2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Meta-Learning Deep Visual Words for Fast Video Object Segmentation

Harkirat Singh Behl[1], Mohammad Najafi[1], Anurag Arnab[2] and Philip H.S. Torr[1]

Abstract— **Personal robots and driverless cars need to be able to operate in novel environments and thus quickly and efficiently learn to recognise new object classes. We address this problem by considering the task of video object segmentation. Previous accurate methods for this task 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, causal 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 (whilst being 1 to 2 orders of magnitude faster), and state-of-the-art in terms of speed/accuracy trade-offs on four video segmentation datasets. Code is available at https://github.com/harkiratbehl/MetaVOS.**

I. Introduction

Personal robots and driverless cars need to be able to operate in novel environments, and thus be able to quickly and efficiently learn to recognise object categories that they were not originally trained on. Furthermore, detailed segmentations of objects are also required for applications such as robot manipulation [2], [3], grasping and learning object affordances [4], [3]. Finally, as live camera streams are processed in such applications, efficient and causal algorithms are required. This paper addresses these problems by considering the task of video object segmentation, following the protocol defined in the DAVIS datasets [5], [6]. Here, the ground-truth object mask of one or more objects are provided only in the first frame, which must then be tracked at a pixel-level throughout the rest of the video. Since obtaining even a pixelwise segmentation for a single-frame may be too onerous in robotics applications, we also further extend the problem definition to only provide a bounding-box of each object in the first frame.

Accurate approaches to video segmentation trained a fully convolutional network (FCN) [7] 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 [8], [9], [10], [11], [12], [13]. Although these methods produce accurate results (and can be improved further by using optical flow [14], [15], [16], [17], [18] or post-processing with DenseCRF [19], [8], [15], [20]), they are extremely time consuming, taking between 700s to 3h to finetune per DAVIS video [8], [11], rendering them unsuitable for real-life applications and robotics.

[1]University of Oxford. [2]Google Research. Work primarily done at Oxford.

This paper, in contrast, considers the more challenging (and practical) scenario where the network is not finetuned at all, and uses no optical flow or extra post-processing, in order to develop a fast and causal algorithm. Our approach is inspired by metric-learning methods 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.* [21] used this idea to formulate video segmentation as a pixel-level 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 [22] 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 [21], [23], [24], [25] 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 [21], [25], [10], [13].

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. Moreover, the robustness of this approach allows us to easily extend it to the scenario where we only have weak bounding-box supervision in the first frame.

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

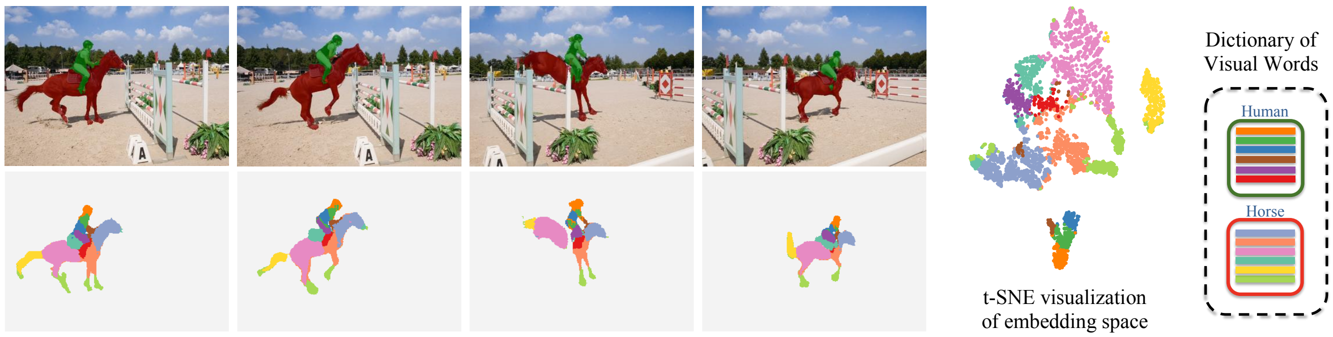
Dictionary of Visual Words
Human
Horse
t-SNE visualization of embedding space

Fig. 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 [1] 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.

inference procedure. This is in contrast to related metric-learning based approaches [21], [24], [23] 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 [26], 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 [27], DAVIS 2017 [6], SegTrack v2 [28], YouTube-Objects [29], [30]) 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 to our knowledge are both faster and more accurate.

