procedure

Each iteration of our meta-training algorithm consists of an unsupervised learning process to construct a dictionary of visual words from the support set $\mathcal{S}$, followed by a supervised learning step where the segmentation network parameters, θ, are updated by minimsing the loss function in Eq. 5 according to Eq. 1. In other words, the model learns to learn deep visual words in the first frame of the video to minimise a pixel-level loss over the rest of the video.

Our proposed meta-learning approach is quite flexible, and generalises previous work such as [41] and [43]. Prototypical networks [41] represent each class with a single prototypical vector (*i.e.* one visual word), whilst Matching networks [43] represent each class using all the samples of that class in the support set (*i.e.* the embedding of each pixel in $\mathcal{S}$ would form a visual word). Our method interpolates between these two approaches to build a more robust representation of the support set $\mathcal{S}$. Also note that previous metric-learning approaches to video object segmentation such as [6] learn an embedding using variants of the triplet loss and perform nearest neighbour classification at test time. This approach is thus similar to Matching networks [43], with the key difference being that the training objective (triplet loss) does not correspond to the inference procedure (nearest neighbour search).

3.4. Online Adaptation

The objects of interest from the first frame, as well as the background, often undergo deformations, occlusions, viewpoint changes and other transformations. As a result, adapting the model throughout the video is vital to achieve good performance and done by state-of-art methods [44, 6, 17].

We adapt our model by simply updating the set of visual words that represent the object. Concretely, given a dictionary of visual words $\mathcal{M}$, captured up to the frame t_j, we predict the segmentation map in frame $t_{j+\delta}$, and treat it as a new support set $\mathcal{S}^\delta = \{x_i^\delta, y_i^\delta\}_{i=1}^N$, where y_i^δ is the predicted object class for pixel x_i^δ. Next, we compute an updated set of deep visual words $\mathcal{M}^\delta$ from the new support set using k-means as described in Sec. 3.2.1, and compute their corresponding cluster centroid representations by

$$\mu_{ck}^\delta = \frac{1}{|\mathcal{S}_{ck}^\delta|}\sum_{x_i\in\mathcal{S}_{ck}^\delta} f_\theta(x_i). \quad (6)$$

To filter out incorrect predictions and prevent errors from compounding, we only add new words that still resemble the existing ones. This is based on the assumption that within a time interval δ, where δ is chosen moderately, the objects will deform slowly and their pixel-level embeddings will also not vary greatly. Concretely, we update the main visual word set $\mathcal{M}$ with the new set $\mathcal{M}^\delta$, if there are $m^\delta \in \mathcal{M}_c^\delta$ and $m \in \mathcal{M}_c$, for which $\left\|\mu_m^\delta - \mu_m\right\| \leq \alpha$.

Additionally, to ensure that online adaptation uses reliable and confident pixel-level predictions to update the visual words, we apply a simple outlier removal process (that assumes spatio-temporal consistency of objects over time) to the pixel-level predictions. Specifically, we discard regions from the adaptation process if they have no intersection with the predicted object mask in the previous frame.

Note that during online adaptation, none of the existing words within $\mathcal{M}$ are discarded, because each object may revert to its original shape, appearance or viewpoint during a video. This is also why the Eq. 4 takes the maximum value to match to only the most relevant visual word.

The effect of this online adaptation procedure, and other design choices, are experimentally validated next.

4. Experimental evaluation

We first describe our experimental settings in Sec. 4.1 before comparing to state-of-art methods on four video segmentation datasets in Sec. 4.2. Finally, Sec. 4.3 presents an ablation study. For reproducibility, we will release training code and models used in our experiments.

4.1. Experimental setup

Model We use a Deeplab v2 architecture as the encoder [5], $f(\theta)$, which uses a ResNet-101 [13] backbone with dilated convolutions. The encoder maps an input frame of size $H \times W$ to a feature of size $H \times W \times 2048$. We add an additional convolutional layer to produce an embedding of $d = 128$ dimensions, and bilinearly upsample this to the original image size. These 128-dimensional embeddings are then clustered to form our visual words.

Unless otherwise specified, we use $k = 50$ visual words for the foreground object classes. As the background typically contains more variation, we use four times as many-clusters for the background. For online adaptation, we set $\alpha = 0.5$. The ablation study in Sec. 4.3 shows the effect of the number of visual words, k, whilst the supplementary shows that our method is not very sensitive to α.

Datasets We evaluate on standard video segmentation datasets for tracking both single objects (DAVIS-2016 [35], YouTube-Objects [37, 18]) and multiple objects (DAVIS-2017 [36], SegTrack v2 [23]) given fully-annotated object masks in the first frame. DAVIS-2016 contains 30 training and 20 validation videos. DAVIS-2017 extends DAVIS-2016 to 60 training and 30 validation videos. Furthermore, multiple objects (ranging from 1 to 5, with an average of 2) are annotated in the first frame and must be tracked through the video, making it considerably more challenging than DAVIS-2016. YouTube-Objects and SegTrack v2 do not have a training split, so we evaluate our model trained on DAVIS-2017 on them.

Training Following competing methods which use a model pretrained on image segmentation datasets [17, 45, 6, 48, 31, 44], we initialise the encoder of our network using the public Deeplab-v2 model [5] that has been trained on COCO [26]. Thereafter, we meta-train our model following the "episodic training" procedure, which is the standard practice [43, 41, 11]. Each training episode is formed by sampling a support set $\mathcal{S}$ and a relevant query set $\mathcal{Q}$. The idea of episodic training is to, at each iteration, mimic the inference procedure. In other words, the query set should be classified given only the support set. Here, we build each episode by first randomly sampling a video from the training dataset, treating the pixels of the first frame of the video as $\mathcal{S}$, and randomly selecting a set of query frames from the rest of the video and treating their pixels as $\mathcal{Q}$.

Evaluation metrics We report standard metrics defined by the DAVIS protocol [35]: The mean IoU ($\mathcal{J}$), the F-score along the boundaries of the object ($\mathcal{F}$) and the mean of these two values ($\mathcal{J}\&\mathcal{F}$). We also report the "decay" [35] in $\mathcal{J}$. This is calculated by splitting a video temporally into four clips, and taking the difference of the IoU in the last clip to the first clip. Lower scores of "decay" are better, and was proposed by [35] to measure whether a model is robust or if its predictions degrade over time.

Finally, we also report our runtime per-frame. Our runtime is measured on a desktop machine with a single Titan X (Pascal) GPU, and an Intel i7-6850K CPU with six cores.

