Object Discovery in Videos as Foreground Motion Clustering

Christopher Xie[1*] Yu Xiang[2] Zaid Harchaoui[1] Dieter Fox[2,1]

[1]University of Washington   [2]NVIDIA

chrisxie@cs.washington.edu   {yux,dieterf}@nvidia.com   zaid@uw.edu

Abstract

We consider the problem of providing dense segmentation masks for object discovery in videos. We formulate the object discovery problem as foreground motion clustering, where the goal is to cluster foreground pixels in videos into different objects. We introduce a novel pixel-trajectory recurrent neural network that learns feature embeddings of foreground pixel trajectories linked across time. By clustering the pixel trajectories using the learned feature embeddings, our method establishes correspondences between foreground object masks across video frames. To demonstrate the effectiveness of our framework for object discovery, we conduct experiments on commonly used datasets for motion segmentation, where we achieve state-of-the-art performance.

1. Introduction

Discovering objects from videos is an important capability that an intelligent system needs to have. Imagine deploying a robot to a new environment. If the robot can discover and recognize unknown objects in the environment by observing, it would enable the robot to better understand its work space. In the interactive perception setting [8], the robot can even interact with the environment to discover objects by touching or pushing objects. To tackle the object discovery problem, we need to answer the question: what defines an object? In this work, we consider an entity that can move or be moved to be an object, which includes various rigid, deformable and articulated objects. We utilize motion and appearance cues to discover objects in videos.

Motion-based video understanding has been studied in computer vision for decades. In low-level vision, different methods have been proposed to find correspondences between pixels across video frames, which is known as optical flow estimation [18, 3]. Both camera motion and object motion can result in optical flow. Since the correspondences are estimated at a pixel level, these methods are not aware of the objects in the scene, in the sense that they do not know which pixels belong to which objects. In high-level vision,

---

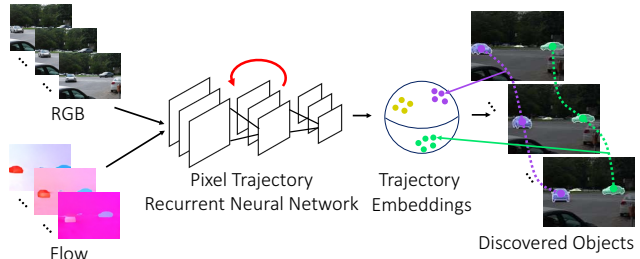*Work partially done while an intern at NVIDIA.

Figure 1: Overview of our framework. RGB images and optical flow are fed into a recurrent neural network, which computes embeddings of pixel trajectories. These embeddings are clustered into different foreground objects.

object detection and object tracking in videos has been well-studied [1, 22, 16, 50, 4, 48]. These methods train models for specific object categories using annotated data. As a result, they are not able to detect nor track unknown objects that have not been seen in the training data. In other words, these methods cannot discover new objects from videos. In contrast, motion segmentation methods [9, 24, 5, 34] aim at segmenting moving objects in videos, which can be utilized to discover new objects based on their motion.

In this work, we formulate the object discovery problem as foreground motion clustering, where the goal is to cluster pixels in a video into different objects based on their motion. There are two main challenges in tackling this problem. First, how can foreground objects be differentiated from background? Based on the assumption that moving foreground objects have different motion as the background, we design a novel encoder-decoder network that takes video frames and optical flow as inputs and learns a feature embedding for each pixel, where these feature embeddings are used in the network to classify pixels into foreground or background. Compared to traditional foreground/background segmentation methods [10, 19], our network automatically learns a powerful feature representation that combines appearance and motion cues from images.

Secondly, how can we consistently segment foreground objects across video frames? We would like to segment individual objects in each video frame and establish correspondences of the same object across video frames. Inspired

by [9] that clusters pixel trajectories across video frames for object segmentation, we propose to learn feature embeddings of pixel trajectories with a novel Recurrent Neural Network (RNN), and then cluster these pixel trajectories with the learned feature embeddings. Since the pixel trajectories are linked in time, our method automatically establishes the object correspondences across video frames by clustering the trajectories. Different from [9] that employs hand-crafted features to cluster pixel trajectories, our method automatically learns a feature representation of the trajectories, where the RNN controls how to combine pixel features along a trajectory to obtain the trajectory features. Figure 1 illustrates our framework for object motion clustering.

Since our problem formulation aims to discover objects based on motion, we conduct experiments on five motion segmentation datasets to evaluate our method: Flying Things 3D [29], DAVIS [35, 37], Freiburg-Berkeley motion segmentation [32], ComplexBackground [30] and CamouflagedAnimal [6]. We show that our method is able to segment potentially unseen foreground objects in the test videos and consistently across video frames. Comparison with the state-of-the-art motion segmentation methods demonstrates the effectiveness of our learned trajectory embeddings for object discovery. In summary, our work has the following key contributions:

- We introduce a novel encoder-decoder network to learn feature embeddings of pixels in videos that combines appearance and motion cues.

- We introduce a novel recurrent neural network to learn feature embeddings of pixel trajectories in videos.

- We use foreground masks as an attention mechanism to focus on clustering of relevant pixel trajectories for object discovery.