II. Related Work

a) Fine-tuning based approaches: The most accurate video segmentation methods using the DAVIS protocol [27], [6] currently finetune models on the first frame of the video [9], [8], [10], [11] and/or use optical flow [17], [18], [16], [14] or DenseCRF [8], [15], [20] post-processing, or use self-paced learning [31] 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.

b) Fast approaches: 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 [32], [26], [33], 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.* [34] addressed this issue using re-identification modules which traverse the video back-and-forth to recover any potential missed objects. However, this method is not causal as it looks at future frames. Oh *et al.* [26] 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, its 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, [26] 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.

c) Metric learning based approaches: Our work is more similar to methods using pixel-to-pixel matching or metric learning [21], [25], [24], [12], [23]. Chen *et al.* [21] 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 [21] 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 [35], [36]. Siamese networks have also been employed in a similar manner [25], [24], [12], 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" [21], [13], [25], [10] 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.

Taking inspiration from classical computer vision, learned feature descriptors have been used in localization [37], motion removal [38] and feature matching [39] because of their robustness and efficiency. Note that our main contributions are that we can learn to segment new object classes in video from very few examples, and use a meta-learning technique to train our model so that the training and testing procedures match each other. The utility of object parts for more robust matching for video segmentation has been identified before by [20]. However, Cheng *et al.* [20] 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.* [20] – 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.

d) Meta-learning: Finally, we note that meta-learning has not been explored much in the context of video segmentation. Yang *et al.* [40] 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 [22] and Matching networks [41] 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.

III. Proposed Approach

We first describe the formulation of video object segmentation as a meta-learning problem. This allows us to train our model in same way that it will be tested, unlike other metric-learning based approaches to video segmentation [21], [25].

A. 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 [42], [43], [44], [45]. 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

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

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 $\mathcal{Q} = \{x_j, y_j\}_{j=1}^{N_Q}$, where N_Q is the number of labelled pixels in the video after the first frame, j is the index. The output of each task $\mathcal{T}$ is the set of predicted class labels for the pixels in $\mathcal{Q}$, $\hat{\mathcal{Y}} = \{\hat{y}_j\}_{j=1}^{N_Q}$.

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.

B. 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. Our method is an online method does not use future-frames during runtime, i.e, to segment a new frame, only the information upto that point is used.

Learning visual words is a challenging task, as we do not have any ground truth information of the object parts that they correspond to. Consequently, as summarised in Fig. 2, we use a meta-training algorithm, where we alternate between the unsupervised learning of deep visual words (Sec. III-B.1) and supervised learning of pixel classification given these visual words (Sec. III-B.2). Our model thus learns to learn a better classifier, by optimising the visual words that it will produce at test-time.

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 $f(\theta)$, which is an artificial neural network with multiple layers between input and output, 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 [46], with μ_{ck} being the respective centroids of the clusters, using the objective:

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

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

In other words, we represent the distribution of the pixels within each set $\mathcal{S}_c$ in the learned embedding space with a