4.2. Comparison to state-of-art

Table 1 shows our state-of-art results on DAVIS-2017, DAVIS-2016, YouTube-Objects and SegTrack-v2. On all these datasets, there is no method that is both faster and more accurate than us. This Pareto front is also visualised in Fig. 4 for DAVIS-2017, the most challenging dataset. The methods that are more accurate than us all finetune on the first frame, use optical flow or additional post-processing such as CRFs [22] and thus have a runtime that is larger by a factor of at least 8 [25, 3].

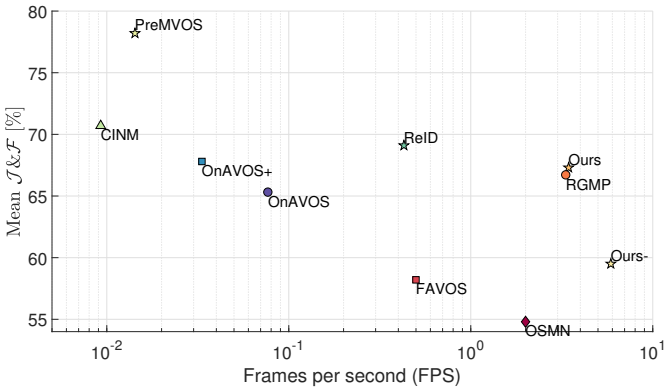

Figure 4: **Comparison of speed and accuracy on DAVIS 2017.** Entries on the Pareto front (*i.e.* no other method is both faster and more accurate) are marked by a star. Note the speed axis uses a logarithmic scale.

The only method that is close to us in terms of speed and accuracy is RGMP [45]. However, the runtime of RGMP increases linearly with the number of objects being tracked from the first frame. This is because RGMP processes each object instance independently through the "encoder" part of their network [45], and combine their results together at the end. Therefore, even though RGMP is faster than our method on DAVIS-2016 (a single-object dataset), it is actually slower on DAVIS-2017 (a multi-object dataset). As the authors did not report the runtime of their method on DAVIS-2017, we ran their publicly available inference code, and obtain an average runtime of 0.30s per frame on DAVIS-2017 (Tab. 1). However, DAVIS-2017 only averages 2 objects per video. We measured RGMP to average 0.11s, 0.41s and 0.60s per frame for videos with 1, 3 and 5 objects respectively. The runtime of our method, in contrast, increases much slower, taking 0.25s, 0.38s and 0.53s respectively. Thus, the runtime of RGMP increases by $5.4\times$ from 1 object to 5 objects, whilst our runtime only increases by $2.1\times$. This is because our model only requires a single forward-pass through the network, irrespective of the number of objects being tracked. Thus, there is only a minor increase in runtime from DAVIS-2016 to DAVIS-2017 as there are more visual words. Note that the runtime advantage of our method would increase over RGMP [45] if even more objects were to be tracked in a video. Moreover, our speed could also be improved by implementing CUDA kernels for k-means clustering.

Note that our method also achieves a lower $\mathcal{J}$ Decay [35] than RGMP [45] indicating greater robustness. This is also shown qualitatively in Fig. 6 where our method overcomes occlusions and can recover from errors made in previous frames, unlike mask propagation methods like RGMP. We believe that representing objects with cluster centroids in the embedding space (visual words), which correspond to object parts in the image space, increases the robustness of our matching, as the appearance of more local parts typ-

Table 1: **State-of-art results among methods not performing finetuning on four common video object segmentation datasets.** Legend: FT: Fine-Tuning on the first frame of the test video; PP: Post-Processing; OF: Optical Flow; Ours$^-$: Our model without online adaptation; †: An ensemble of models are used. *As the original authors did not report the runtime, we timed it using the public inference code. Evaluation metrics are detailed in Sec. 4.1.

				DAVIS-2017					DAVIS-2016					YouTube-Objects	SegTrack-v2
Method	FT	PP	OF	$\mathcal{J}$(%)	$\mathcal{J}$ Decay(%)	$\mathcal{F}$(%)	$\mathcal{J}\&\mathcal{F}$(%)	Time(s)	$\mathcal{J}$(%)	$\mathcal{J}$ Decay(%)	$\mathcal{F}$(%)	$\mathcal{J}\&\mathcal{F}$(%)	Time(s)	$\mathcal{J}$(%)	$\mathcal{J}$(%)
MSK [34]	✓	✓	✓	–	–	–	–	–	79.7	–	75.4	77.5	12	72.6	70.3
MaskRNN [16]	✓		✓	60.5	–	–	–	9	80.7	–	80.9	80.8	0.60	–	72.1
OnAVOS [44]	✓	✓		61.6	27.9	69.1	65.3	13	86.1	5.2	84.9	85.5	13	77.4	–
DRL [12]	✓			–	–	–	–	–	84.1	–	84.6	84.3	–	78.1	–
Lucid [21]	✓	✓	✓	–	–	–	–	–	84.8	–	–	–	∼ 190	76.2	77.6
OSVOS^S [3]	✓	✓		64.7	15.1	71.3	68.0	–	**85.6**	5.5	87.5	86.5	4.50	**83.2**	65.4
CINM† [2]	✓	✓	✓	67.2	24.6	74.4	70.7	∼ 108	83.4	12.3	85.0	84.2	–	78.4	–
ReID [25]	✓		✓	67.3	–	71.0	69.1	2.33	–	–	–	–	–	79.6	**78.7**
PReMVOS† [28]	✓		✓	**74.3**	16.2	**82.2**	**78.2**	∼ 70	85.5	8.8	**88.6**	**87.0**	∼ 70	–	–
MaskRNN [16]			✓	45.5	–	–	–	0.60	–	–	–	–	–	–	–
FAVOS [7]		✓		54.6	**14.1**	61.8	58.2	>1.80	**82.4**	**4.5**	79.5	80.9	1.80	–	–
CTN [20]			✓	–	–	–	–	–	73.5	15.6	69.3	71.4	1.33	–	–
FAVOS [7]				–	–	–	–	–	77.9	–	76.0	76.9	0.60	–	–
VPN [19]				–	–	–	–	–	70.2	12.4	65.5	67.8	0.63	–	–
BVS [30]				–	–	–	–	–	60.0	28.9	58.8	59.4	0.37	68.0	60.0
PML [6]				–	–	–	–	–	75.5	8.5	79.3	77.4	0.27	–	–
OSMN [48]				52.5	21.5	57.5	54.8	0.50*	74.0	9.0	–	–	0.14	69.0	–
VideoMatch [17]				56.5	–	–	–	0.35	81.0	–	–	–	0.32	79.7	–
RGMP [45]				**64.8**	18.9	68.6	66.7	0.30*	81.5	10.9	82.0	81.7	**0.13**	–	71.1
Ours$^-$				55.8	24.6	63.1	59.5	0.17	76.2	11.2	77.6	76.9	0.17	77.4	64.6
Ours				63.9	14.4	**70.7**	**67.3**	**0.29**	81.5	5.0	**82.7**	**82.1**	0.25	**81.1**	**72.0**

Table 2: **The effect of different object representations** The same MS-COCO pretrained network is used, without any online adaptation. We use the 5 nearest neighbours, following [6].