- We achieve state-of-the-art performance on commonly used motion segmentation datasets.

This paper is organized as follows. After discussing related work, we introduce our foreground motion clustering method designed for object discovery, followed by experimental results and a conclusion.

2. Related Work

Video Foreground Segmentation. Video foreground segmentation is the task of classifying every pixel in a video as foreground or background. This has been well-studied in the context of video object segmentation [6, 33, 44, 44, 21], especially with the introduction of unsupervised challenge of the DAVIS dataset [35]. [6] uses a probabilistic model that acts upon optical flow to estimate moving objects. [33] predicts video foreground by iteratively refining motion boundaries while encouraging spatio-temporal smoothness. [44, 45, 21]

adopt a learning-based approach and train Convolutional Neural Networks (CNN) that utilize RGB and optical flow as inputs to produce foreground segmentations. Our approach builds on these ideas and uses the foreground segmentation as an attention mechanism for pixel trajectory clustering.

Instance Segmentation. Instance segmentation algorithms segment individual object instances in images. Many instance segmentation approaches have adopted the general idea of combining segmentation with object proposals [17, 36]. While these approaches only work for objects that have been seen in a training set, we make no such assumption as our intent is to discover objects. Recently, a few works have investigated the instance segmentation problem as a pixel-wise labeling problem by learning pixel embeddings [11, 31, 27, 13]. [31] predicts pixel-wise features using translation-variant semi-convolutional operators. [13] learns pixel embeddings with seediness scores that are used to compose instance masks. [11] designs a contrastive loss and [27] unrolls mean shift clustering as a neural network to learn pixel embeddings. We leverage these ideas to design our approach of learning embeddings of pixel trajectories.

Motion Segmentation. Pixel trajectories for motion analysis were first introduced by [42]. [9] used them in a spectral clustering method to produce motion segments. [32] provided a variational minimization to produce pixel-wise motion segmentations from trajectories. Other works that build off this idea include formulating trajectory clustering as a multi-cut problem [23, 24, 25] or as a density peaks clustering [46], and detecting discontinuities in the trajectory spectral embedding [15]. More recent approaches include using occlusion relations to produce layered segmentations [43], combining piecewise rigid motions with pre-trained CNNs to merge the rigid motions into objects [7], and jointly estimating scene flow and motion segmentations [39]. We use pixel trajectories in a recurrent neural network to learn trajectory embeddings for motion clustering.

3. Method

Our approach takes video frames and optical flow between pairs of frames as inputs, which are fed through an encoder-decoder network, resulting in pixel-wise features. These features are used to predict foreground masks of moving objects. In addition, a recurrent neural network is designed to learn feature embeddings of pixel trajectories inside the foreground masks. Lastly, the trajectory embeddings are clustered into different objects, giving a consistent segmentation mask for each discovered object. The network architecture is visualized in Figure 2.

3.1. Encoder-Decoder: Y-Net

Let $I_t \in \mathbb{R}^{H \times W \times 3}, F_t \in \mathbb{R}^{H \times W \times 2}$ be an RGB image and forward optical flow image at time t, respectively. Our
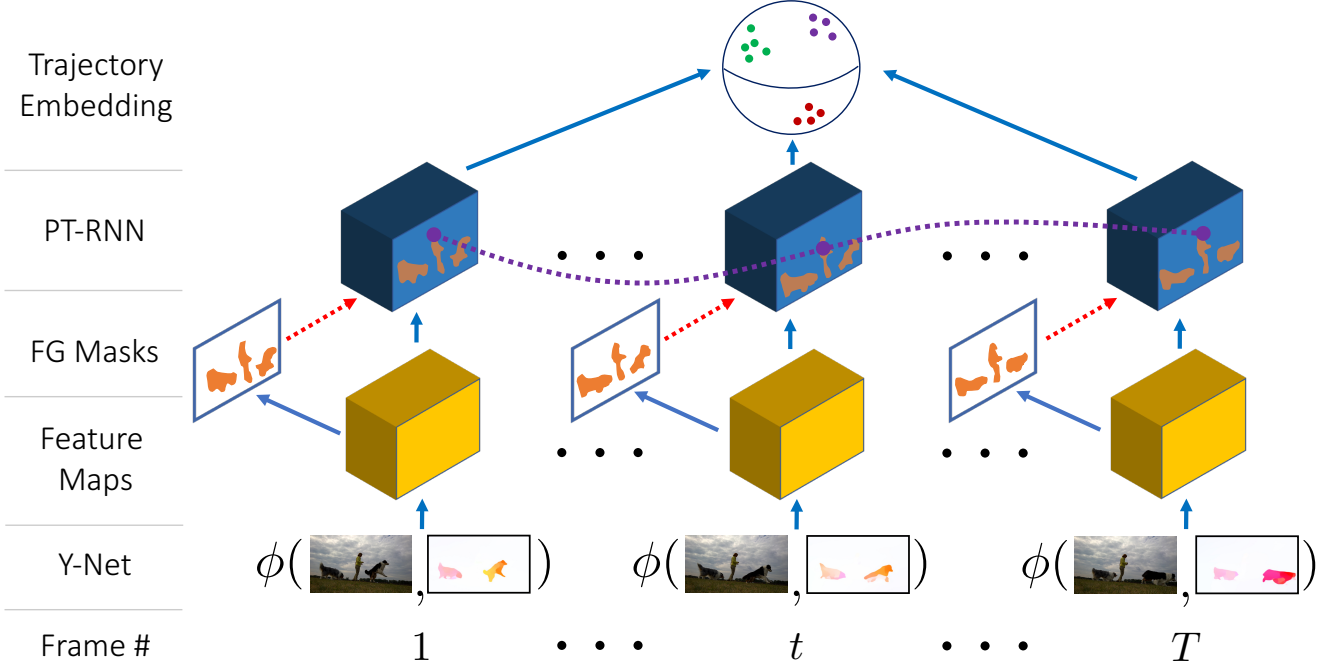
Figure 2: Overall architecture. First, feature maps of each frame are extracted from the Y-Net. Next, foreground masks are computed, shown in orange. The PT-RNN uses these foreground masks to compute trajectory embeddings (example foreground trajectory from frame 1 to T shown in purple), which are normalized to produce unit vectors. Backpropagation passes through the blue solid arrows, but not through the red dashed arrows.