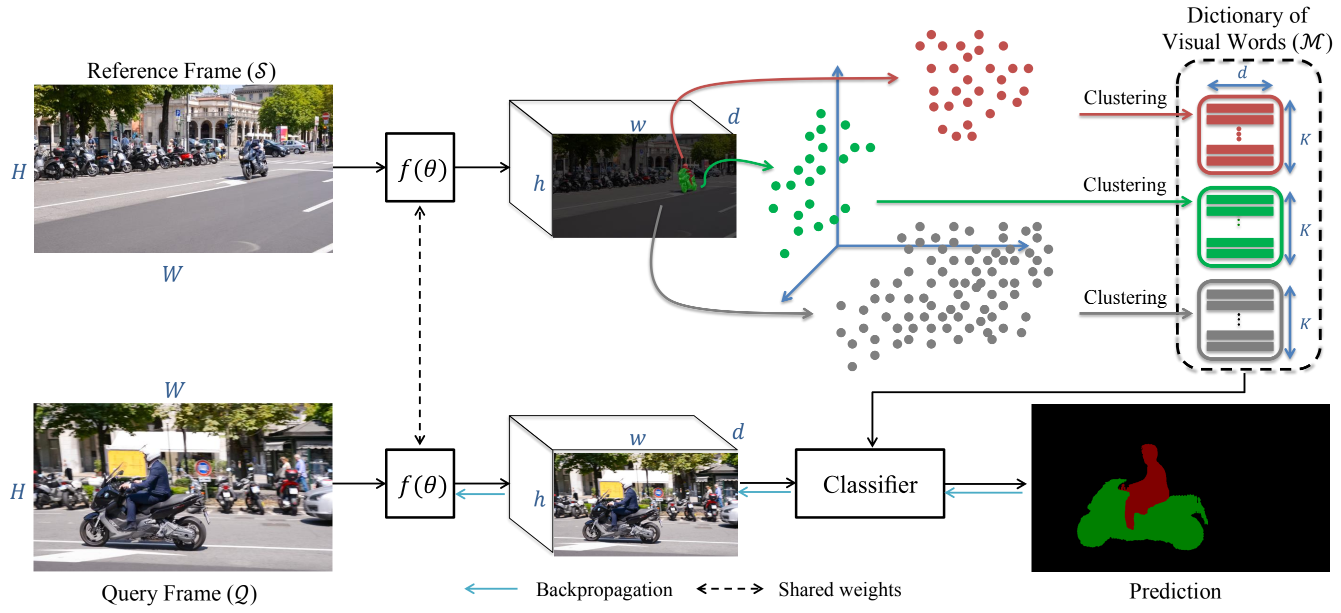

Fig. 2: **Overview of the proposed method.** The first frame of the video (reference frame), which forms the support set $\mathcal{S}$ in our meta-learning setup, passes through a deep segmentation network $f(\theta)$ to compute a $d = 128$ dimensional embedding for each pixel. A dictionary of deep visual 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. During testing, the bottom path, without blue lines, is applied to all frames, whereas, the top path is only applied to the first frame and the online adaptation frames.

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.

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,

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

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 cos 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

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

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 is the cross-entropy loss.

C. 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 cross-entropy loss function 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 method is a form of non-parametric meta-learning, as described in [47]. Note that the cluster centroids can be seen as the parameters of the final classification layer (Eq. 3). Prototypical networks [22] represent each class with a single prototypical vector (*i.e.* one visual word), whilst Matching networks [41] 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 [21] 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 [41], with the key difference being that the training objective (triplet loss) does not correspond to the inference procedure (nearest neighbour search), which is an essential component for meta-learning [47]. Our method ensures that meta-train and meta-test setup match.

D. Online Adaptation

The objects of interest from the first frame, as well as the background, often undergo deformations, occlusions, viewpoint changes. As a result, adapting the model throughout the video is vital to achieve good performance and done by state-of-art video methods [10], [21], [25], [48].

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. III-B.1, and compute their corresponding cluster centroid representations by

$$\mu_{ck}^\delta = \frac{1}{|\mathcal{S}_{ck}^\delta|} \sum_{x_i \in \mathcal{S}_{ck}^\delta} f_\theta(x_i). \tag{5}$$

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, using connected components. 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.

IV. Experimental evaluation

A. Experimental setup

a) Model: We use a Deeplab v2 architecture as the encoder [49], $f(\theta)$, which uses a ResNet-101 [50] 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 manyclusters for the background. For online adaptation, we set $\alpha = 0.5$. The ablation study in Sec. IV-D shows the effect of the number of visual words, k.

b) Datasets: We evaluate on standard video segmentation datasets for tracking both single objects (DAVIS-2016 [27], YouTube-Objects [29], [30]) and multiple objects (DAVIS-2017 [6], SegTrack v2 [28]) 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.

c) Training: Following competing methods which use a model pretrained on image segmentation datasets [25], [26], [21], [40], [51], [10], we initialise the encoder of our network using the public Deeplab-v2 model [49] that has been trained on COCO [52]. Thereafter, we meta-train our model following the "episodic training" procedure, which is the standard practice [41], [22], [45], [53]. 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 $\mathcal{Q}$. Randomly selecting sets of frames in the video make the method robust to temporal discontinuties, occlusions and object complexity, thereby making it more generic. We sample from as low as 3 frames to the entire video length.

d) Evaluation metrics: We report standard metrics defined by the DAVIS protocol [27]: 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" [27] 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 [27] 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. More details and experiments are available in the supplementary material[1].

B. Comparison to state-of-art

Table I 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 [19] and thus have a runtime that is larger by a factor of at least 8 [34], [8].

