Self-supervised Video Object Segmentation by Motion Grouping

Charig Yang Hala Lamdouar Erika Lu
Andrew Zisserman Weidi Xie
Visual Geometry Group, University of Oxford
{charig,lamdouar,erika,az,weidi}@robots.ox.ac.uk

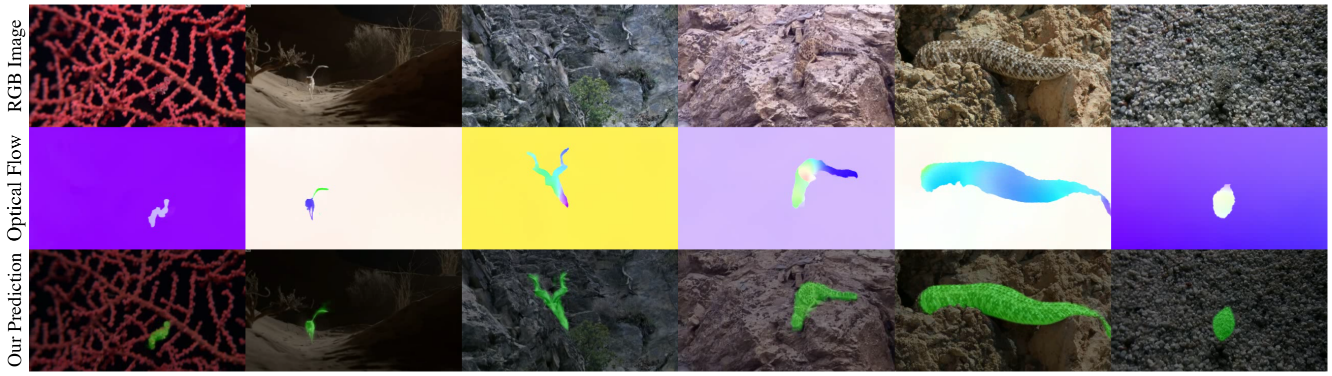

Figure 1. **Segmenting camouflaged animals.** Motion plays a critical role in augmenting the capability of our visual system for perceptual grouping in complex scenes – for example, in these sequences (MoCA dataset [41]), the visual appearance (RGB images) is clearly uninformative. We propose a self-supervised approach to segment objects using *only* motion, *i.e.* optical flow. From top to bottom rows, we show the video frames, optical flow between consecutive frames, and the segmentation produced by our approach.

Abstract

Animals have evolved highly functional visual systems to understand motion, assisting perception even under complex environments. In this paper, we work towards developing a computer vision system able to segment objects by exploiting motion cues, i.e. *motion segmentation. We make the following contributions: First, we introduce a simple variant of the Transformer to segment optical flow frames into primary objects and the background. Second, we train the architecture in a self-supervised manner,* i.e. *without using any manual annotations. Third, we analyze several critical components of our method and conduct thorough ablation studies to validate their necessity. Fourth, we evaluate the proposed architecture on public benchmarks (DAVIS2016, SegTrackv2, and FBMS59). Despite using only optical flow as input, our approach achieves superior or comparable results to previous state-of-the-art self-supervised methods, while being an order of magnitude faster. We additionally evaluate on a challenging camouflage dataset (MoCA), significantly outperforming the other self-supervised approaches, and comparing favourably to the top supervised approach, highlighting the importance of motion cues, and the potential bias towards visual appearance in existing video segmentation models.*

1. Introduction

When looking around the world, we effortlessly perceive a complex scene as a set of distinct objects. This phenomenon is referred to as *perceptual grouping* – the process of organizing the incoming visual information – and is usually considered a fundamental cognitive ability that enables understanding and interacting with the world efficiently. How do we accomplish such a remarkable perceptual achievement, given that the visual input is, in a sense, just a spatial distribution of variously colored individual points/pixels? In 1923, Wertheimer [80] first introduced the *Gestalt principles* with the goal of formulating the underlying causes by which sensory data is organized into groups, or Gestalten. The principles are much like heuristics with "a bag of tricks" [59] that the visual system may exploit for grouping, for example, proximity, similarity, closure, continuation, common fate, *etc*.

In computer vision, *perceptual grouping* is often closely related to the problem of segmentation, *i.e.* extracting the objects with arbitrary shape (pixel-wise labels) from cluttered scenes. In the recent literature of semantic or instance segmentation, tremendous progress has been made by training deep neural networks on image or video datasets. While it is exciting to see machines with the ability to detect, segment, and classify objects in images or video frames, training such segmentation models through supervised learning requires massive human annotation, and consequently limits their scalability. Even more importantly, the assumption that objects can be well-identified by their appearance alone in static frames is often an oversimplification – objects are not always visually distinguishable from their background environment. For instance when trying to discover camouflaged animals/objects from the background (Figure 1), extra cues, such as motion or sound, are usually required.

Among the numerous cues, motion is usually simple to obtain as it can be implicitly generated from unlabeled videos. In this paper, we aim to exploit such cues for object segmentation in a self-supervised manner, *i.e. zero* human annotation is required for training. At a high level, we aim to exploit the common fate principle, with the basic assumption being that **elements tend to be perceived as a group if they move in the same direction at the same rate (have similar optical flow)**. Specifically, we tackle the problem by training a generative model that decomposes the optical flow into foreground (object) and background layers, describing each as a homogeneous field, with discontinuities occurring only between layers. We adopt a variant of the Transformer [3], with the self-attention being replaced by slot attention [46], where iterative grouping and binding have been built into the architecture. With some critical architectural changes, we show that pixels undergoing similar motion are grouped together and assigned to the same layer.

To summarize, we make the following contributions: *first*, we introduce a simple architecture for video object segmentation by exploiting motions, using only optical flow as input. *Second*, we propose a self-supervised proxy task that is used to train the architecture without any manual supervision. *Third*, we conduct thorough ablation studies on the components that are key to the success of our architecture, such as a consistency loss on optical flow computed from various frame gaps. *Fourth*, we evaluate the proposed architecture on public benchmarks (DAVIS2016 [55], SegTrackv2 [42], and FBMS59 [56]), outperforming previous state-of-the-art self-supervised models, with comparable performance to the supervised approaches. Moreover, we also evaluate on a camouflage dataset (MoCA [41]), demonstrating a significant performance improvement over the other self- and supervised approaches, highlighting the importance of motion cues, and the potential bias towards visual appearance in existing video segmentation models.

2. Related Work

Video object segmentation has been a longstanding task in computer vision, which involves segmenting the pixels (or edges) belonging to an image into groups (e.g. objects). In recent literature [5, 10, 12, 16, 24, 25, 31, 34, 35, 39, 40, 51, 52, 53, 54, 57, 57, 71, 73, 74, 75, 77, 84, 84, 87], two protocols have attracted increasing interest from the vision community, namely, semi-supervised video object segmentation (**semi-supervised VOS**), and unsupervised video object segmentation (**unsupervised VOS**). The former aims to re-localize one or multiple targets that are specified in the first frame of a video with pixel-wise masks, and the latter considers automatically separating the object of interest (usually the most salient one) from the background in a video sequence. Despite being called **unsupervised VOS**, in practice, the popular methods to address such problems extensively rely on supervised training, for example, by using two-stream networks [16, 31, 54, 71] trained on large-scale external datasets. As an alternative, in this work, we consider a *completely unsupervised* problem, where no manual annotation is used for training whatsoever.

Motion segmentation shares some similarity with unsupervised VOS, but focuses on discovering *moving* objects. In the literature, [10, 36, 52, 63, 83] consider clustering the pixels with similar motion patterns; [16, 70, 71] train deep networks to map the motions to segmentation masks. Another line of work has tackled the problem by explicitly leveraging the independence, in the flow field, between the moving object and its background. For instance, [86] proposes an adversarial setting, where a generator is trained to produce masks, altering the input flow, such that the inpainter fails to estimate the missing information. In [6, 7, 41], the authors propose to highlight the independently moving object by compensating for the background motion, either by registering consecutive frames, or explicitly estimating camera motion. In constrained scenarios, such as the autonomous driving domain, [61] proposes to jointly optimize depth, camera motion, optical flow and motion segmentation.

Optical flow computation is one of the fundamental tasks in computer vision. Deep learning methods allow efficient computation of optical flow, both in supervised learning on synthetic data [68, 69], and in the self-supervised [44, 45] setting. Flow has been useful for a wide range of problems, occasionally even used in lieu of appearance cues (RGB images) for tracking [62], pose estimation [19], representation learning [50], and motion segmentation [10].

Transformer architectures have proven extremely adept at modeling long-term relationships within an input sequence via attention mechanisms. Originally used for language tasks [3, 9, 18], they have since been adapted for solving popular computer vision problems such as image classification [20], generation [14, 60], video understanding [4, 27, 79], object detection [13], and zero-shot classification [58]. In this work, we take inspiration from a specific variant of self-attention, namely slot attention [46], which was demonstrated to be effective for learning object-centric representations on synthetic datasets, *e.g.* CLEVR.