network receives these images from a video as inputs and feeds them into an encoder-decoder network separately at each time step, where the encoder-decoder network extracts dense features for each video frame. Our encoder-decoder network is an extension of the U-Net architecture [38] (Figure 3a) to two different input types, i.e., RGB images and optical flow images, by adding an extra input branch. We denote this mid-level fusion of low-resolution features as Y-Net. We illustrate the Y-Net architecture in Figure 3b.

In detail, our network has two parallel encoder branches for the RGB and optical flow inputs. Each encoder branch consists of four blocks of two 3×3 convolutions (each of which is succeeded by a GroupNorm layer [47] and ReLU activation) followed by a 2×2 max pooling layer. The encodings of the RGB and optical flow branches are then concatenated and input to a decoder network, which consists of a similar architecture to [38] with skip connections from both encoder branches to the decoder.

We argue that this mid-level fusion performs better than early fusion and late fusion (using completely separate branches for RGB and optical flow, similar to two-stream networks [40, 14, 45]) of encoder-decoder networks while utilizing less parameters, and show this empirically in Section 4.1. The output of Y-Net, $\phi(I_t, F_t) \in \mathbb{R}^{H \times W \times C}$, is a pixel-dense feature representation of the scene. We will refer to this as pixel embeddings of the video.

3.2. Foreground Prediction

The Y-Net extracts a dense feature map for each video frame that combines appearance and motion information of the objects. Using these features, our network predicts a foreground mask for each video frame by simply applying another convolution on top of the Y-Net outputs to compute foreground logits. These logits are passed through a sigmoid layer and thresholded at 0.5. For the rest of the paper, we will denote m_t to be the binary foreground mask at time t.

The foreground masks are used as an attention mechanism to focus on the clustering of the trajectory embeddings. This results in more stable performance, as seen in Section 4.1. Note that while we focus on moving objects in our work, the foreground can be specified depending on the problem. For example, if we specify that certain objects such as cars should be foreground, then we would learn a network that learns to discover and segment car instances in videos.

3.3. Trajectory Embeddings

In order to consistently discover and segment objects across video frames, we propose to learn deep representations of *foreground pixel trajectories* of the video. Specifically, we consider dense pixel trajectories throughout the videos, where trajectories are defined as in [42, 9]. Given the outputs of Y-Net, we compute the trajectory embedding as a weighted sum of the pixel embeddings along the trajectory.
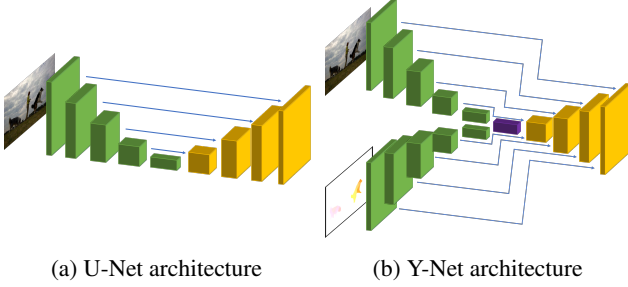
(a) U-Net architecture (b) Y-Net architecture

Figure 3: We show U-Net [38] and our proposed Y-Net to visually demonstrate the difference. Y-Net has two encoding branches (shown in green) for each input modality, which is fused (shown in purple) and passed to the decoder (shown in yellow). Skip connections are visualized as blue arrows.

3.3.1 Linking Foreground Trajectories

We first describe the method to calculate pixel trajectories according to [42]. Denote $F_{t-1} \in \mathbb{R}^{H \times W \times 2}$ to be the forward optical flow field at time $t-1$ and $\hat{F}_t \in \mathbb{R}^{H \times W \times 2}$ to be the backward optical flow field at time t. As defined in [42], we say the optical flow for two pixels (i, j) at time $t-1$ and (i', j') at time t is consistent if

$$\left\| F_{t-1}^{i,j} + \hat{F}_t^{i',j'} \right\|^2 \leq 0.01 \left(\left\| F_{t-1}^{i,j} \right\|^2 + \left\| \hat{F}_t^{i',j'} \right\|^2 \right) + 0.5, \tag{1}$$

where $F_{t-1}^{i,j}$ denotes the i, j-th element of F_{t-1}. Essentially, this condition requires that the backward flow points in the inverse direction of the forward flow, up to a tolerance interval that is linear in the magnitude of the flow. Pixels (i, j) and (i', j') are linked in a pixel trajectory if Eq. (1) holds.