The only method that is close to us in terms of speed and accuracy is RGMP [26]. However, the runtime of RGMP increases linearly with the number of objects being tracked

[1]Supplementary materials https://harkiratbehl.github.io/

				DAVIS-2017					DAVIS-2016					YouTube-Objects	SegTrack-v2
Method	FT	PP	OF	$\mathcal{J}\&\mathcal{F}$(%)	$\mathcal{J}$(%)	$\mathcal{J}$ Decay(%)	$\mathcal{F}$(%)	Time(s)	$\mathcal{J}\&\mathcal{F}$(%)	$\mathcal{J}$(%)	$\mathcal{J}$ Decay(%)	$\mathcal{F}$(%)	Time(s)	$\mathcal{J}$(%)	$\mathcal{J}$(%)
OnAVOS [10]	✓	✓		65.3	61.6	27.9	69.1	13	85.5	86.1	5.2	84.9	13	77.4	–
OSVOS^S [8]	✓	✓		68.0	64.7	15.1	71.3	–	86.5	**85.6**	5.5	87.5	4.50	**83.2**	65.4
PReMVOS† [9]	✓		✓	**78.2**	**74.3**	16.2	**82.2**	~ 70	**87.0**	85.5	8.8	**88.6**	~ 70	–	–
MaskRNN [17]			✓	–	45.5	–	–	0.60	–	–	–	–	–	–	–
FAVOS [20]		✓		58.2	54.6	**14.1**	61.8	>1.80	80.9	**82.4**	**4.5**	79.5	1.80	–	–
CTN [54]			✓	–	–	–	–	–	71.4	73.5	15.6	69.3	1.33	–	–
FAVOS [20]				–	–	–	–	–	76.9	77.9	–	76.0	0.60	–	–
VPN [33]				–	–	–	–	–	67.8	70.2	12.4	65.5	0.63	–	–
BVS [55]				–	–	–	–	–	59.4	60.0	28.9	58.8	0.37	68.0	60.0
OSMN [40]				54.8	52.5	21.5	57.5	0.50*	–	74.0	9.0	–	0.14	69.0	–
VideoMatch [25]				–	56.5	–	–	0.35	–	81.0	–	–	0.32	79.7	–
RGMP [26]				66.7	**64.8**	18.9	68.6	0.30*	81.7	81.5	10.9	82.0	**0.13**	–	71.1
Ours$^-$				63.1	59.5	55.8	24.6	0.17	76.9	76.2	11.2	77.6	0.17	77.4	64.6
Ours				**67.3**	63.9	14.4	**70.7**	**0.29**	**82.1**	81.5	5.0	**82.7**	0.25	**81.1**	**72.0**

TABLE I: **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. IV-A.

from the first frame. This is because RGMP processes each object instance independently through the "encoder" part of their network [26], 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. I). 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 [26] 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 [27] than RGMP [26] indicating greater robustness. This is also shown qualitatively in Fig. 3 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 typically 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. 3) unlike RGMP.

C. Bounding box based initial mask

In this section, we do experiments when only bounding box supervision is available in the first frame. Hence the aim is to segment the object in the video, given only the bounding box in the first frame. To generate the dictionary of visual words for a given object, we use a mechanism similar to our online adaptation. We first take the embeddings of all pixels in the bounding box of the object and segment them into clusters. We then discard the ones that resemble the background, with the difference that here we use resemblance with the background to eliminate some regions of the bounding box. The remaining clusters are then used to construct the visual word dictionary for this object. The rest of the algorithm remains the same. The results are shown in Table II. It can be seen that there is not a substantial drop in performance in comparison to the mask based case. We believe that our part-based approach makes our model more robust to the noise in the input in this scenario.

	DAVIS-2017	DAVIS-2016	YouTube-Objects	SegTrack-v2
Ours	63.8	81.5	81.1	72.0
Ours-BB	51.5	77.5	75.8	66.5

TABLE II: **Results of our method ($\mathcal{J}\&\mathcal{F}$) with only bounding box based initialization.** Ours-BB: Our model with only bounding box mask provided in first frame.

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

TABLE III: **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 [21].

D. Ablation study

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

a) 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. I). Perhaps surprisingly, our initialisation already outperforms published work like Mask-RNN [17] (Tab. I).