Layered representations were originally proposed by Wang and Adelson [76] to represent a video as a composition of layers with simpler motions. Since then, layered representations of images and videos have been widely adopted in computer vision [8, 33, 38, 85, 90], often to estimate optical flow [66, 67, 81, 82]. More recently, deep learning-based layer decomposition methods have been used to infer depth for novel view synthesis [65, 88], separate reflections and other semi-transparent effects [1, 2, 26, 47], or perform foreground/background estimation [26]. These works operate on RGB inputs and produce RGB layers, whereas we propose a layered decomposition of optical flow inputs for unsupervised moving object discovery.

Object-centric representations interpret scenes with "objects" as the basic building block (instead of individual pixels), which is considered an essential step towards human-level generalization. There is a rich literature on this topic, for example, in [29], the IODINE model uses iterative variational inference to infer a set of latent variables recurrently, with each representing one object in an image. Similarly, MONet [11] and GENESIS [21] also adopt the multiple encode-decode steps. In contrast, [46] proposes Slot Attention, which enables single step encoding-decoding with iterative attention. However, all of the works mentioned above have only shown applications for synthetic datasets, *e.g.* CLEVR [32]. In this paper, we are the first to demonstrate its use for object segmentation of realistic videos by exploiting motion, where the challenging nuances in visual appearance (*e.g.* the complex background textures) have been removed.

3. Method

Our goal is to take an input optical flow frame and predict a segment containing the moving object. We aim to train this model in a self-supervised manner, specifically, we adopt an autoencoder-like framework. Our model outputs two layers: one representing the background, and the other for one or more moving objects in the foreground, as well as their oppacity layers (weighted masks). Formally, we have:

$$\{\hat{I}^i_{t\to t+n}, \alpha^i_{t\to t+n}\}^N_{i=1} = \Phi(I_{t\to t+n}) \tag{1}$$

where $I_{t\to t+n}$ refers to the t to $t+n$ input flow (backward flow when $n < 0$), $\Phi(\cdot)$ is the parametrized model, $\hat{I}^i_{t\to t+n}$ is the ith layer reconstruction, $\alpha^i_{t\to t+n}$ is its mask, and $N = 2$ is the number of layers (foreground and background). These layers can then be composited linearly to reconstruct the input image $I_{t\to t+n}$:

$$\hat{I}_{t\to t+n} = \sum_{i=1}^{N} \alpha^i_{t\to t+n} \hat{I}^i_{t\to t+n} \tag{2}$$

3.1. Flow Segmentation Architecture

For simplicity, we first consider the case of a single flow field as input (depicted in the top part of Figure 2). The entire model consists of three components: (1) a CNN encoder to extract a compact feature representation, (2) an iterative binding module with learnable queries that plays a similar role as soft clustering, *i.e.* assigning each pixel to one of motion groups, and (3) a CNN decoder that individually decodes each query to full resolution layer outputs (where thresholding the alpha channel yields the predicted segment).

CNN Encoder. We first pass the precomputed optical flow field between two frames, $I_{t\to t+n} \in \mathcal{R}^{3\times H_0\times W_0}$, to a CNN encoder Φ_{enc}, which outputs a lower-resolution feature map:

$$F_{t\to t+n} = \Phi_{\text{enc}}(I_{t\to t+n}) \in \mathcal{R}^{D\times H\times W} \tag{3}$$

where H_0, W_0 and H, W refer to the spatial dimensions of the input and output feature maps respectively. Note that, we convert the optical flow field into a three-channel image according to the traditional method specified in the optical flow literature [68].

Iterative Binding. The iterative binding module Φ_{bind} aims to group image regions into single entities based on their similarities in motion, *i.e.* pixels moving in the same direction at the same rate should be grouped together. Intuitively, such a binding process requires a data-dependent parameter updating mechanism, iteratively adapting and enriching the model, gradually including more pixels undergoing similar motions.

To accomplish this task, we adopt a simple variant of slot attention [46], where instead of Gaussian-initialized slots, we use learnable query vectors. Slot attention has recently shown remarkable performance for object-centric representation learning, where the query vectors compete to explain parts of the inputs via a softmax-based attention mechanism, and the representations of these slots are iteratively updated with a recurrent update function. In our case of motion segmentation, ideally, the final representation in each query vector separately encodes the moving object or the background, which can then be decoded and combined to reconstruct the input flow fields.

Formally, our inputs to Φ_{bind} are feature maps $F_{t\to t+n}$ and two learnable queries (representing foreground and

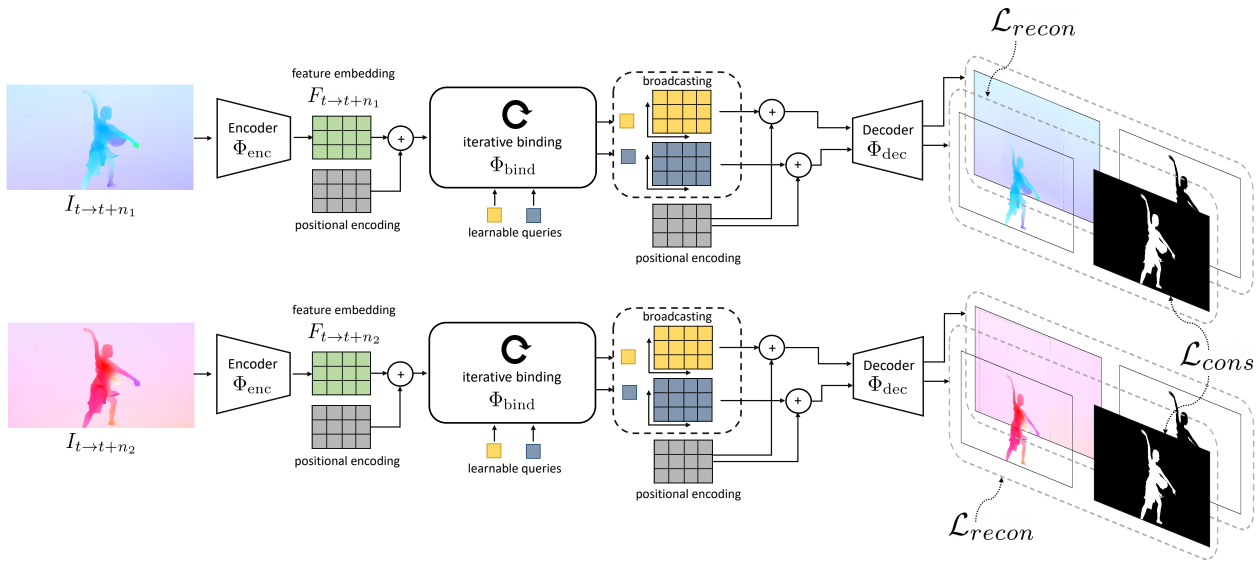

Figure 2. **Pipeline.** Our model takes optical flow $I_{t\to t+n}$ as input, and outputs a set of reconstruction and opacity layers. Specifically, it consists of three components: feature encoding, iterative binding, and decoding to layers, which are combined to reconstruct the input flow. To resolve motion ambiguities (small motion), or noise in optical flow, consistency between two flow fields computed under different frame gaps is enforced during training. At inference time, only the top half of the figure is used to predict masks from a single-step flow.

background) $Q \in \mathcal{R}^{D\times 2}$. Learnable spatial positional encodings are summed with $F_{t\to t+n}$; with some abuse of notation, we still refer to this sum as $F_{t\to t+n}$. We use *three* different linear transformations to generate the *query* , *key* and *value*: $q \in \mathcal{R}^{D\times 2}$, $k, v \in \mathcal{R}^{D\times HW}$,

$$q,\ k,\ v = W^Q \cdot Q,\ W^K \cdot F_{t\to t+n},\ W^V \cdot F_{t\to t+n} \quad (4)$$

where $W^Q, W^K, W^V \in \mathcal{R}^{D\times D}$.

In contrast to the standard Transformer [3], the coefficients in slot attention are normalized over all slots. This choice of normalization introduces competition between the slots to explain parts of the input, and ensures each pixel is assigned to a query vector:

$$\text{attn}_{i,j} := \frac{e^{M_{i,j}}}{\sum_l e^{M_{i,l}}} \quad (5)$$
$$M := \frac{1}{\sqrt{D}} k^T \cdot q,\ \ \text{attn} \in \mathcal{R}^{HW\times 2}$$

To aggregate the input values to their assigned query slot, a weighted mean is used as follows:

$$U := v \cdot A \ \ \in \mathcal{R}^{D\times 2} \quad (6)$$
$$\text{where,}\ \ A_{i,j} := \frac{\text{attn}_{i,j}}{\sum_l \text{attn}_{l,j}}$$

To maintain a smooth update of the query slots Q, the aggregated vectors U are fed into a recurrent function, parametrized with Gated Recurrent Units (GRU),

$$Q := \text{GRU}(\text{inputs} = U, \text{states} = Q) \quad (7)$$

This whole binding process is then iterated T times. The pseudocode can be found in the Supplementary Material.