To define *foreground pixel trajectories*, we augment the above definition and say pixels (i, j) and (i', j') are linked if Eq. (1) holds and both pixels are classified as foreground. Using this, we define a foreground-consistent warping function $g : \mathbb{R}^{H \times W} \to \mathbb{R}^{H \times W}$ that warps a set of pixels $\mathbf{v} \in \mathbb{R}^{H \times W}$ forward in time along their foreground trajectories:

$$g(\mathbf{v})^{i',j'} = \begin{cases} \mathbf{v}^{i,j} & \text{if } (i, j), (i', j') \text{ linked} \\ 0 & \text{otherwise.} \end{cases}$$

This can be achieved by warping $\mathbf{v}$ with $\hat{F}_t$ with bilinear interpolation and multiplying by a binary consistency mask. This mask can be obtained by warping the foreground mask m_{t-1} with $\hat{F}_t$ using Eq. (1) and intersecting it with m_t, resulting in a mask that is 1 if (i', j') is linked to a foreground pixel at time $t-1$. Figure 4 demonstrates the linking of pixels in a foreground pixel trajectory.

3.3.2 Pixel Trajectory RNN

After linking foreground pixels into trajectories, we describe our proposed Recurrent Neural Network (RNN) to learn fea-
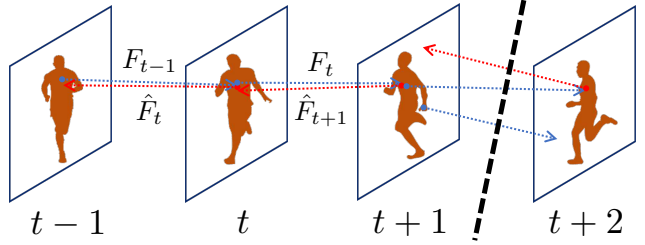

Figure 4: We illustrate pixel linking in foreground pixel trajectories. The foreground mask is shown in orange, forward flow is denoted by the blue dashed arrow, and backward flow is denoted by the red dashed arrow. The figure shows a trajectory that links pixels in frames $t - 1, t, t + 1$. Two failure cases that can cause a trajectory to end are shown between frames $t + 1$ and $t + 2$: 1) Eq. (1) is not satisfied, and 2) one of the pixels is not classified as foreground.

ture embeddings of these trajectories. Denote $\{(i_t, j_t)\}_{t=1}^{L}$ to be the pixel locations of a foreground trajectory, $\{\mathbf{x}_t^{i_t,j_t} \in \mathbb{R}^C\}_{t=1}^{L}$ to be the pixel embeddings of the foreground trajectory (Y-Net outputs, i.e. $\mathbf{x}_t = \phi(I_t, F_t)$), and L as the length of the trajectory. We define the foreground trajectory embeddings to be a weighted sum of the pixel embeddings along the foreground trajectory. Specifically, we have

$$\psi \left(\{\mathbf{x}_t^{i_t,j_t}\}_{t=1}^{L} \right) = \frac{\sum_{t=1}^{L} \mathbf{w}_t^{i_t,j_t} \odot \mathbf{x}_t^{i_t,j_t}}{\sum_{t=1}^{L} \mathbf{w}_t^{i_t,j_t}}, \tag{2}$$

where $\odot$ denotes element-wise multiplication, the division sign denotes element-wise division, and $\mathbf{w}_t^{i_t,j_t} \in [0,1]^C$.

To compute the trajectory embeddings, we encode $\psi(\cdot)$ as a novel RNN architecture which we denote Pixel Trajectory RNN (PT-RNN). In its hidden state, PT-RNN stores

$$\left\{ \mathbf{h}_t^{i_t,j_t} := \sum_{\tau=1}^{t} \mathbf{w}_\tau^{i_\tau,j_\tau} \odot \mathbf{x}_\tau^{i_\tau,j_\tau}, \mathbf{W}_t^{i_t,j_t} := \sum_{\tau=1}^{t} \mathbf{w}_\tau^{i_\tau,j_\tau} \right\}, \tag{3}$$

which allows it to keep track of the running sum and total weight throughout the foreground trajectory. While Eq. (3) describes the hidden state at each pixel location and time step, we can efficiently implement the PT-RNN for all pixels by doing the following: at time step t, PT-RNN first applies the foreground consistent warping function to compute $\widetilde{\mathbf{h}}_{t-1} := g(\mathbf{h}_{t-1})$, $\widetilde{\mathbf{W}}_{t-1} = g(\mathbf{W}_{t-1})$. Next, we compute $\mathbf{w}_t$. We design three variants of PT-RNN to compute $\mathbf{w}_t$, named *standard* (based on simple RNNs), *conv* (based on convRNNs), and *convGRU* (based on [2]). For example, our *conv* architecture is described by