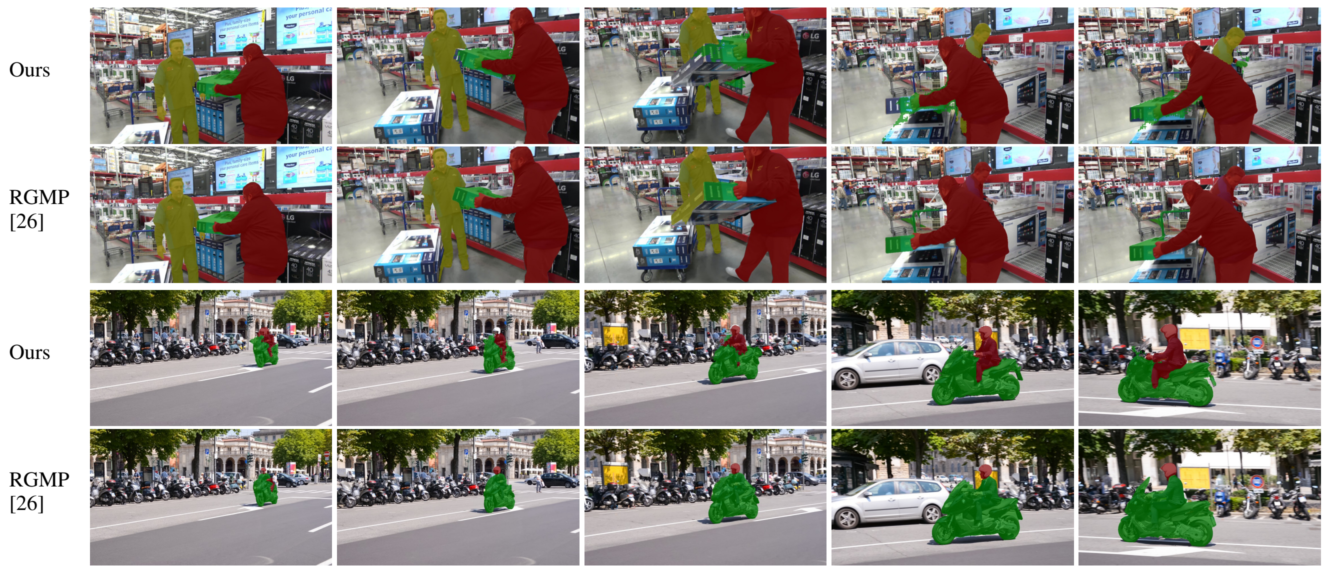

Fig. 3: **Qualitative comparison of our method to RGMP [26].** RGMP obtains good results initially in the video (first two columns), but cannot recover after making errors (third column). Note how it misclassifies the yellow person (first example) and loses track of the rider (second example). In contrast, our method overcomes occlusions in all of these cases by robustly matching an object to its constituent parts.

Dictionary Size (k)	1	5	10	50	100	200	400	1500	3000
$\mathcal{J}$(%)	49.9	54.5	54.8	55.8	56.3	56.3	56.4	56.2	54.5
Time (s)	0.140	0.168	0.170	0.173	0.199	0.254	0.373	0.97	1.42

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

b) Object representation: We represent the object given in the first frame 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 [22]). 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, like [21] and Matching networks [41].

Table III 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 [21] in Tab. I.

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. III, we do the matching at full resolution. The runtime could be greatly reduced by doing the look-up on a subsampled image (for example, [21] do the look-up at $1/8$ resolution which reduce the time by about a factor of 64).

c) Number of visual words: Table IV examines the effect that the number of visual words in the dictionary has on accuracy and runtime on DAVIS-2017. We can see

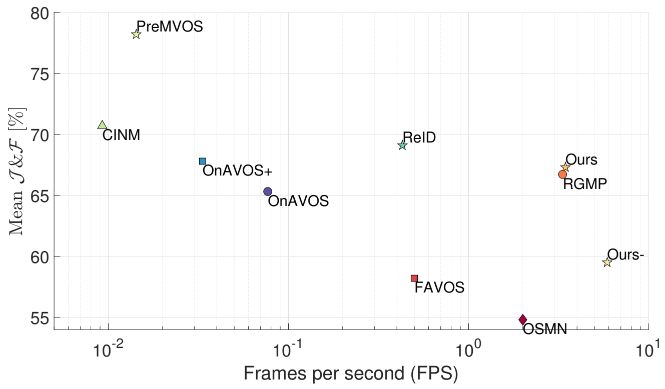

Fig. 4: **Comparison of speed and accuracy on DAVIS 2017.** Entries on the Pareto front (*i.e.* no other method is both faster and more accurate) are marked by a star. Note the speed axis uses a logarithmic scale.

that accuracy steadily increases as the number of clusters is increased from $k = 1$ (which corresponds to Prototypical networks [22]) and plateaus at $k = 50$. We believe that complex objects with high intra-object variations produce embeddings with multi-modal distributions, 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. I. 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 [21].

V. 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. Visual words enable robust matching, as the appearance of 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. Finally, the robustness of our part-based algorithm allows us to easily extend it to the scenario where we only have bounding-box supervision in first frame. Future work is to learn the number of clusters automatically, and learn to generate synthetic data [56] for video segmentation.