CNN Decoder. The CNN decoder Φ_{dec} individually decodes each of the slots to outputs of original resolution ($\{\hat{I}^i_{t\to t+n}, \alpha^i_{t\to t+n}\} \in \mathcal{R}^{4\times H_0\times W_0}$), which includes an (unnormalized) single-channel alpha mask and the reconstructed flow fields. Specifically, the input to the decoder is the slot vector broadcasted onto a 2D grid augmented with a learnable spatial positional encoding.

Reconstruction. Once each slot has been decoded, we apply softmax to the alpha masks across the slot dimension, and use them as mixture weights to obtain the reconstruction $\hat{I}_{t\to t+n}$ (Eq. 2). Our reconstruction loss is an L2 loss between the input and reconstructed flow,

$$\mathcal{L}_{recon} = \frac{1}{\Omega}\sum_{p\in\Omega} |I_{t\to t+n}(p) - \hat{I}_{t\to t+n}(p)|^2 \quad (8)$$

where p is the pixel index, and Ω is the entire spatial grid.

Entropy Regularization. We impose a pixel-wise entropy regularisation on inferred masks:

$$\mathcal{L}_{entr} = \frac{1}{\Omega}\sum_{p\in\Omega}(-\alpha^1_{t\to t+n}(p)\log\alpha^1_{t\to t+n}(p) \quad (9)$$
$$-\alpha^0_{t\to t+n}(p)\log\alpha^0_{t\to t+n}(p))$$

This loss is zero when the alpha channels are one-hot, and maximum when they are of equal probability. Intuitively, this helps encourage the masks to be binary, which aligns with our goal in obtaining segmentation masks.

Instance Normalisation. In the case of motion segmentation, objects can only be detected if they undergo an independent motion from the camera; thus previous work attempts to compensate for camera motion [6, 41]. We are inspired by these ideas, but instead of explicitly estimating homography or camera motion, we take a poor-man's approach by simply using Instance Normalisation (IN) [72] in the CNN encoder and decoder, which normalizes each channel in each training sample independently. Intuitively, the *mean* activation tends to be dominated by the motions in the large homogeneous region, which is usually the background. This normalization, in combination with ReLU activations, helps gradually separate the background motion from the foreground motions.

3.2. Self-supervised Temporal Consistency Loss

The segmentation computed for the current frame should be identical irrespective of whether the 'second' frame is consecutive, or earlier or later in time. We harness this constraint to form a self-supervised temporal consistency loss by first defining a set of 'second' frames, and then requiring consistency between their pairwise predictions. We describe the set first, followed by the loss.

Multi-step flow. As objects may be static for some frames, we make our predictions more robust by leveraging observations from multiple timesteps. We consider the flow fields computed from various temporal gaps as an input set, *i.e.* $\{I_{t \to t+n_1}, I_{t \to t+n_2}\}$, $n_1, n_2 \in \{-2, -1, 1, 2\}$, and use a permutation invariant consistency loss to encourage the model to predict the same foreground/background segmentation for all flow fields in the set.

Consistency loss. We randomly sample two flow fields from the input set and pass them through the model ($\Phi(\cdot)$), outputting the flow reconstruction and alpha masks for each. As the reconstruction loss is commutative, it is not guaranteed that the same slot will always output the background layer; therefore, we use a permutation-invariant consistency loss, *i.e.* only backpropagating through the lowest-error permutation:

$$\mathcal{L}_{cons} = \frac{1}{\Omega}\min\Big(\sum_{p\in\Omega} |\alpha^1_{t\to t+n_1}(p) - \alpha^1_{t\to t+n_2}(p)|^2, \\ \sum_{p\in\Omega} |\alpha^1_{t\to t+n_1}(p) - \alpha^0_{t\to t+n_2}(p)|^2\Big)$$

Note that, this consistency enforcement only occurs during training. At inference time, a single-step flow is used, as shown in the top half of Figure 2.

Total Loss. The total loss for training the architecture is:

$$\mathcal{L}_{total} = \gamma_r \mathcal{L}_{recon} + \gamma_c \mathcal{L}_{cons} + \gamma_e \mathcal{L}_{entr} \tag{10}$$

we use $\gamma_r = 10^2$, $\gamma_c = 10^{-2}$ and $\gamma_e = 10^{-2}$. However, we found the model is fairly robust to these hyper-parameters.

3.3. Discussion

Differences from slot attention. Slot attention was originally introduced for self-supervised object segmentation for RGB images [46], and its usefulness was demonstrated on synthetic data (CLEVR [32]), where objects are primitive shapes with simple textures. However, this assumption is unlikely to hold in the case of natural images or videos, making it challenging to generalize such object-centric representations.

In this work, we build on the insight that although objects in images may not be naturally textureless, their motions typically are. Hence, we develop the self-supervised object segmentation model by exploiting their optical flows, where the nuance in visual appearance is discarded, thus not restricted to simple synthetic cases. As an initial trial, we experimented with the same setting as [46], where query vectors are sampled from a Gaussian distribution; however, we were unable to train it. Instead, we use a learnable embedding here, which we highlight as one of the architectural changes critical to our model's success. Other critical changes include instance normalization and temporal consistency, which we demonstrate in ablations in Section 5.1.

Why does it work for motion segmentation? The proposed idea can be seen as training a generative model to segment the flow fields. With the layered formulation, reconstruction is limited to be a simple *linear* composition of layer-wise flow, decoded from a single slot vector.

Conceptually, this design has effectively introduced a representational bottleneck, encouraging each slot vector to represent minimal information, *i.e.* homogeneous motion, and with minimal redundancy (mutual information) between slots. All these properties make such an architecture well-suited to the task of segmenting objects undergoing independent motions.

4. Experimental Setup

In this section, we describe the datasets and evaluation metrics, followed by implementation details.

4.1. Datasets

We benchmark on four different datasets that are commonly used for unsupervised video object segmentation.

DAVIS2016 [55] contains a total of 50 sequences (30 for training and 20 for validation), depicting diverse moving objects such as animals, people, and cars. The dataset contains 3455 1080p frames with pixel-wise annotations at 480p for the predominantly moving object.

SegTrackv2 [42] contains 14 sequences and 976 annotated frames. Each sequence contains 1-6 moving objects, and presents challenges including motion blur, appearance change, complex deformation, occlusion, slow motion, and interacting objects.

FBMS59 [56] consists of 59 sequences and 720 annotated frames (every 20th frame is annotated), which vary greatly in image resolution. Sequences involve multiple moving objects, some of which may be static for periods of time.

Moving Camouflaged Animals (MoCA) [41] contains 141 HD video sequences, depicting 67 kinds of camouflaged animals moving in natural scenes. Both temporal and spatial annotations are provided in the form of tight bounding boxes for every 5th frame. Using the provided motion labels (locomotion, deformation, static), we filter out videos with predominantly no locomotion, resulting in 88 video sequences and 4803 frames (this subset will be released).

4.2. Evaluation Metrics

We use two different evaluation metrics, depending on whether pixel-wise annotations or bounding boxes are provided.

Segmentation (Jaccard). For DAVIS2016, SegTrackv2 and FBMS59, pixelwise segmentation is provided; thus we report the standard metric, region similarity ($\mathcal{J}$), computing the mean over the test set. For FBMS59 and SegTrackv2, we follow the common practice [30, 86] and combine multiple objects as one single foreground.

Localization (Jaccard & Success Rate). As the MoCA dataset provides only bounding box annotations, we evaluate for the detection task and report results in the form of detection success rate [22, 43], for varying IoU thresholds ($\tau \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$).

4.3. Implementation Details

We evaluate three different approaches for computing optical flow, namely, PWC-Net [68], RAFT [69] and ARFlow [44]; the first two are supervised, while the latter is self-supervised. We extract the optical flow at the original resolution of the image pairs, with the frame gaps $n \in \{-2, -1, 1, 2\}$ for all datasets, except for FBMS59, where we use $n \in \{-6, -3, 3, 6\}$ to compensate for small motion. To generate inputs to the network for training, the flows are resized to 128×224 (and scaled accordingly), converted to 3-channel images with the standard visualization used for optical flow, and normalized to $[-1, 1]$.

In the iterative binding module ($\Phi_{\text{bind}}(\cdot)$), we use two learnable query vectors (as we consider the case of segmenting a single moving object from the background), and choose $T = 5$ iterations (as explained in Section 3.1). We adopt a simple VGG-style network for the CNN encoder and decoder, with 3 blocks each; in each block, we use 2 sets of convolutions, instance normalization and ReLU activation. We train with a batch size of 64 images and use the Adam optimizer [37] with an initial learning rate of 5×10^{-4}, decreasing every 80k iterations. The exact architecture description and training schedule can be found in the Supplementary Material, and the submitted source code.

5. Results

In this section, we compare primarily with a top-performing approach trained without manual annotations – Contextual Information Separation (CIS [86]). However, as the architecture, input resolution, modality and post-processing are all different, we try our best to conduct the comparison as fairly as possible. Note that benchmarks are evaluated at full resolution by simply upsampling the predicted masks.