$$\mathbf{c}_t = \text{ReLU} \left(W_c * \begin{bmatrix} \widetilde{\mathbf{h}}_{t-1} & \mathbf{x}_t \\ \widetilde{\mathbf{W}}_{t-1} & \end{bmatrix} \right)$$

$$\mathbf{w}_t = \sigma(W_w * \mathbf{c}_t), \tag{4}$$

where $*$ denotes convolution, and W_c, W_w are 3×3 convolution kernels. After $\mathbf{w}_t$ is computed by the PT-RNN, we update the hidden state with:

$$\begin{aligned}
\mathbf{h}_t &= \widetilde{\mathbf{h}}_{t-1} + \mathbf{w}_t \odot \mathbf{x}_t \\
\mathbf{W}_t &= \widetilde{\mathbf{W}}_{t-1} + \mathbf{w}_t.
\end{aligned} \quad (5)$$

All model variants are described in detail in the supplement. Essentially, *standard* treats each set of linked pixels as a simple RNN, *conv* includes information from neighboring pixels, and *convGRU* allows the network to capture longer term dependencies by utilizing an explicit memory state.

When a trajectory is finished, i.e., pixel (i, j) does not link to any pixel in the next frame, PT-RNN outputs $\mathbf{h}_t^{i,j}/\mathbf{W}_t^{i,j}$, which is equivalent to Eq. (2). This results in a C-dimensional embedding for every foreground pixel trajectory, regardless of its length, when it starts, or when it ends. Note that these trajectory embeddings are pixel-dense, removing the need for a variational minimization step [32]. The embeddings are normalized so that they lie on the unit sphere.

A benefit to labeling the trajectories is that we are enforcing consistency in time, since consistent forward and backward optical flow usually means that the pixels are truely linked [42]. However, issues can arise around the motion and object boundaries, which can lead to trajectories erroneously drifting and representing motion of two different objects or an object and background [42]. In this case, the foreground masks are beneficial and able to sever the trajectory before it drifts. We also note the similarity to the DA-RNN architecture [49] that uses data association in a RNN for semantic labeling.

3.3.3 Spatial Coordinate Module

The foreground trajectory embeddings incorporate information from the RGB and optical flow images. However, they do not encode information about the location of the trajectory in the image. Thus, we introduce a spatial coordinate module which computes location information for each foreground trajectory. Specifically, we compute a 4-dimensional vector consisting of the average x, y pixel location and displacement for each trajectory and pass it through two fully connected (FC) layers to inflate it to a C-dimensional vector, which we add to the output of $\psi(\cdot)$ (before the normalization of the foreground trajectory embeddings).

3.4. Loss Function

To train our proposed network, we use a loss function that is comprised of three terms

$$\mathcal{L} = \lambda_{\text{fg}} \ell_{\text{fg}} + \lambda_{\text{intra}} \ell_{\text{intra}} + \lambda_{\text{inter}} \ell_{\text{inter}} ,$$

where we set $\lambda_{\text{fg}} = \lambda_{\text{intra}} = \lambda_{\text{inter}} = 1$ in our experiments. ℓ_{fg} is a pixel-wise binary cross-entropy loss that is commonly used in foreground prediction. We apply this on the predicted foreground logits. ℓ_{intra} and ℓ_{inter} operate on the foreground trajectory embeddings. Inspired by [11], its goal is to encourage trajectory embeddings of the same object to be close while pushing trajectories that are different objects apart. For simplicity of notation, let us overload notation and define $\{\mathbf{x}_i^k\}, k = 1, \dots, K, i = 1, \dots, N_k$ to be a list of trajectory embeddings of dimension C where k indexes the object and i indexes the embedding. Since all the feature embeddings are normalized to have unit length, we use the cosine distance function $d(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(1 - \mathbf{x}^\mathsf{T}\mathbf{y})$ to measure the distance between two feature embeddings $\mathbf{x}$ and $\mathbf{y}$.

Proposition 1 *Let $\{\mathbf{y}_i\}_{i=1}^N$ be a set of unit vectors such that $\sum_{i=1}^n \mathbf{y}_i \neq 0$. Define the* spherical mean *of this set of unit vectors to be the unit vector that minimizes the cosine distance*

$$\mu := \operatorname*{argmin}_{\|\mathbf{w}\|_2 = 1} \frac{1}{n} \sum_{i=1}^n d(\mathbf{w}, \mathbf{y}_i) \quad (6)$$

Then $\mu = \frac{\sum_{i=1}^n \mathbf{y}_i}{\|\sum_{i=1}^n \mathbf{y}_i\|}$. For the proof, see the supplement.

The goal of the intra-object loss ℓ_{intra} is to encourage these learned trajectory embeddings of an object to be close to their spherical mean. This results in

$$\ell_{\text{intra}} = \frac{1}{K} \sum_{k=1}^K \sum_{i=1}^{N_k} \frac{\mathbb{1}\left\{d(\mu_k, \mathbf{x}_i^k) - \alpha \geq 0\right\} d^2(\mu_k, \mathbf{x}_i^k)}{\sum_{i=1}^{N_k} \mathbb{1}\left\{d(\mu_k, \mathbf{x}_i^k) - \alpha \geq 0\right\}} ,$$

where μ_k is the spherical mean of trajectories $\{\mathbf{x}_i^k\}_{i=1}^{N_k}$ for object k, and $\mathbb{1}$ denotes the indicator function. Note that μ_k is a function of the embeddings. The indicator function acts as a hard negative mining that focuses the loss on embeddings that are further than margin α from the spherical mean. In practice, we do not let the denominator get too small as it could result in unstable gradients, so we allow it to reach a minimum of 50.