Acknowledgements Harkirat is wholly funded by a Tencent grant. This work was supported by the Royal Academy of Engineering, EPSRC/MURI grant EP/N019474/1 and FiveAI.

References

[1] L. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, 2008.

[2] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *ICRA*, 2009.

[3] M. Siam, C. Jiang, S. Lu, L. Petrich, M. Gamal, M. Elhoseiny, and M. Jagersand, "Video object segmentation using teacher-student adaptation in a human robot interaction (hri) setting," in *ICRA*, 2019.

[4] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *ICRA*, 2018.

[5] 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*, 2018.

[6] 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*, 2017.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

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

[9] 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.

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

[11] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid data dreaming for object tracking," in *CVPR Workshops*, 2017.

[12] 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*, 2017.

[13] H. Ci, C. Wang, and Y. Wang, "Video object segmentation by learning location-sensitive embeddings," in *ECCV*, 2018.

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

[15] 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*, 2018.

[16] J. Cheng, Y. Tsai, S. Wang, and M. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *ICCV*, 2017.

[17] Y.-T. Hu, J.-B. Huang, and A. Schwing, "Maskrnn: Instance level video object segmentation," in *NIPS*, 2017.

[18] H. Xiao, J. Feng, G. Lin, Y. Liu, and M. Zhang, "Monet: Deep motion exploitation for video object segmentation," in *CVPR*, 2018.

[19] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *NeurIPS*, 2011.

[20] 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*, 2018.

[21] Y. Chen, J. Pont, A. Montes, and L. Van Gool, "Blazingly fast video object segmentation with pixel-wise metric learning," in *CVPR*, 2018.

[22] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017.

[23] 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*, 2018.

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

[25] Y.-T. Hu, J.-B. Huang, and A. G. Schwing, "Videomatch: Matching based video object segmentation," in *ECCV*, 2018.

[26] S. Wug Oh, J. Lee, K. Sunkavalli, and S. Joo Kim, "Fast video object segmentation by reference-guided mask propagation," in *CVPR*, 2018.

[27] 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.

[28] F. Li, T. Kim, A. Humayun, D. Tsai, and J. Rehg, "Video segmentation by tracking many figure-ground segments," in *ICCV*, 2013.

[29] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *CVPR*, 2012.

[30] S. D. Jain and K. Grauman, "Supervoxel-consistent foreground propagation in video," in *ECCV*, 2014.

[31] D. Zhang, L. Yang, D. Meng, D. Xu, and J. Han, "Spftn: A self-paced fine-tuning network for segmenting objects in weakly labelled videos," in *CVPR*, 2017.

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

[33] V. Jampani, R. Gadde, and P. V. Gehler, "Video propagation networks," in *CVPR*, 2017.

[34] X. Li and C. Change Loy, "Video object segmentation with joint re-identification and attention-aware mask propagation," in *ECCV*, 2018.

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

[36] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," in *arXiv*, 2017.

[37] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019.

[38] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable rgb-d slam in dynamic environments," *RAS*, 2018.

[39] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: learning local features from images," in *NIPS*, 2018.

[40] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos, "Efficient video object segmentation via network modulation," in *CVPR*, 2018.

[41] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NIPS*, 2016.

[42] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-..." Ph.D. dissertation, 1987.

[43] D. K. Naik and R. Mammone, "Meta-neural networks that learn by learning," in *IJCNN*, 1992.

[44] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *ICANN*, 2001.

[45] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[46] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *ACM-SIAM Discrete Algorithms*, 2007, pp. 1027–1035.

[47] C. Finn and S. Levine, "ICML tutorial: Meta-learning," 2019, https://sites.google.com/view/icml19metalearning.

[48] H. S. Behl, M. Sapienza, G. Singh, S. Saha, F. Cuzzolin, and P. Torr, "Incremental tube construction for human action detection," in *BMVC*, 2018.

[49] 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*, 2018.

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

[51] 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.

[52] 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 *ECCV*, 2014.

[53] H. S. Behl, A. G. Baydin, and P. H. S. Torr, "Alpha maml: Adaptive model-agnostic meta-learning," *ICML Workshops*, 2019.

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

[55] N. Maerki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, "Bilateral space video segmentation," in *CVPR*, 2016.

[56] H. S. Behl, A. G. Baydin, R. Gal, P. Torr, and V. Vineet, "Autosimulate: (quickly) learning synthetic data generation," in *ECCV*, 2020.