In order to guarantee a fair comparison of the model, we disregard the gains from post-processing, which typically include averaging an ensemble of multiple crops and flow steps, temporal smoothing, and CRFs, at the expense of runtime. Here, we consider only raw framewise predictions with single-step flow at over 80fps, which is more likely to be the application of motion segmentation in practice.

5.1. Ablation Studies

We conduct all ablation studies on DAVIS2016, and vary one variable each time, as shown in Table 1.

Choice of Optical Flow Algorithm. With the same flow extraction method (PWC-Net), our proposed model (Ours-A) outperforms CIS by about 4.5 points on mean Jaccard ($\mathcal{J}$), and using improved optical flow (RAFT) provides further performance gains. We therefore use RAFT from hereon.

Instance Normalization and Grouping. We observe two phenomena: *first*, when holding constant the number of grouping iterations T (3 or 5), models trained with instance normalization perform consistently better; *second*, iterative grouping with $T = 5$ is better than that trained with $T = 3$. However, at $T = 8$, the model did not converge in the same number of training steps, and thus we do not include it in the table. For the remainder of the experiments, we use instance normalization and $T = 5$ to balance performance and training time.

Consistency and Entropy Regularization. While comparing Ours-B and Ours-I, we observe that the performance degrades significantly without the temporal consistency loss, and that the entropy regularization is also important, as shown by Ours-B and Ours-H.

5.2. Comparison with State-of-the-art

We show comparison results for different datasets in Table 2. On DAVIS2016, we improve upon the state-of-the-art

Model	Flow	IN	T	$\mathcal{L}_e$	$\mathcal{L}_c$	DAVIS ($\mathcal{J}\uparrow$)
CIS [86]	PWC-Net	–	–	–	–	59.2
Ours-A	PWC-Net	✓	5	✓	✓	63.7
Ours-B	**RAFT**	✓	**5**	✓	✓	**68.3**
Ours-C	ARFlow	✓	5	✓	✓	53.2
Ours-D	RAFT	✓	3	✓	✓	65.8
Ours-E	RAFT	✗	3	✓	✓	63.3
Ours-F	RAFT	✗	5	✓	✓	64.5
Ours-G	RAFT	✓	5	✗	✗	48.0
Ours-H	RAFT	✓	5	✗	✓	60.3
Ours-I	RAFT	✓	5	✓	✗	51.2

Table 1. **Ablation studies** on flow extraction methods, instance normalization (IN), grouping iterations (T), entropy regularization ($\mathcal{L}_e$) and set consistency ($\mathcal{L}_c$).

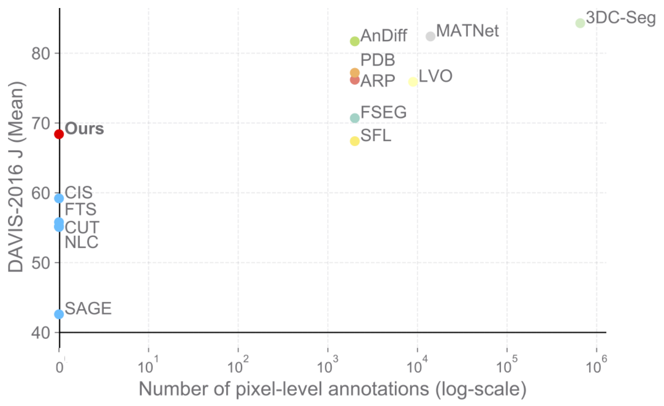

Figure 3. **Comparison on DAVIS2016.** Note that, supervised approaches may use models pretrained on ImageNet [17], but here we only count number of images with pixel-wise annotations.

Model	Sup.	RGB	Flow	Res.	DAVIS16 ($\mathcal{J}\uparrow$)	STv2 ($\mathcal{J}\uparrow$)	FBMS59 ($\mathcal{J}\uparrow$)	Runtime (sec $\downarrow$)
SAGE [78]	✗	✓	✓	–	42.6	57.6	**61.2**	0.9s
NLC [23]	✗	✓	✓	–	55.1	**67.2**	51.5	11s
CUT [36]	✗	✓	✓	–	55.2	54.3	57.2	103s
FTS [54]	✗	✓	✓	–	55.8	47.8	47.7	0.5s
CIS [86]	✗	✓	✓	192 × 384	59.2 (71.5)	45.6 (62.0)	36.8 (63.5)	0.1s (11s)
Ours	✗	✗	✓	128 × 224	**68.3**	58.6	53.1	**0.012s**
SFL [15]	✓	✓	✓	854 × 480	67.4	–	–	7.9s
FSEG [30]	✓	✓	✓	854 × 480	70.7	61.4	68.4	–
LVO [71]	✓	✓	✓	–	75.9	57.3	65.1	–
ARP [64]	✓	✓	✓	–	76.2	57.2	59.8	74.5s
COSNet [48]	✓	✓	✗	473 × 473	80.5	–	75.6	–
MATNet [89]	✓	✓	✓	473 × 473	82.4	–	–	0.55s
3DC-Seg [49]	✓	✓	✓	854 × 480	84.3	–	–	0.84s

Table 2. **Full comparison on moving object segmentation** (unsupervised video segmentation). We consider three popular datasets, DAVIS2016, SegTrack-v2 (STv2), and FBMS59. Models above the horizontal dividing line are trained without using *any* manual annotation, while models below are pre-trained on image or video segmentation datasets, *e.g.* DAVIS, YouTube-VOS, thus requiring ground truth annotations at training time. Numbers in parentheses denote the additional usage of significant post-processing, *e.g.* multi-step flow, multi-crop, temporal smoothing, CRFs.

for unsupervised methods (CIS) by a large margin (+9.1%). As shown in Figure 5, although we do not use any pixel-level annotations during training, our method is nearing the performance of supervised models trained on thousands of images. In addition, we argue that, motion segmentation in realistic scenarios, *e.g.* by predator or prey, is likely to require fast processing. Our model operates on small resolution (potentially sacrificing some accuracy) with over 80fps.

For SegTrackv2 and FBMS59, they occasionally include multiple objects in a single video, and only a subset of them are moving, making it challenging to spot all objects using flow-only input, but we achieve competitive performance nonetheless. We discuss this limitation below.

5.3. Camouflage Breaking

In addition to evaluating on moving object segmentation, we also benchmark the model on camouflaged object detection on the MoCA dataset, where visual cues are often less effective than motion cues. To compare fairly with CIS [86], we use the code and model released by the authors, and fine-tune their model on MoCA in a self-supervised manner. We convert our model's output segmentation mask to a bounding box by keeping only the largest connected region in the prediction before taking the bounding box around it.

We report quantitative results in Table 3 and show qualitative results in Figure 4. Our model significantly outperforms CIS (14% when allowing no post-processing), pre-

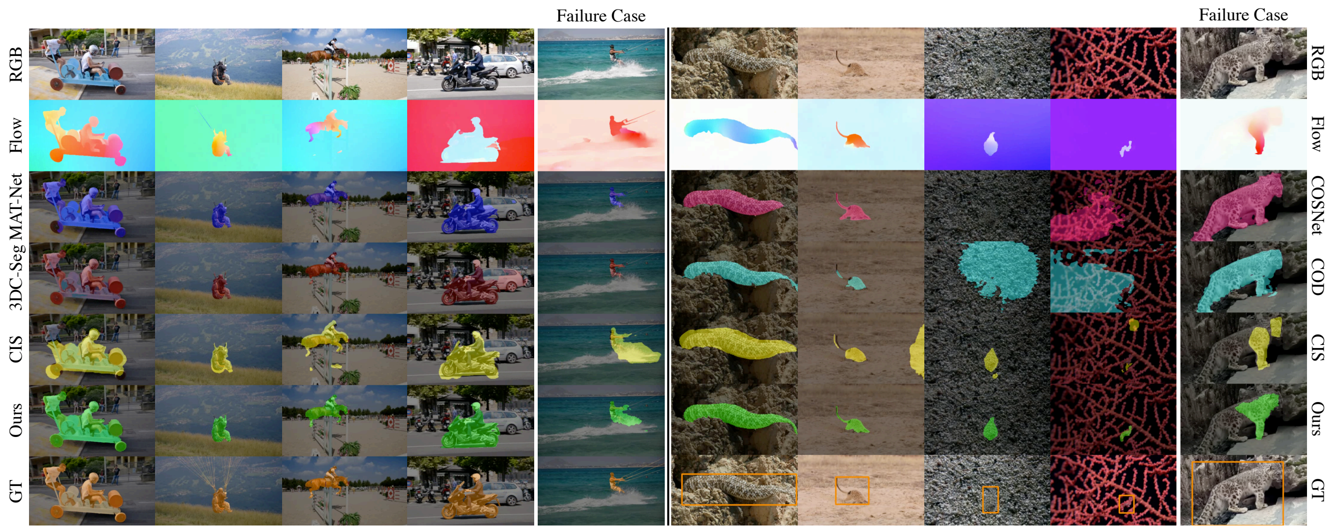

Figure 4. **Qualitative results.** On DAVIS2016 (left), our method is able to segment a variety of challenging objects such as people, animals, and vehicles, often on-par with top supervised approaches. On MoCA (right), our model is able to accurately segment well-camouflaged objects even when previous supervised methods fail completely (3rd, 4th columns). We show a failure case (left) where the splash created by the person is incorrectly included in our predicted segment, and another failure case (right) where the animal is only partially moving and thus partially segmented.