Model	$\mathcal{J}$(%)	Time(s)
Single prototype	32.9	**0.14**
5 Nearest neighbours	45.9	5.50
Deep visual words ($k = 50$)	**48.4**	0.17

Table 3: **The effect of the size of visual word dictionary on model performance** Results are on DAVIS-2017, without any online adaptation.

Dictionary Size (k)	1	5	10	20	50	100	200	400
$\mathcal{J}$(%)	49.9	54.5	54.8	54.9	55.8	56.3	56.3	56.4
Time (s)	0.140	0.168	0.170	0.171	0.173	0.199	0.254	0.373

ically stays consistent even though the object as a whole transforms. And as our online adapatation process retains a memory of previous visual words, our method can handle objects disappearing and reappearing (Fig. 6) unlike RGMP.

4.3. Ablation study

This section studies how different design choices in our algorithm impact overall performance on DAVIS-2017.

Effect of Meta-Learning To evaluate the efficacy of meta-learning, we evaluated our MS-COCO initialised network and obtained a mean IoU of ($\mathcal{J}$) of 50.7. Meta-training significantly improves our IoU to 63.9% (Tab. 1). Perhaps surprisingly, our initialisation already outperforms published work like Mask-RNN [16] (Tab. 1). The performance of our initialisation leads us to study the effect of our object representation next.

Object representation We represent the object given in the first frame of the video with a dictionary of k visual words in the embedding space. An alternative is to represent each object with a single vector, *i.e.* $k = 1$ (as in Prototypical networks [41]). In our case, this prototype is formed by taking the mean embedding of all pixels of the object labelled in the first frame. The other end of the spectrum is to represent each object with separate embeddings for all of its pixels, *i.e.* $k = n$ where n is the number of labelled pixels in the first frame. This corresponds to the approach of Chen *et al*. [6] and also Matching networks [43].

Table 2 compares these approaches for our MS-COCO pretrained network. It shows that using $k = 50$ clusters outperforms both nearest neighbour classification and a single prototypical vector per class. This motivates our reason for using k visual words to represent an object and suggests why we outperform methods such as [6] in Tab. 1.

Note how matching using our visual words representation has a similar runtime to a single prototype and is significantly faster than performing a nearest neighbour search. This is because the search time is linear in the number of pixels, $O(n)$. And like the other approaches we compare to in Tab. 2, we do the matching at full resolution. The runtime could be greatly reduced by doing the look-up on a subsampled image (for example, [6] do the look-up at $1/8$ resolution which reduce the time by about a factor of 64).

Effect of visual word dictionary size Table 3 examines the effect that the number of visual words in the dictionary has on accuracy and runtime on DAVIS-2017. We can see that accuracy steadily increases as the number of clusters is increased from $k = 1$ (which corresponds to Prototypical networks [41]) and plateaus at $k = 50$. We believe that complex objects with high intra-object variations produce embeddings with multi-modal distributions,

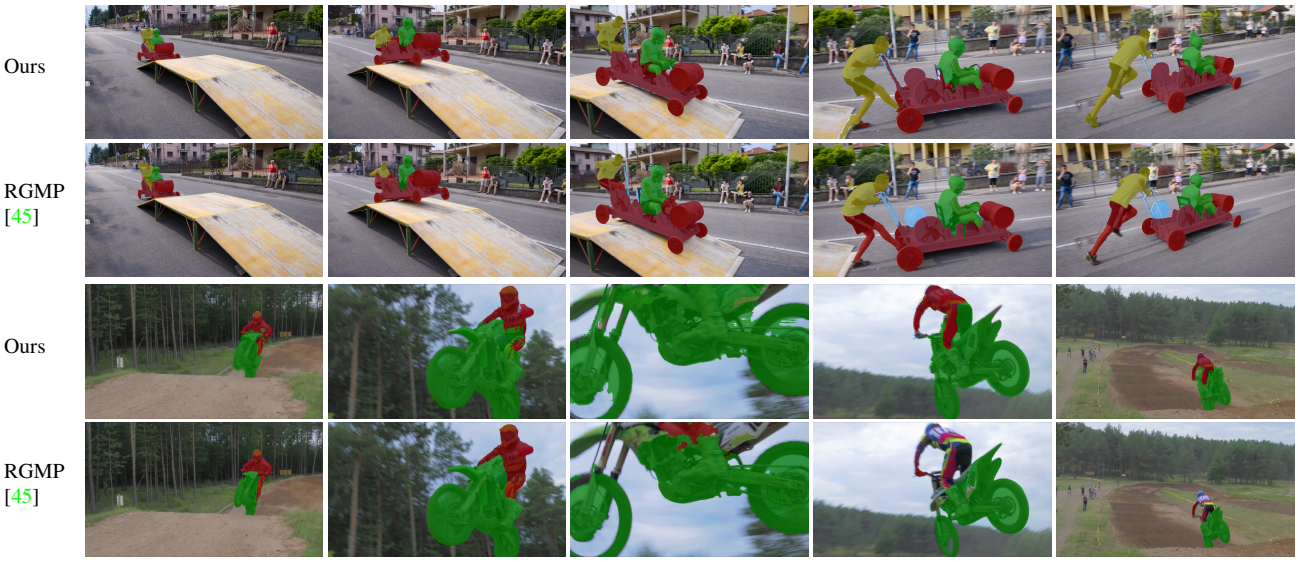

Figure 5: **Qualitative comparison of our method to RGMP [45].** RGMP obtains good results initially in the video (first two columns), but cannot recover after making errors (third column). Note how it misclassifies the yellow person (first example) and loses track of the rider (second example). In contrast, our method overcomes occlusions in all of these cases by robustly matching an object to its constituent parts. Additional results are in the supplementary.