Lastly, the inter-object loss ℓ_{inter} is designed to push trajectories of different objects apart. We desire the clusters to be pushed apart by some margin δ, giving

$$\ell_{\text{inter}} = \frac{2}{K(K-1)} \sum_{k<k'} [\delta - d(\mu_k, \mu_{k'})]_+^2 ,$$

where $[x]_+ = \max(x, 0)$. This loss function encourages the spherical means of different objects to be at least δ away from each other. Since our embeddings lie on the unit sphere and our distance function measures cosine distance, δ does not need to depend on the feature dimension C. In our experiments, we set $\delta = 0.5$ which encourages the clusters to be at least 90 degrees apart.

	FT3D	DAVIS	FBMS
Y-Net	<u>0.905</u>	<u>0.701</u>	<u>0.631</u>
Early Fusion	0.883	0.636	0.568
Late Fusion	0.897	0.631	0.570

Table 1: Fusion ablation. Performance is measured in IoU.

3.5. Trajectory Clustering

At inference time, we cluster the foreground trajectory embeddings with the von Mises-Fisher mean shift (vMF-MS) algorithm [26]. This gives us the clusters as well as the number of clusters, which is the estimated number of objects in a video. vMF-MS finds the modes of the kernel density estimate using the von Mises-Fisher distribution. The density can be described as $p(\mathbf{y}; \mathbf{m}, \kappa) = C(\kappa) \exp(\kappa \mathbf{m}^\intercal \mathbf{y})$ for unit vector $\mathbf{y}$ where κ is a scalar parameter, $\|\mathbf{m}\|_2 = 1$, and $C(\kappa)$ is a normalization constant. κ should be set to reflect the choice of α. If the training loss is perfect and $d(\mu_k, \mathbf{x}_i^k) < \alpha, \forall i = 1, \ldots, N_k$, then all of the $\mathbf{x}_i^k$ lie within a ball with angular radius $\cos^{-1}(1 - 2\alpha)$ of μ_k. In our experiments, we set $\alpha = 0.02$, giving $\cos^{-1}(1 - 2\alpha) \approx 16$ degrees. Thus, we set $\kappa = 10$, resulting in almost 50% of the density being concentrated in a ball with radius 16 degrees around $\mathbf{m}$ (by eyeing Figure 2.12 of [41]).

Running the full vMF-MS clustering is inefficient due to our trajectory representation being pixel-dense. Instead, we run the algorithm on a few randomly chosen seeds that are far apart in cosine distance. If the network learns to correctly predict clustered trajectory embeddings, then this random initialization should provide little variance in the results. Furthermore, we use a PyTorch-GPU implementation of the vMF-MS clustering for efficiency.

4. Experiments

Datasets. We evaluate our method on video foreground segmentation and multi-object motion segmentation on five datasets: Flying Things 3d (FT3D) [29], DAVIS2016 [35], Freibug-Berkeley motion segmentation [32], Complex Background [30], and Camouflaged Animal [6]. For FT3D, we combine object segmentation masks with foreground labels provided by [44] to produce motion segmentation masks. For DAVIS2016, we use the $\mathcal{J}$-measure and $\mathcal{F}$-measure for evaluation. For FBMS, Complex Background, and Camouflaged Animal, we use precision, recall, and F-score, and ΔObj metrics for evaluation as defined in [32, 7]. Full details of each dataset can be found in the supplement.

It is well-understood that the original FBMS labels are ambiguous [5]. Some labels exhibit multiple segmentations for one aggregate motion, or segment the (static) background into multiple regions. Thus, [5] provides corrected labels which we use for evaluation.

Implementation Details. We train our networks using

	Multi-object				Foreground		
	P	R	F	ΔObj	P	R	F
conv PT-RNN	75.9	<u>66.6</u>	<u>67.3</u>	4.9	90.3	87.6	87.7
standard PT-RNN	72.2	<u>66.6</u>	66.0	4.27	88.1	<u>89.3</u>	87.5
convGRU PT-RNN	73.6	63.8	64.8	4.07	89.6	85.8	86.3
per-frame embedding	<u>79.9</u>	56.7	59.7	11.2	<u>92.1</u>	85.4	87.4
no FG mask	63.5	60.3	59.6	<u>1.97</u>	82.5	85.7	82.1
no SCM	70.4	65.5	63.2	3.70	89.3	89.1	<u>88.1</u>
no pre-FT3D	70.2	63.6	63.1	3.66	87.6	88.2	86.3
no DAVIS-m	66.9	63.6	62.1	2.07	87.1	86.9	85.2

Table 2: Architecture and Dataset ablation on FBMS testset.

stochastic gradient descent with a fixed learning rate of 1e-2. We use backpropagation through time with sequences of length 5 to train the PT-RNN. Each image is resized to 224×400 before processing. During training (except for FT3D), we perform data augmentation, which includes translation, rotation, cropping, horizontal flipping, and color warping. We set $C = 32, \alpha = 0.02, \delta = 0.5, \kappa = 10$. We extract optical flow via [20].