					Success Rate					
Model	Sup.	RGB	Flow	$\mathcal{J}\uparrow$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$	SR_{mean}
COD [41]	✓	✗	✓	44.9	0.414	0.330	0.235	0.140	0.059	0.236
COD (two-stream) [41]	✓	✓	✓	55.3	0.602	0.523	0.413	0.267	0.088	0.379
COSNet [48]	✓	✓	✗	50.7	0.588	0.534	0.457	0.337	0.167	0.417
MATNet [89]	✓	✓	✓	**64.2**	**0.712**	**0.670**	**0.599**	**0.492**	**0.246**	**0.544**
CIS	✗	✓	✓	49.4	0.556	0.463	0.329	0.176	0.030	0.311
CIS (post-processing)	✗	✓	✓	54.1	0.631	0.542	0.399	0.210	0.033	0.363
Ours	✗	✗	✓	63.4	**0.742**	**0.654**	**0.524**	**0.351**	**0.147**	**0.484**

Table 3. **Comparison results on MoCA dataset.** We report the successful localization rate for various thresholds τ (see Section 4.2). Both CIS and Ours were pre-trained on DAVIS and finetuned on MoCA in a self-supervised manner. Note that, our method achieves comparable Jaccard ($\mathcal{J}$) to MATNet (2nd best model on DAVIS), without using RGB inputs and without any manual annotation for training.

vious supervised approaches *e.g*. COD [41] (18.5% on Jaccard), and even COSNet [48] (among the top supervised approaches on DAVIS). We conjecture that COSNet's weaker performance is due to its sole reliance on visual appearance (which is distracting for the MoCA data) rather than using motion inputs. This is particularly interesting, as it clearly indicates that no single information cue is able to do the task perfectly, echoing the two-stream hypothesis [28] that both appearance and motion are essential to visual systems.

5.4. Limitations

Despite showing remarkable improvements on motion segmentation in accuracy and runtime, we note the following limitations of the proposed approach (shown in Figure 4) and treat them as future work: *first*, the existing benchmarks are mostly limited to motion segmentation into foreground and background, thus, we choose to use two slots in this paper; however, in real scenarios, videos may contain multiple independently moving objects, which the current model will assign to a single layer. It may be desirable to further separate these objects into different layers. *Second*, we have only explored motion-only (optical flow) as input, which significantly limits the model in segmenting objects when flow is uninformative or incomplete (as in Fig. 4, right); however, the self-supervised video object segmentation objective is applicable also to a two-stream approach, and so RGB could be incorporated. *Third*, the current method may fail when optical flow is noisy or low-quality (Fig. 4, left); jointly optimizing flow and segmentation is a possible way forward in this case.

6. Conclusion

In this paper, we present a self-supervised model for motion segmentation. The algorithm takes only flow as input, and is trained without any manual annotation, surpassing previous self-supervised methods on public benchmarks such as DAVIS2016, narrowing the gap with supervised methods. On the more challenging camouflage dataset (MoCA), our model actually compares favourably to the top approaches in video object segmentation that are trained with heavy supervision. As computation power grows and more high quality videos become available, we believe that self-supervised learning algorithms can serve as a strong competitor to the supervised counterparts for their scalability and generalizability.

References

[1] Jean-Baptiste Alayrac, João Carreira, and Andrew Zisserman. The visual centrifuge: Model-free layered video representations. In *Proc. CVPR*, 2019. 3

[2] Jean-Baptiste Alayrac, Joao Carreira, Relja Arandjelovic, and Andrew Zisserman. Controllable attention for structured layered video decomposition. In *Proc. ICCV*, 2019. 3

[3] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. In *NIPS*, 2017. 2, 4

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 3

[5] Pia Bideau and Erik Learned-Miller. A detailed rubric for motion segmentation. *arXiv preprint arXiv:1610.10033*, 2016. 2

[6] Pia Bideau and Erik Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In *Proc. ECCV*, 2016. 2, 5

[7] Pia Bideau, Rakesh R Menon, and Erik Learned-Miller. Moa-net: self-supervised motion segmentation. In *Proc. ECCV Workshop*, 2018. 2

[8] Gabriel J Brostow and Irfan A Essa. Motion based decompositing of video. In *Proc. ICCV*, 1999. 3

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. 2

[10] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Proc. ECCV*, 2010. 2

[11] Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. 3

[12] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proc. CVPR*, 2017. 2

[13] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. ECCV*, 2020. 3

[14] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proceedings of the International Conference on Machine Learning*, 2020. 3

[15] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Proc. ICCV*, 2017. 7

[16] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In *Proc. ICCV Workshop*, 2019. 2

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 7

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[19] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Proc. NeurIPS*, 2019. 2

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021. 3

[21] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, , and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *Proc. ICLR*, 2020. 3

[22] Mark Everingham, Luc Van Gool, Chris K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 6

[23] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *Proc. BMVC*, 2014. 7

[24] Deng-Ping Fan, Wenguan Wang, Ming-Ming Cheng, and Jianbing Shen. Shifting more attention to video salient object detection. In *Proc. CVPR*, 2019. 2

[25] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *Proc. CVPR*, 2012. 2

[26] Yossi Gandelsman, Assaf Shocher, and Michal Irani. "Double-DIP": Unsupervised image decomposition via coupled deep-image-priors. In *Proc. CVPR*, 2019. 3

[27] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video Action Transformer Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3

[28] Melvyn A. Goodale and A. David Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, 1992. 8

[29] Klaus Greff, Raphael Lopez Kaufman, Rishabh Kabra,

Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proc. ICML*, 2019. 3

[30] Suyog Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2017. 6, 7

[31] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *Proc. CVPR*, 2017. 2

[32] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proc. CVPR*, 2017. 3, 5

[33] Njegica Jojic and B.J. Frey. Learning flexible sprites in video layers. In *Proc. CVPR*, 2001. 3

[34] Yeong Jun Koh and Chang-Su Kim. Primary object segmentation in videos based on region augmentation and reduction. In *Proc. CVPR*, 2017. 2

[35] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *Proc. ICCV*, 2015. 2

[36] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *Proc. ICCV*, 2015. 2, 7

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[38] M Pawan Kumar, Philip HS Torr, and Andrew Zisserman. Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76(3):301–319, 2008. 3

[39] Zihang Lai, Erika Lu, and Weidi Xie. MAST: A memory-augmented self-supervised tracker. In *Proc. CVPR*, 2020. 2

[40] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. In *BMVC*, 2019. 2

[41] Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. Betrayed by motion: Camouflaged object discovery via motion segmentation. *Proc. ACCV*, 2020. 1, 2, 5, 6, 8

[42] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *Proc. ICCV*, 2013. 2, 6

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 2014. 6

[44] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *Proc. CVPR*, 2020. 2, 6

[45] Pengpeng Liu, Michael R. Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *Proc. CVPR*, 2019. 2

[46] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Proc. NeurIPS*, 2020. 2, 3, 5

[47] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. In *SIGGRAPH Asia*, 2020. 3

[48] Xiankai Lu, Wenguan Wang, Chao Ma, Jianbing Shen, Ling Shao, and Fatih Porikli. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *Proc. CVPR*, 2019. 7, 8

[49] Sabarinath Mahadevan, Ali Athar, Aljoša Ošep, Sebastian Hennen, Laura Leal-Taixé, and Bastian Leibe. Making a case for 3d convolutions for object segmentation in videos. In *Proc. BMVC*, 2020. 7

[50] Aravindh Mahendran, James Thewlis, and Andrea Vedaldi. Cross pixel optical-flow similarity for self-supervised learning. In *Proceedings of the Asian Conference on Computer Vision*, 2018. 2

[51] K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. 2

[52] Peter Ochs and Thomas Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *Proc. ICCV*, 2011. 2

[53] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proc. ICCV*, 2019. 2

[54] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *Proc. ICCV*, 2013. 2, 7

[55] 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 *Proc. CVPR*, 2016. 2, 5

[56] Ochs Peter, Malik Jitendra, and Brox Thomas. Segmentation of moving objects by long term video analysis. *TPAMI*, 2014. 2, 6

[57] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 2

[58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 3

[59] VS Ramachandran. Guest editorial: The neurobiology of perception. *Perception*, 14(97):103, 1985. 1

[60] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 3

[61] Anurag Ranjan, Varun Jampani, Lukas Balles, Deqing Sun, Kihwan Kim, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proc. CVPR*, 2019. 2

[62] Hedvig Sidenbladh, Michael J Black, and David J Fleet.