Table 4: **The effect of online adaptation on model performance.** δ denotes the frame interval before every step of online adaptation. Small values of δ help the model to adapt to quickly changing scenes, but too small a δ can introduce noise into the visual words. NA: No online adaptation.

Interval (δ)	NA	30	20	10	5	2	1
$\mathcal{J}(\%)$	55.8	58.6	59.2	61.3	**63.9**	63.7	62.8

Table 5: **The effect of outlier removal and online adaptation**

Outlier Removal	Online Adaptation	$\mathcal{J}(\%)$
✗	✗	55.8
✗	✓	60.4
✓	✓	63.9

which is why they are better represented with multiple visual words. Although increasing k beyond 50 does not substantially change the accuracy, it does increase the runtime, which is why we use $k = 50$ when comparing to existing methods in Tab. 1. Setting the number of visual words to n, the number of pixels in the first frame, would amount to the nearest neighbour search done by [6], which our method outperformed in Tab. 1.

Effect of Online Adaptation Online adapation, as described in Sec. 3.4, updates the dictionary of visual words to account for appearance changes in the object throughout the video. Table 4 shows how updating the dictionary, $\mathcal{M}$, improves performance. Smaller values of the update interval, δ, means $\mathcal{M}$ is updated more frequently and thus helps the system smoothly adapt to dynamic scenes and fast-moving objects. However, very small values of δ, such as $\delta = 1$ also increase the chance of adding noisy visual words, which may explain why $\delta = 5$ performs the best. The supplementary shows that this online update is not sensitive to the distance threshold, α. Table 5 also shows that our simple "outlier removal" step improves results as well, by encouraging spatio-temporal consistency from one frame to the next.

Note that our online adaptation system adds little overhead as it simply updates the existing dictionary of visual words. No additional forward or backward passes through the network are required, unlike methods such as [44].

An experiment about the consistency of visual words over time is included in the supplementary.

5. Conclusion and Future Work

We proposed a novel representation of objects by their cluster centroids in the embedding space (visual words) which correspond to object parts. These visual words were learned without supervision, using meta-learning to ensure that the training and inference procedure are identical. Visual words provide enable more robust matching, as the appearance of more local parts may stay consistent whilst the object as a whole deforms or is occluded. Our novel representation, and meta-training procedure enabled our method to achieve state-of-art performance on four common datasets in terms of speed and accuracy trade-offs (with comparable accuracy to expensive finetuning-based methods that take at least 8 times longer). Moreover, our method readily scales to multiple objects in videos, with its runtime only increasing slightly from single-object DAVIS-2016 to multi-object DAVIS-2017. Future work is to learn the number of clusters automatically, and to incorporate more temporal information into our model.

References

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *ACM-SIAM Discrete Algorithms*, pages 1027–1035, 2007. 4

[2] L. Bao, B. Wu, and W. Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. In *CVPR*, June 2018. 1, 2, 7

[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. 1, 2, 6, 7

[4] S. Caelles, A. Montes, K.-K. Maninis, Y. Chen, L. Van Gool, F. Perazzi, and J. Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018. 1

[5] L. 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. *IEEE PAMI*, 40(4):834–848, 2018. 5, 6

[6] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *CVPR*, June 2018. 1, 2, 3, 5, 6, 7, 8

[7] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang. Fast and accurate online video object segmentation via tracking parts. In *CVPR*, June 2018. 1, 2, 3, 7

[8] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, Oct 2017. 1, 2

[9] H. Ci, C. Wang, and Y. Wang. Video object segmentation by learning location-sensitive embeddings. In *ECCV*, September 2018. 1, 2, 3

[10] L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari. Discovering the physical parts of an articulated object class from multiple videos. In *CVPR*, 2016. 13, 14

[11] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. 2017. 3, 6

[12] J. Han, L. Yang, D. Zhang, X. Chang, and X. Liang. Reinforcement cutting-agent learning for video object segmentation. In *CVPR*, June 2018. 7

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[14] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. In *arXiv preprint arXiv:1703.07737*, 2017. 2

[15] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001. 3

[16] Y.-T. Hu, J.-B. Huang, and A. Schwing. Maskrnn: Instance level video object segmentation. In *NIPS*, 2017. 1, 2, 7

[17] Y.-T. Hu, J.-B. Huang, and A. G. Schwing. Videomatch: Matching based video object segmentation. In *ECCV*, September 2018. 2, 3, 5, 6, 7

[18] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV*, 2014. 2, 6, 18

[19] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, July 2017. 2, 7

[20] W.-D. Jang and C.-S. Kim. Online video object segmentation via convolutional trident network. In *CVPR*, July 2017. 7

[21] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. In *CVPR Workshops*, 2017. 1, 2, 7

[22] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NeurIPS*, pages 109–117, 2011. 1, 6

[23] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 2, 6

[24] S. Li, B. Seybold, A. Vorobyov, A. Fathi, Q. Huang, and C.-C. Jay Kuo. Instance embedding transfer to unsupervised video object segmentation. In *CVPR*, June 2018. 2

[25] X. Li and C. Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *ECCV*, September 2018. 2, 6, 7

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV*, pages 740–755, 2014. 6

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[28] J. Luiten, P. Voigtlaender, and B. Leibe. Premvos: Proposal-generation, refinement and merging for the davis challenge on video object segmentation 2018. In *CVPR Workshops*, 2018. 1, 2, 7

[29] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008. 1

[30] N. Maerki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *CVPR*, 2016. 7

[31] 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. *IEEE PAMI*, 2018. 6

[32] D. K. Naik and R. Mammone. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 437–442. IEEE, 1992. 3

[33] M. Najafi, V. Kulharia, T. Ajanthan, and P. H. S. Torr. Similarity learning for dense label transfer. *CVPR Workshops*, 2018. 2

[34] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017. 2, 7

[35] 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, 6, 18

[36] 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. 1, 2, 6, 18

[37] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 2, 6, 18

[38] J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-...*

hook. PhD thesis, Technische Universität München, 1987. 3

[39] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 2

[40] J. Shin Yoon, F. Rameau, J. Kim, S. Lee, S. Shin, and I. So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *ICCV*, Oct 2017. 1, 2

[41] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NIPS*. 2017. 2, 3, 5, 6, 7

[42] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, June 2016. 1, 2

[43] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*. 2016. 3, 5, 6, 7

[44] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 1, 2, 3, 5, 6, 7, 8

[45] S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *CVPR*, June 2018. 2, 6, 7, 8, 11, 12

[46] S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7376–7385, 2018. 10

[47] H. Xiao, J. Feng, G. Lin, Y. Liu, and M. Zhang. Monet: Deep motion exploitation for video object segmentation. In *CVPR*, June 2018. 1, 2

[48] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, June 2018. 3, 6, 7

Appendix

In this supplementary material, we present more qualitative results (Sec. A), additional ablation studies (Sec. B) and also further quantitative results (Sec. C).