Labels for each foreground trajectory are given by the frame-level label of the last pixel in the trajectory. Due to sparse labeling in the FBMS training dataset, we warp the labels using Eq. (1) so that each frame has labels. Lastly, due to the small size of FBMS (29 videos for training), we leverage the DAVIS2017 dataset [37] and hand select 42 videos from the 90 videos that roughly satisfy the rubric of [5] to augment the FBMS training set. We denote this as DAVIS-m. The exact videos in DAVIS-m can be found in the supplement.

When evaluating the full model on long videos, we suffer from GPU memory constraints. Thus, we devise a sliding window scheme to handle this. First, we cluster all foreground trajectories within a window. We match the clusters of this window with the clusters of the previous window using the Hungarian algorithm. We use distance between cluster centers as our matching cost, and further require that matched clusters must have $d(\mu_k, \mu_{k'}) < 0.2$. When a cluster is not matched to any of the previous clusters, we declare it a new object. We use a 5-frame window and adopt this scheme for the FBMS and Camouflaged Animal datasets.

In Section 4.2, we use the *conv* PT-RNN variant of Figure 2, trained for 150k iterations on FT3D, then fine-tuned on FBMS+DAVIS-m for 100k iterations.

Our implementation is in PyTorch, and all experiments run on a single NVIDIA TitanXP GPU. Given optical flow, our algorithm runs at approximately 15 FPS. Note that we do not use a CRF post-processing step for motion segmentation.

4.1. Ablation Studies

Fusion ablation. We show the choice of mid-level fusion with Y-Net is empirically a better choice than early fusion and late fusion of encoder-decoder networks. For early fusion, we concatenate RGB and optical flow and pass it through a single U-Net. For late fusion, there are two U-

		Video Foreground Segmentation							Multi-object Motion Segmentation			
		PCM [6]	FST [33]	NLC [12]	MPNet [44]	LVO [45]	CCG [7]	Ours	CVOS [43]	CUT [24]	CCG [7]	Ours
FBMS	P	79.9	83.9	86.2	87.3	92.4	85.5	90.3	72.7	74.6	74.2	75.9
	R	80.8	80.0	76.3	72.2	85.1	83.1	87.6	54.4	62.0	63.1	66.6
	F	77.3	79.6	77.3	74.8	87.0	81.9	87.7	56.3	63.6	65.0	67.3
	ΔObj	-	-	-	-	-	-	-	11.7	7.7	4.0	4.9
CB	P	84.3	87.6	79.9	86.8	74.6	87.7	83.1	60.8	67.6	64.9	57.7
	R	91.7	85.0	69.3	77.5	77.0	93.1	89.7	44.7	58.3	67.3	61.9
	F	86.6	80.6	73.7	78.2	70.5	90.1	83.5	45.8	60.3	65.6	58.3
	ΔObj	-	-	-	-	-	-	-	3.4	3.4	3.4	3.2
CA	P	81.9	73.3	82.3	77.8	77.6	80.4	78.5	84.7	77.8	83.8	77.2
	R	74.6	56.7	68.5	62.0	51.1	75.2	79.7	59.4	68.1	70.0	77.2
	F	76.3	60.4	72.5	64.8	50.8	76.0	77.1	61.5	70.0	72.2	75.3
	ΔObj	-	-	-	-	-	-	-	22.2	5.7	5.0	5.4
All	P	80.8	82.1	84.7	85.3	87.4	84.7	87.1	73.8	74.5	75.1	74.1
	R	80.7	75.8	73.9	70.7	77.2	82.7	86.2	54.3	62.8	65.0	68.2
	F	78.2	75.8	75.9	73.1	77.7	81.5	85.1	56.2	64.5	66.5	67.9
	ΔObj	-	-	-	-	-	-	-	12.9	6.8	4.1	4.8

Table 3: Results for FBMS, ComplexBackground (CB), CamouflagedAnimal (CA), and averaged over all videos in these datasets (ALL). Best results are highlighted in red with second best in blue.

		FST [33]	FSEG [21]	MPNet [44]	LVO [45]	Ours
DAVIS	$\mathcal{J}$	55.8	70.7	70.0	75.9	74.2
	$\mathcal{F}$	51.1	65.3	65.9	72.1	73.9
FT3D	IoU	-	-	85.9	-	90.7

Table 4: Results on Video Foreground Segmentation for DAVIS2016 and FT3D. Best results are highlighted in red.

Nets: one for RGB and one for optical flow, with a conv layer at the end to fuse the outputs. Note that Y-Net has more parameters than early fusion but less parameters than late fusion. Table 1 shows that Y-Net outperforms the others in terms of foreground IoU. Note that the performance gap is more prominent on the real-world datasets.