Stochastic tracking of 3d human figures using 2d image motion. In *Proc. ECCV*, 2000. 2

[63] Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. Object level grouping for video shots. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, 2004. 2

[64] Hongmei Song, Wenguan Wang, Sanyuan Zhao, Jianbing Shen, and Kin-Man Lam. Pyramid dilated deeper convlstm for video salient object detection. In *Proc. ECCV*, 2018. 7

[65] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proc. CVPR*, 2019. 3

[66] D. Sun, E. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. In *Proc. CVPR*, 2012. 3

[67] Deqing Sun, Jonas Wulff, Erik Sudderth, Hanspeter Pfister, and Michael Black. A fully-connected layered model of foreground and background flow. In *Proc. CVPR*, 2013. 3

[68] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proc. CVPR*, 2018. 2, 3, 6

[69] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, 2020. 2, 6

[70] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning motion patterns in videos. In *Proc. CVPR*, 2017. 2

[71] Pavel Tokmakov, Cordelia Schmid, and Karteek Alahari. Learning to segment moving objects. *International Journal of Computer Vision*, 2019. 2, 7

[72] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 5

[73] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proc. CVPR*, 2019. 2

[74] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. 2017. 2

[75] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proc. ECCV*, 2018. 2

[76] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *The IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625–638, September 1994. 3

[77] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J Crandall, and Ling Shao. Zero-shot video object segmentation via attentive graph neural networks. In *Proc. ICCV*, 2019. 2

[78] Wenguan Wang, Jianbing Shen, Ruigang Yang, and Fatih Porikli. Saliency-aware video object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(1):20–33, 2018. 7

[79] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. CVPR*, 2018. 3

[80] Max Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische forschung*, 4(1):301–350, 1923. 1

[81] Jonas Wulff and Michael J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Proc. ECCV*. 3

[82] Jonas Wulff and Michael J. Black. Modeling blurred video with layers. In *Proc. ICCV*, 2014. 3

[83] Christopher Xie, Yu Xiang, Zaid Harchaoui, and Dieter Fox. Object discovery in videos as foreground motion clustering. In *Proc. CVPR*, 2019. 2

[84] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. In *Proc. ECCV*, 2018. 2

[85] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T. Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 34(4), 2015. 3

[86] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *Proc. CVPR*. 2, 6, 7

[87] Zhao Yang, Qiang Wang, Luca Bertinetto, Song Bai, Weiming Hu, and Philip H.S. Torr. Anchor diffusion for unsupervised video object segmentation. In *Proc. ICCV*, 2019. 2

[88] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 3

[89] Tianfei Zhou, Shunzhou Wang, Yi Zhou, Yazhou Yao, Jianwu Li, and Ling Shao. Motion-attentive transition for zero-shot video object segmentation. In *AAAI*, volume 34, pages 13066–13073, 2020. 7, 8

[90] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM transactions on graphics (TOG)*, 23(3):600–608, 2004. 3

A. Training Details

In this section, we outline the training settings and architecture details used in our paper. Codes and models will be released.

A.1. Encoder & Decoder

The backbone of network architecture for the model are shown in Table 4, we refer the readers to the pseudocode for iterative binding module.

	stage	operation	output sizes
	input	–	$3 \times 128 \times 224$
Encoder	conv1	$[5 \times 5, 64] \times 2$	$64 \times 128 \times 224$
	mp1	maxpool, stride = 2	$64 \times 64 \times 112$
	conv2	$[5 \times 5, 128] \times 2$	$128 \times 64 \times 112$
	mp2	maxpool, stride = 2	$128 \times 32 \times 56$
	conv3	$[5 \times 5, 256] \times 2$	$256 \times 32 \times 56$
Decoder	conv^{T}1	$5 \times 5, 64$, stride $= 2$	$64 \times 2 \times 32 \times 56$
	conv^{T}2	$5 \times 5, 64$, stride $= 2$	$64 \times 2 \times 64 \times 112$
	conv^{T}3	$5 \times 5, 64$, stride $= 2$	$64 \times 2 \times 128 \times 224$
	outconv	$5 \times 5, 64$, $5 \times 5, 4$	$4 \times 2 \times 128 \times 224$

Table 4. Network architecture. All convolutions have padding 2 to preserve spatial resolution, and are followed by instance normalization and ReLU activation, except the final layer. The details of the iterative binding module is in Figure 5.

A.2. Iterative Binding Module

The pseudocode for the iterative binding module is shown in Figure 5. The full code is available in the submitted source code, and will be made publicly available.

```
Algorithm: Pseudo Code for Iterative Binding / Grouping

# inputs: feature maps + position encoding
slots = Embedding(D, 2)    # [bsz, 2, D]
K = project_K(inputs)      # [bsz, HW, D]
V = project_V(inputs)      # [bsz, HW, D]

for _ in range(T):         # T iterations
    slots = LayerNorm(slots) # [bsz, 2, D]
    Q = project_Q(slots)     # [bsz, 2, D]
    scores = dot(K, Q.t())   # [bsz, HW, 2]
    attn = softmax(scores / sqrt(D), axis=-1) # [bsz, HW, 2]
    updates = weighted_mean(attn.t(), V)      # [bsz, 2, D]
    slots = GRU(slots, updates)               # [bsz, 2, D]
    slots = slots + MLP(LayerNorm(slots))     # [bsz, 2, D]
```

Figure 5. Pseudocode for iterative binding. The linear projections for key (K) and value (V) have 256 dimensions. The MLP has two layers, both with 256 dimensions, with ReLU in between.

A.3. Hyperparameters

For all datasets, we train with batch size of 64, although note that this corresponds to 32 *pairs* of optical flow in order to train with consistency loss. Our initial learning rate is 5e-4, with first 200 steps being warmup, and decays by half every 8e4 iterations. During this decay, the scale for entropy and consistency loss is also increased by a factor of 5, gradually encouraging the predicted alpha channel to be binary. We train the algorithm for about 300k iterations.

B. MoCA dataset curation

The MoCA dataset contains 141 high-definition video sequences, with an average duration of 11 seconds. These sequences were collected from YouTube with resolution 720×1280, and sampled at 24 fps, resulting in 37K frames depicting 67 kinds of camouflaged animals moving in natural scenes. Both temporal and spatial annotations are provided in the form of tight bounding boxes on every 5th frame. We use a modified version of this dataset in order to make it more suitable for segmentation tasks. We outline the process and rationale below.

- We crop away the channel logos and empty border spaces, and resize the low-resolution images to the same resolution as the other images in the dataset (at 720×1280). We then adjust the ground-truth annotations accordingly.
- The original authors resampled all videos to 24 fps even when some original videos have less, causing some consecutive frames to be identical. To alleviate this, we sample every 3 frames from the original dataset, up to 100 frames per video.
- For annotations, we use linear interpolation to generate the missing frames' bounding boxes, resulting in a dense frame-wise annotation.
- The authors also provided the motion labels for each annotated frame (locomotion, deformation, static), so we filter away videos with predominantly no locomotion.
- We also further discard videos that contain large amount of frames where the motion does not belong to the primary object.

This eventually results in 88 video sequences and 4803 frames, which we will release for fair comparison.

C. Results breakdown

The main evaluation metric used in this paper is the Jaccard score ($\mathcal{J}$), which is the intersection-over-union between the predicted and ground-truth masks. In line with previous works, we show the per-category results breakdown for our model on DAVIS2016, SegTrackv2, FBMS59, and MoCA in Tables 5-8. Note that, since we focus on both speed and accuracy, the predictions are only of 128×224 pixels, and we directly upsample this prediction to the original resolution and compare with the groundtruth.

Sequence	$\mathcal{J}$(M)	$\mathcal{J}$(R)	$\mathcal{J}$(D)	$\mathcal{F}$(M)	$\mathcal{F}$(R)	$\mathcal{F}$(D)
bear	0.766	1.000	-4.5	0.640	0.914	-1.9
blackswan	0.795	1.000	5.9	0.658	0.980	7.3
bmx-bumps	0.373	0.114	8.3	0.260	0.011	10.8
bmx-trees	0.744	1.000	14.5	0.666	0.909	13.9
boat	0.602	0.787	44.5	0.724	0.936	22.4
breakdance	0.792	0.990	-2.4	0.814	1.000	10.8
breakdance-flare	0.826	1.000	10.2	0.771	1.000	16.7
bus	0.391	0.000	-4.4	0.248	0.000	-5.0
camel	0.540	0.679	9.3	0.667	0.923	-9.8
car-roundabout	0.479	0.500	1.2	0.404	0.146	-10.6
car-shadow	0.879	1.000	-0.2	0.818	1.000	-3.3
car-turn	0.492	0.537	3.4	0.560	0.683	6.8
cows	0.690	0.898	-5.0	0.653	0.852	4.4
dance-jump	0.815	1.000	-4.8	0.683	1.000	5.9
dance-twirl	0.739	0.960	7.4	0.803	0.900	28.5
dog	0.701	0.789	2.9	0.477	0.526	9.3
dog-agility	0.810	1.000	7.7	0.858	1.000	4.0
drift-chicane	0.777	1.000	7.3	0.611	0.646	26.3
drift-straight	0.864	1.000	-5.2	0.685	1.000	-6.8
drift-turn	0.593	0.654	27.7	0.220	0.000	20.4
Average	0.683	0.795	6.2	0.611	0.721	7.5

Table 5. Full results on DAVIS2016. J refers to the Jaccard score, while the F-measure refers to the contour accuracy. M, R, and D refers to mean, recall and decay respectively.