A. Qualitative Results

Figures 6 and 7 present further qualitative comparisons of our method with RGMP [46]. We compare to RGMP as it is the closest to us in both accuracy and speed.

The attached video also contains results for the entire clip.

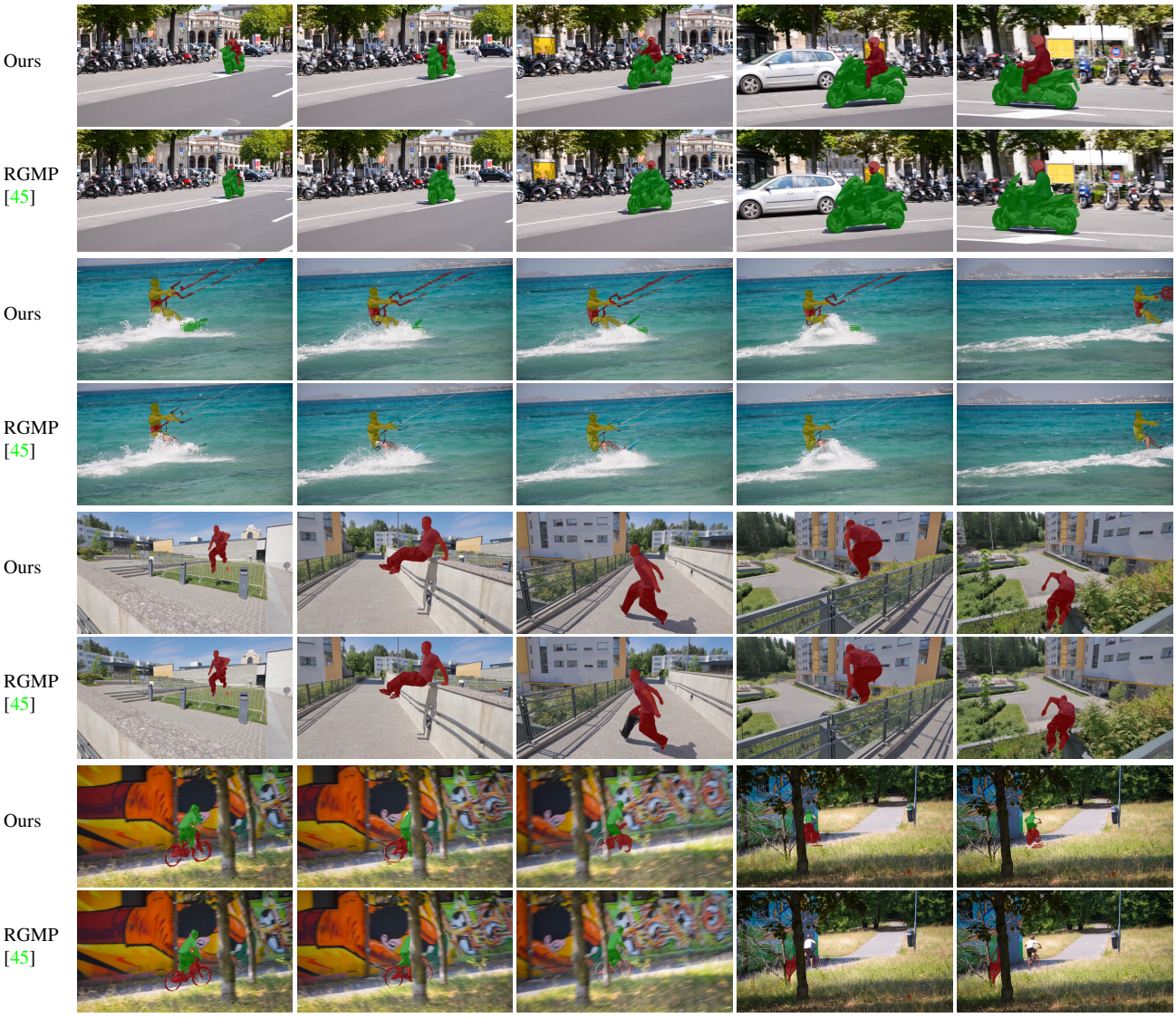

Figure 6: **Success cases of our method, and comparison to RGMP [45].** In each of these videos, our method is able to accurately track the objects labelled in the first frame throughout the video. *First video:* Our algorithm accurately segments the person throughout the video, whilst RGMP cannot deal with the scale and viewpoint changes of the person and mistakes him for the motorbike. *Second video:* Our method is able to segment the kite-surfing harness and wires whilst RGMP loses track of these fine structures. Additionally, note how we are able to segment the heavily-occluded surf-board throughout the video, unlike RGMP. *Third video:* Both methods perform well on this example. *Fourth video:* RGMP loses track of the cyclist from the fourht frame onwards, whereas our method is robust to this occlusion. Mask propagation methods, such as RGMP, struggle with such occlusions. Our representation of objects as visual words is more robust in these situations. Full video results of these clips are included in the supplementary video.

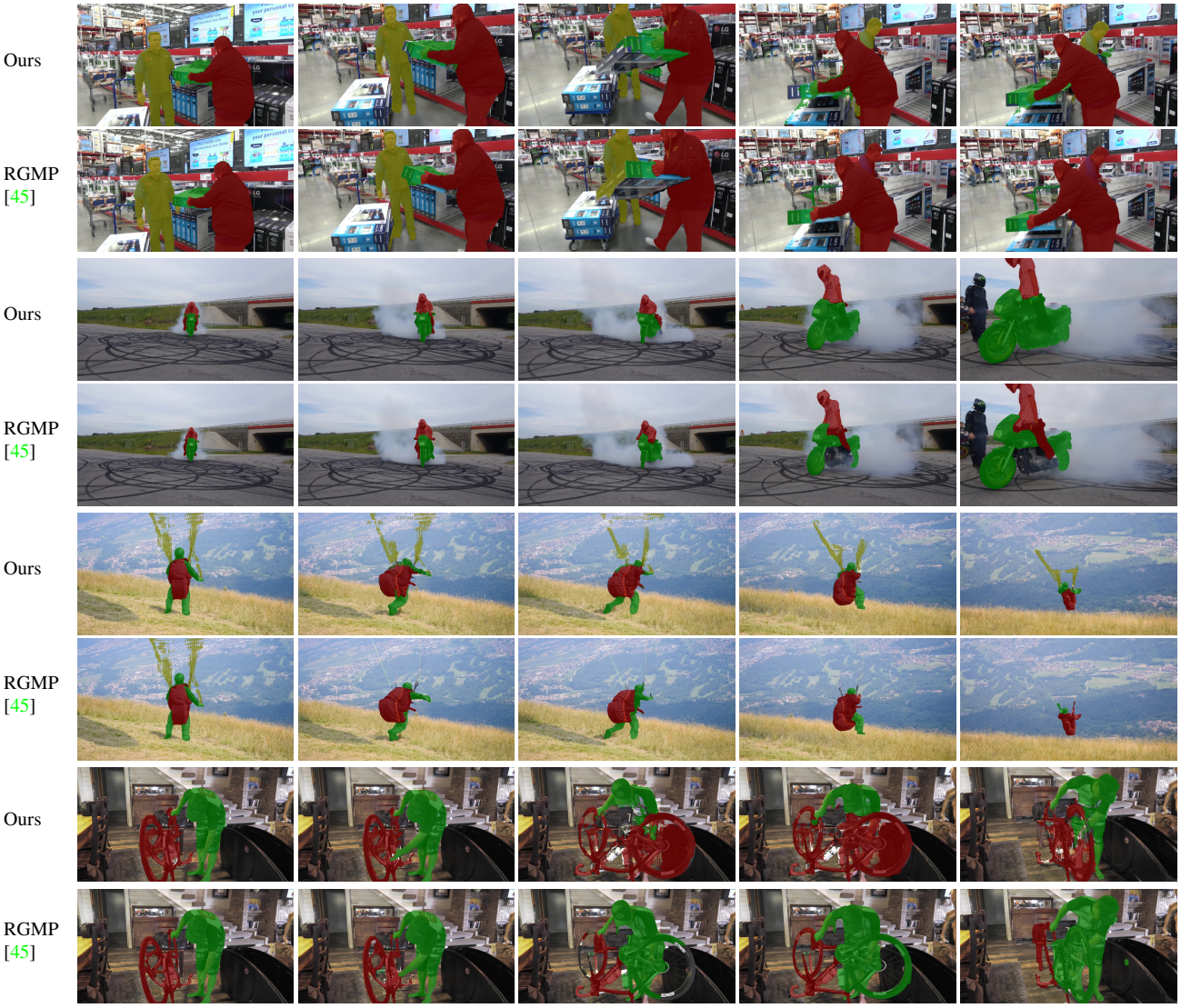

Figure 7: **Failure cases of our method and RGMP [45].** *First video:* Note how our method does not properly segment the green box through the video. RGMP, on the other hand, loses track of the whole box, and also cannot deal with the two people occluding each other in the last two frames. *Second video:* In the fourth and fifth frames, our method confuses the cyclists legs and motorbike. RGMP segments the person properly, but not the entire motorbike. *Third video:* Our algorithm struggles to segment the fine structures of the parachute. RGMP, on the other hand, completely loses track of the parachute after the first frame. *Fourth video:* Our segmentation of the bicycle (particularly its spokes) is not very accurate. RGMP, on the other hand, makes a larger error between the bicycle and person when they occlude each other from the third frame onwards. Full video results of these clips are included in the supplementary video.

B. Additional ablation studies

In this section, we present an additional experiment on the temporal consistency of our visual words, the effect of online adaptation and the size of the visual word dictionary.

B.1. Temporal consistency of our visual words

In this experiment we aim to infer whether our visual words remain in the same region/part on the object throughout the video. We use the Physical Parts Discovery Dataset [10] for this experiment. This dataset contains videos for two object classes, namely, "tiger" and "horse". There are 16 videos for each class, with 8 videos with the animal facing right and the other 8 videos with the animal facing left. Ground truth annotation for 10 body parts of the object is provided for a subset of frames in the video.

We use this dataset, as it is the only one that we are aware of that annotates the parts of an object throughout the video. However, as our method produces object parts in an unsupervised manner, we first form a mapping from each of our visual words to a ground truth object part. Thereafter, we evaluate the consistency of these visual-word-to-object-part mappings over time.

Mapping visual words to ground truth object parts We initially run our algorithm on the first frame of the video, to obtain k visual words for the object. We then map each visual word to a ground truth object part. This assignment is done based on the majority vote from all the pixels in that visual word, *i.e*. a visual word is assigned to the part which it has the maximum spatial overlap with it. This is then the ground truth assignment of the visual words to the object parts and will be used next for evaluation.

Evaluation For all other frames of the video, we run our method and match each pixel to a visual word of the object. We then find the part that these visual words now belong to, which is following the previous paragraph performed according to the maximum spatial overlap. We then calculate how many visual words still belong to the part that they were assigned to in the first annotated frame. The percentage of these consistent mappings, averaged over the entire dataset, is our performance metric, which we denote the "Part consistency score". This metric measures the consistency of our visual words over time, as it indicates if a visual word is still in the same region of the object as it was in the first frame.

Results Table 6 performs this experiment for different numbers of visual words, k. For $k = 10$, the accuracy is 77.4%, showing that our method is indeed consistently tracking parts of the object. This is also shown visually in Fig. 8 and in the supplementary video. Table 6 also shows that the part tracking accuracy decreases as k is increased. We believe this is because each of the regions corresponding to a visual word decreases as k is increased, and thus there is higher variance in visual-word-to-object-part mapping.

Table 6: **Part consistency score as a function of the number of visual words,** k. The relatively high score (the maximum is 100) suggests that our method indeed forms visual words that consistently correspond to the same parts of the tracked object throughout the video. This is performed on the Physical Parts Discovery Dataset [10].

Dictionary Size (k)	10	15	20	30	40	50	60	80	100
Part consistency score (%)	77.4	75.1	74.1	71.9	71.1	71.1	70.9	69.9	69.6

B.2. Effect of α on online adaptation

Figure 9 shows the effect of the distance threshold during online adaptation, α, on the accuracy of our method on DAVIS-2017. It shows that our algorithm is robust to the choice of alpha, performing similarly in a wide range of values when α lies in the interal $[0, 0.7]$, thus motivating our decision to use $\alpha = 0.5$ in our experiments. Although our method is tolerant to a wide range of α values, setting it too high ($\alpha \geq 0.8$) introduces noisy visual words into the online adaptation process which harms performance.