Sequence	$\mathcal{J}$(M)
bird of paradise	0.791
birdfall	0.300
bmx	0.609
cheetah	0.370
drift	0.797
frog	0.733
girl	0.746
hummingbird	0.506
monkey	0.751
monkeydog	0.133
parachute	0.914
penguin	0.697
soldier	0.741
worm	0.326
seq avg	0.601
frames avg	0.586

Table 6. Sequence-wise results on SegTrackv2.

Sequence	$\mathcal{J}$(M)
camel01	0.281
cars1	0.846
cars10	0.322
cars4	0.826
cars5	0.842
cats01	0.672
cats03	0.640
cats06	0.362
dogs01	0.629
dogs02	0.636
farm01	0.816
giraffes01	0.322
goats01	0.375
horses02	0.628
horses04	0.566
horses05	0.334
lion01	0.399
marple12	0.680
marple2	0.750
marple4	0.799
marple6	0.450
marple7	0.567
marple9	0.537
people03	0.598
people1	0.761
people2	0.842
rabbits02	0.415
rabbits03	0.319
rabbits04	0.400
tennis	0.561
seq avg	0.573
frames avg	0.531

Table 7. Sequence-wise results on FBMS59.

D. Qualitative results

We show more qualitative results in Figures 6 and 7. We also submit video sequences of the results as part of the Supplementary Material.

sequence	$\mathcal{J}$	$\tau_{0.5}$	$\tau_{0.6}$	$\tau_{0.7}$	$\tau_{0.8}$	$\tau_{0.9}$	avg
arabian_horn_viper	0.709	0.990	0.909	0.586	0.091	0.000	0.515
arctic_fox	0.381	0.404	0.383	0.298	0.191	0.000	0.255
arctic_fox_1	0.879	0.913	0.913	0.913	0.913	0.652	0.861
arctic_wolf_0	0.705	0.929	0.859	0.596	0.202	0.051	0.527
arctic_wolf_1	0.712	0.795	0.795	0.795	0.744	0.256	0.677
bear	0.508	0.611	0.453	0.221	0.074	0.000	0.272
black_cat_0	0.499	0.556	0.476	0.302	0.048	0.000	0.276
black_cat_1	0.086	0.030	0.020	0.010	0.000	0.000	0.012
crab	0.594	0.800	0.400	0.200	0.000	0.000	0.280
crab_1	0.288	0.309	0.200	0.073	0.000	0.000	0.116
cuttlefish_0	0.222	0.194	0.032	0.000	0.000	0.000	0.045
cuttlefish_1	0.034	0.043	0.000	0.000	0.000	0.000	0.009
cuttlefish_4	0.655	1.000	0.846	0.231	0.000	0.000	0.415
cuttlefish_5	0.724	0.870	0.870	0.783	0.304	0.043	0.574
dead_leaf_butterfly_1	0.784	0.913	0.783	0.739	0.696	0.304	0.687
desert_fox	0.470	0.362	0.234	0.191	0.149	0.106	0.209
devil_scorpionfish	0.938	1.000	1.000	1.000	0.913	0.826	0.948
devil_scorpionfish_1	0.913	1.000	1.000	1.000	1.000	0.565	0.913
devil_scorpionfish_2	0.857	0.968	0.903	0.871	0.806	0.548	0.819
egyptian_nightjar	0.765	0.905	0.842	0.789	0.611	0.158	0.661
elephant	0.728	0.783	0.652	0.609	0.565	0.348	0.591
flatfish_0	0.575	0.677	0.657	0.636	0.485	0.141	0.519
flatfish_1	0.682	0.848	0.835	0.722	0.354	0.051	0.562
flatfish_2	0.765	0.839	0.774	0.774	0.742	0.645	0.755
flatfish_4	0.697	0.958	0.895	0.568	0.126	0.000	0.509
flounder	0.896	1.000	1.000	0.986	0.986	0.437	0.882
flounder_3	0.505	0.429	0.286	0.143	0.143	0.000	0.200
flounder_4	0.767	0.949	0.897	0.769	0.487	0.128	0.646
flounder_5	0.681	0.797	0.747	0.696	0.468	0.165	0.575
flounder_6	0.683	0.768	0.677	0.616	0.465	0.263	0.558
flounder_7	0.719	0.930	0.845	0.662	0.254	0.028	0.544
flounder_8	0.707	0.925	0.774	0.645	0.409	0.000	0.551
flounder_9	0.636	0.821	0.718	0.436	0.231	0.026	0.446
fossa	0.280	0.143	0.000	0.000	0.000	0.000	0.029
goat_0	0.589	0.707	0.636	0.414	0.212	0.020	0.398
goat_1	0.744	0.930	0.930	0.704	0.380	0.085	0.606
groundhog	0.525	0.646	0.525	0.374	0.162	0.010	0.343
hedgehog_0	0.329	0.298	0.170	0.085	0.021	0.021	0.119
hedgehog_1	0.471	0.533	0.400	0.333	0.067	0.067	0.280
hedgehog_2	0.771	0.800	0.733	0.733	0.600	0.267	0.627
hedgehog_3	0.486	0.564	0.359	0.282	0.128	0.026	0.272
hermit_crab	0.674	0.806	0.806	0.677	0.419	0.065	0.555
ibex	0.390	0.513	0.359	0.128	0.000	0.000	0.200
jerboa	0.555	0.739	0.435	0.348	0.130	0.000	0.330

sequence	$\mathcal{J}$	$\tau_{0.5}$	$\tau_{0.6}$	$\tau_{0.7}$	$\tau_{0.8}$	$\tau_{0.9}$	avg
jerboa_1	0.450	0.452	0.323	0.226	0.097	0.000	0.219
lichen_katydid	0.502	0.455	0.323	0.162	0.020	0.010	0.194
lion_cub_0	0.728	0.972	0.873	0.549	0.282	0.127	0.561
lion_cub_1	0.571	0.687	0.556	0.354	0.141	0.010	0.349
lion_cub_3	0.179	0.182	0.141	0.081	0.051	0.000	0.091
lioness	0.423	0.419	0.129	0.000	0.000	0.000	0.110
marine_iguana	0.376	0.217	0.130	0.000	0.000	0.000	0.070
markhor	0.808	0.909	0.855	0.800	0.709	0.364	0.727
meerkat	0.778	0.871	0.774	0.742	0.710	0.323	0.684
mountain_goat	0.742	1.000	0.968	0.710	0.226	0.065	0.594
nile_monitor_1	0.524	0.616	0.434	0.283	0.121	0.000	0.291
octopus	0.637	0.838	0.687	0.273	0.131	0.020	0.390
octopus_1	0.411	0.242	0.091	0.061	0.061	0.010	0.093
peacock_flounder_0	0.884	0.931	0.931	0.931	0.931	0.701	0.885
peacock_flounder_1	0.812	0.970	0.929	0.859	0.667	0.222	0.729
peacock_flounder_2	0.873	1.000	1.000	0.989	0.926	0.368	0.857
polar_bear_0	0.622	0.845	0.676	0.352	0.141	0.000	0.403
polar_bear_1	0.487	0.603	0.556	0.476	0.175	0.016	0.365
polar_bear_2	0.792	0.949	0.846	0.744	0.641	0.308	0.697
pygmy_seahorse_2	0.478	0.582	0.364	0.255	0.036	0.000	0.247
pygmy_seahorse_4	0.654	0.935	0.839	0.516	0.000	0.000	0.458
rodent_x	0.738	0.870	0.826	0.696	0.435	0.174	0.600
scorpionfish_0	0.622	0.761	0.648	0.521	0.408	0.127	0.493
scorpionfish_1	0.616	0.681	0.574	0.426	0.319	0.128	0.426
scorpionfish_2	0.842	0.962	0.949	0.924	0.823	0.380	0.808
scorpionfish_3	0.804	0.975	0.861	0.785	0.595	0.354	0.714
scorpionfish_4	0.800	1.000	0.949	0.821	0.513	0.231	0.703
scorpionfish_5	0.899	1.000	1.000	1.000	0.957	0.478	0.887
seal_1	0.865	1.000	0.913	0.870	0.870	0.652	0.861
seal_2	0.676	0.800	0.655	0.455	0.291	0.145	0.469
seal_3	0.445	0.400	0.200	0.067	0.000	0.000	0.133
shrimp	0.772	0.933	0.867	0.733	0.667	0.133	0.667
snow_leopard_0	0.760	1.000	0.949	0.692	0.359	0.077	0.615
snow_leopard_1	0.772	0.826	0.696	0.652	0.652	0.522	0.670
snow_leopard_2	0.883	1.000	0.989	0.968	0.863	0.526	0.869
snow_leopard_3	0.608	0.778	0.556	0.417	0.333	0.083	0.433
snow_leopard_6	0.816	0.830	0.787	0.787	0.723	0.660	0.757
snow_leopard_7	0.679	0.871	0.806	0.581	0.258	0.000	0.503
snow_leopard_8	0.556	0.702	0.596	0.447	0.191	0.000	0.387
snowy_owl_0	0.564	0.532	0.340	0.298	0.128	0.000	0.260
spider_tailed_horned_viper_0	0.473	0.400	0.333	0.267	0.067	0.067	0.227
spider_tailed_horned_viper_1	0.558	0.620	0.423	0.310	0.254	0.056	0.332
spider_tailed_horned_viper_2	0.848	0.964	0.945	0.909	0.782	0.564	0.833
spider_tailed_horned_viper_3	0.860	1.000	1.000	1.000	0.677	0.387	0.813
seq avg	0.634	0.734	0.640	0.522	0.361	0.166	0.485
frames avg	0.634	0.742	0.654	0.524	0.351	0.147	0.484

Table 8. Results breakdown for MoCA.