Figure 10 also shows an example of the online adaptation of our model on the DAVIS-2017 dataset.

B.3. Effect of visual word dictionary size

Table 3 of the main paper already showed the effect of the visual word dictionary size on performance. Figure 11 examines this in more detail by showing the accuracy (measured by the IoU, $\mathcal{J}$), the runtime in seconds and the memory consumption as a function of the size of the visual word dictionary. We can see that the accuracy starts saturating after around $k = 50$ clusters. However, the runtime and memory consumption increase linearly as the number of visual words is increased. This motivates our decision to use $k = 50$ clusters, as it provides a good balance between accuracy and speed.

Additionally, Fig. 12 presents some of the visual words, corresponding to object parts, that are automatically learned by our model.

Finally, Fig. 13 shows the qualitative effect on our final segmentation results by increasing the number of visual words in the dictionary.

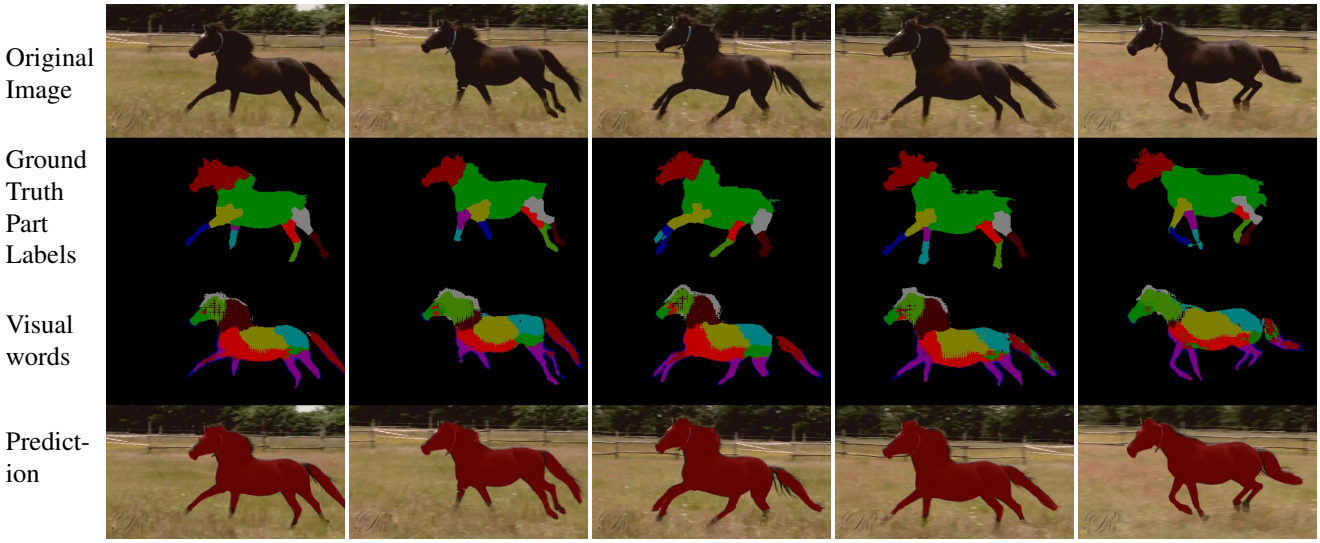

Figure 8: **Qualitative examples of our experiment on the temporal consistency of visual words on the Physical Parts Discovery (PPD) Dataset [10]** Note how the visual words that our model learns in an unsupervised manner (third row) are consistent over time. The PPD dataset contains ground-truth annotations for animal classes (second row) which we leverage for our experiment. Note that our visual words will not correspond exactly to the ground truth parts as we learn our visual words in an unsupervised manner.

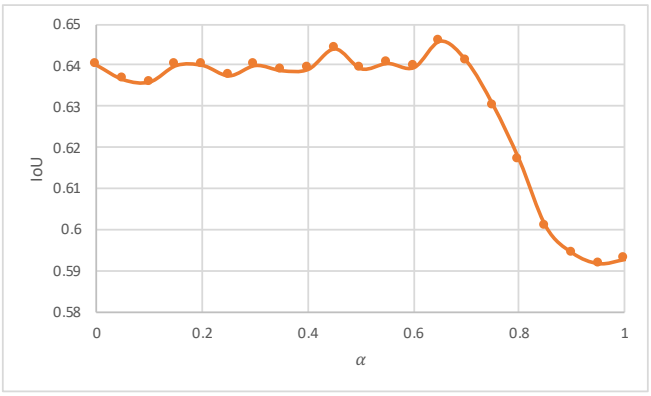

Figure 9: **The effect of the α hyper-parameter used in the online adaptation of on our method.** Note how our IoU on the DAVIS-2017 validation set barely changes for $\alpha \in [0, 0.7]$ showing that our algorithm is not very sensitive to this hyper-parameter.

Figure 10: **The effect of online adaptation on the representation of dynamic objects.** As the object pose changes over time, newer visual words (denoted by gray and light-blue colors) are learned and added to the dictionary of visual words.

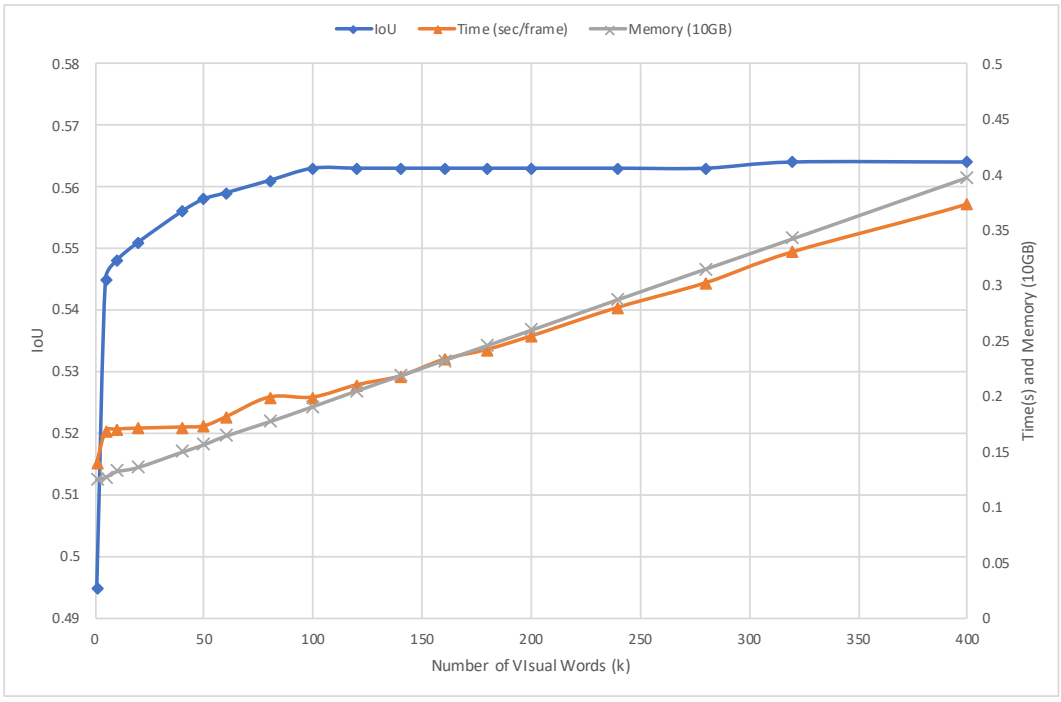

Figure 11: **The effect of visual word dictionary size hyper-parameter on our algorithm's accuracy (IoU), runtime (s) and memory consumption (GB)**. The accuracy, in terms of the IoU ($\mathcal{J}$) starts saturating around $k = 50$. However, the runtime and memory increase linearly with the number of clusters, k. As a result, we use $k = 50$ which provides a good balance between accuracy and speed.

Figure 12: **Visual words formed by our model.** Each row represents a video from DAVIS-2017 dataset, with the original image (left), the object parts formed by our model in different colors (middle), and the segmentation output (right) obtained using our model. It can be seen that our model forms meaningful visual words which represent body parts in objects.

Figure 13: **The effect of visual word object representation on qualitative segmentation outputs.** This figure shows that increasing the size of the visual word dictionary (K) improves the representation of the object, and thus improves segmentation outputs (qualitative). This is because it can better capture the intra-object variance. For instance, the lost face of the human in yellow (first row), the lost tail of the dog (third row), and the missing legs of the horse (last row) have been recovered in the last column (50 visual words), because of this property. Similarly, our method can address partial occlusions by representing different object parts using visual words, and tracking them robustly over the video (fourth row). All the visual words are learned in an unsupervised manner to represent object parts, as described in the main paper. The results are obtained without any fine-tuning.

C. Additional quantitative results

Table 7: Per-sequence video object segmentation results for DAVIS-2016 [35] dataset.

Sequence	Blackswan	Bmx-Trees	Breakdance	Camel	Car-Roundabout	Car-Shadow	Cows	Dance-Twirl	Dog	Drift-Chicane
F	0.98	0.77	0.70	0.85	0.92	0.99	0.98	0.81	0.95	0.72
J	0.94	0.55	0.69	0.81	0.96	0.96	0.94	0.81	0.93	0.65

Sequence	Drift-Straight	Goat	Horsejump-High	Kite-Surf	Libby	Motocross-Jump	Paragliding-Launch	Parkour	Scooter-Black	Soapbox
F	0.83	0.91	0.93	0.64	0.89	0.65	0.45	0.94	0.82	0.89
J	0.90	0.87	0.82	0.62	0.75	0.82	0.61	0.89	0.85	0.90

Table 8: Per-sequence video object segmentation results for DAVIS-2017 [36] dataset.

Sequence	Bike-Packing_1	Bike-Packing_2	Blackswan_1	Bmx-Trees_1	Bmx-Trees_2	Breakdance_1	Camel_1	Car-Roundabout_1	Car-Shadow_1	Cows_1	Dance-Twirl_1	Dog_1	Dogs-Jump_1	Dogs-Jump_2	Dogs-Jump_3
F	0.81	0.63	0.98	0.72	0.73	0.81	0.88	0.96	0.99	0.98	0.83	0.95	0.12	0.70	0.92
J	0.59	0.72	0.95	0.33	0.57	0.76	0.82	0.97	0.95	0.94	0.82	0.92	0.09	0.55	0.85

Sequence	Drift-Chicane_1	Drift-Straight_1	Goat_1	Gold-Fish_1	Gold-Fish_2	Gold-Fish_3	Gold-Fish_4	Gold-Fish_5	Horsejump-High_1	Horsejump-High_2	India_1	India_2	India_3	Judo_1	Judo_2
F	0.70	0.87	0.90	0.58	0.59	0.49	0.60	0.60	0.93	0.89	0.45	0.41	0.56	0.84	0.77
J	0.68	0.91	0.86	0.62	0.56	0.60	0.65	0.75	0.78	0.67	0.46	0.45	0.60	0.81	0.76

Sequence	Kite-Surf_1	Kite-Surf_2	Kite-Surf_3	Lab-Coat_1	Lab-Coat_2	Lab-Coat_3	Lab-Coat_4	Lab-Coat_5	Libby_1	Loading_1	Loading_2	Loading_3	Mbike-Trick_1	Mbike-Trick_2	Motocross-Jump_1
F	0.67	0.26	0.94	0.44	0.51	0.49	0.41	0.62	0.91	0.89	0.55	0.89	0.75	0.76	0.57
J	0.28	0.16	0.72	0.07	0.13	0.55	0.43	0.66	0.76	0.93	0.47	0.84	0.62	0.75	0.48

Sequence	Motocross-Jump_2	Paragliding-Launch_1	Paragliding-Launch_2	Paragliding-Launch_3	Parkour_1	Pigs_1	Pigs_2	Pigs_3	Scooter-Black_1	Scooter-Black_2	Shooting_1	Shooting_2	Shooting_3	Soapbox_1	Soapbox_2	Soapbox_3
F	0.52	0.84	0.83	0.58	0.95	0.55	0.62	0.84	0.77	0.77	0.31	0.55	0.68	0.82	0.81	0.84
J	0.69	0.77	0.58	0.15	0.90	0.52	0.48	0.93	0.64	0.80	0.33	0.59	0.44	0.84	0.75	0.76

Table 9: Per-sequence video object segmentation results for Youtube-Objects dataset [37, 18].

Sequence	Aeroplane	Bird	Boat	Car	Cat	Cow	Dog	Horse	Motorbike	Train
$\mathcal{J}(\%)$	87.1	84.7	81.0	82.9	78.8	76.3	80.3	72.9	77.8	89.5