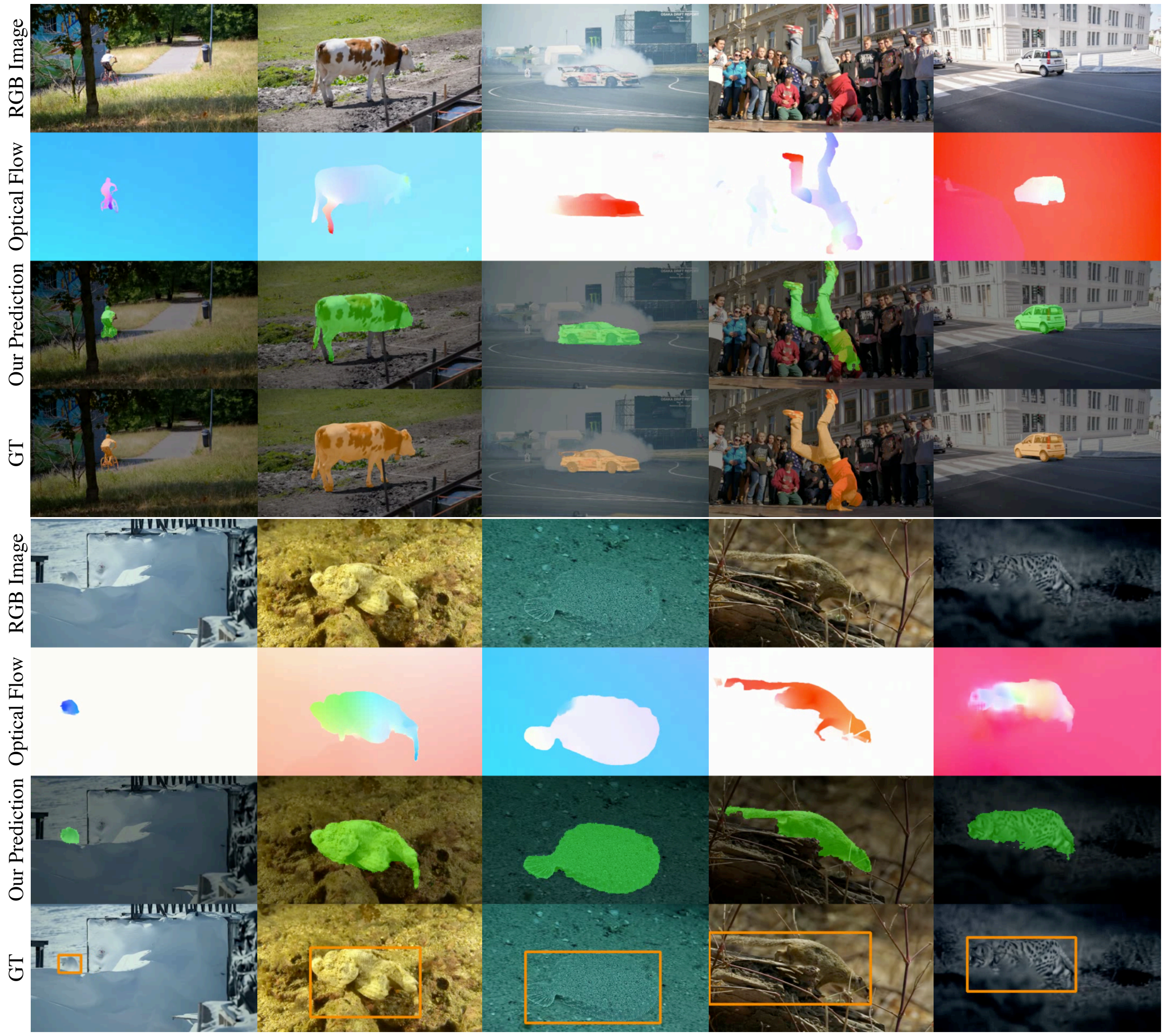

Figure 6. Qualitative results on DAVIS2016 and MoCA, respectively.

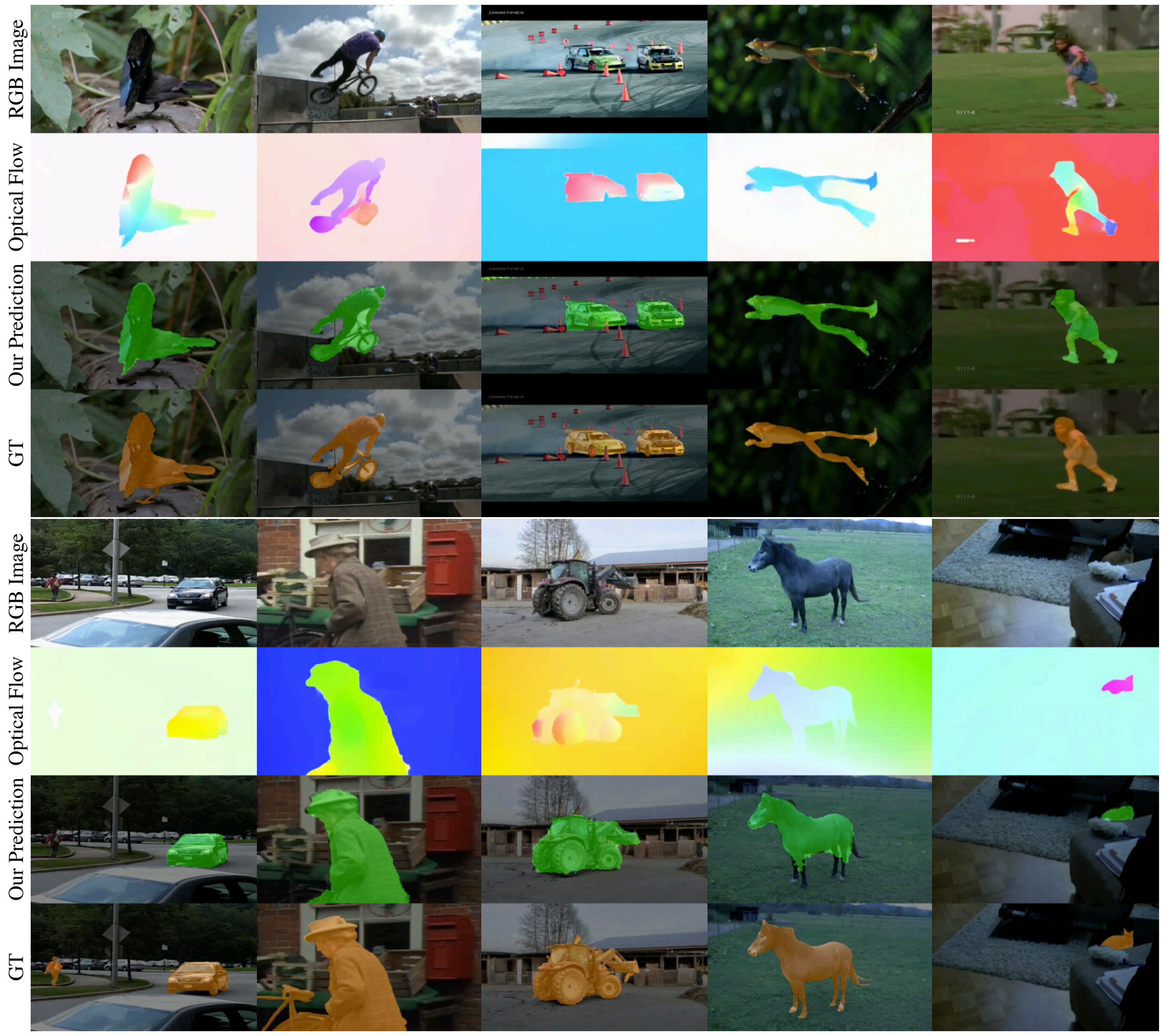

Figure 7. Qualitative results on SegTrackv2 and FBMS59, respectively.