Architecture ablation. We evaluate the contribution of each part of the model and show results in both the multi-object setting and the binary setting (foreground segmentation) on the FBMS testset. All models are pre-trained on FT3D for 150k iterations and trained on FBMS+DAVIS-m for 100k iterations. Experiments with the different PT-RNN variants shows that *conv* PT-RNN performs the best empirically in terms of F-score, thus we use this in our comparison with state-of-the-art methods. *Standard* performs similarly, while *convGRU* performs worse perhaps due to overfitting to the small dataset. Next, we remove the PT-RNN architecture (per-frame embedding) and cluster the foreground pixels at each frame. The F-score drops significantly and ΔObj is much worse, which is likely due to this version not labeling clusters consistently in time. Because the foreground prediction is not affected, these numbers are still reasonable. Next, we remove foreground masks (no FG mask) and cluster all foreground and background trajectories. The clustering is more sensitive; if the background trajectories are not clustered adequately in the embedding space, the performance

will suffer. Lastly, we removed the spatial coordinate module (no SCM) and observed lower performance. Similar to the per-frame embedding experiment, foreground prediction is not affected.

Dataset ablation. We also study the effects of the training schedule and training dataset choices. In particular, we first explore the effect of not pre-training on FT3D, shown in the bottom portion of Table 2. Secondly, we explore the effect of training the model only on FBMS (without DAVIS-m). Both experiments show a noticeable drop in performance in both the multi-object and foreground/background settings, showing that these ideas are crucial to our performance.

4.2. Comparison to State-of-the-Art Methods

Video Foreground Segmentation. For FBMS, ComplexBackground and CamouflagedAnimal, we follow the protocol in [7] which converts the motion segmentation labels into a single foreground mask and use the metrics defined in [32] and report results averaged over those three datasets. We compare our method to state-of-the-art methods including PCM [6], FST [33], NLC [12], MPNet [44], LVO [45], and CCG [7]. We report results in Table 3. In terms of F-score, our model outperforms all other models on FBMS and CamouflagedAnimal, but falls just short on ComplexBackground behind PCM and CCG. Looking at all videos, we show a relative gain of 4.4% on F-score compared to the second best method CCG, due to our high recall.

Additionally, we report results of our model on FT3D and the validation set of DAVIS2016. We compare our model to state-of-the-art methods: including LVO [45], FSEG [21], MPNet [44], and FST [33] in Table 4. For this experiment only, we train a Y-Net with $C = 64$ channels on FT3D
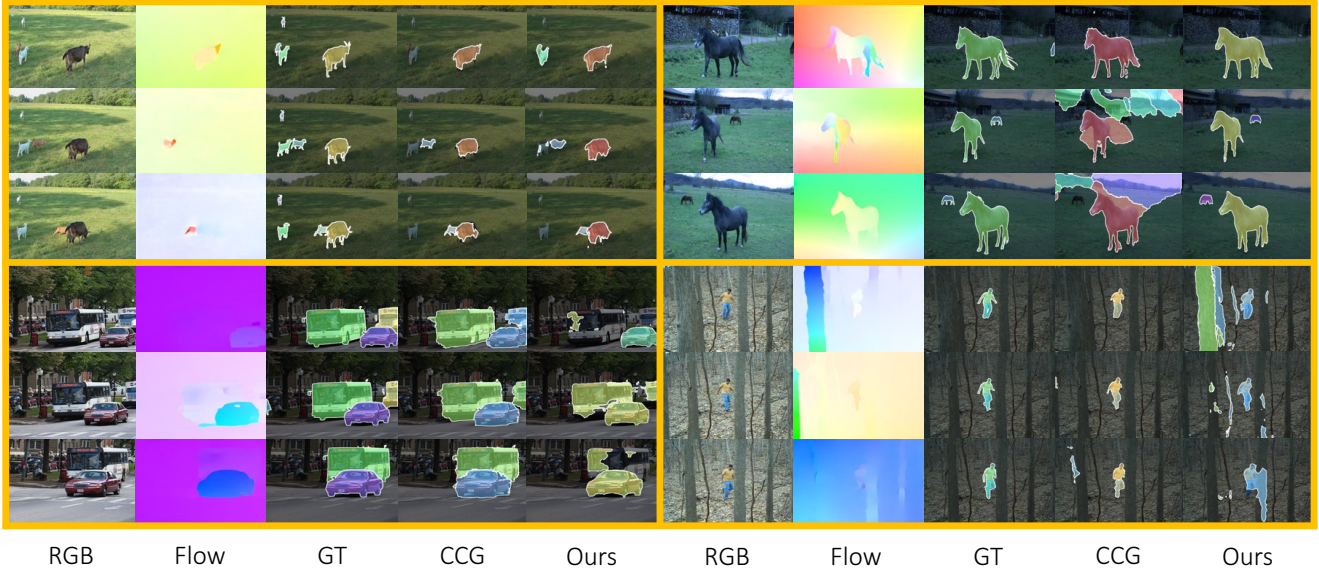
Figure 5: Qualitative results for motion segmentation. The videos are: *goats01*, *horses02*, and *cars10* from FBMS, and *forest* from ComplexBackground.

for 100k iterations, resulting in outperforming MPNet by a relative gain of 5.6%. We then fine-tune for 50k iterations on the training set of DAVIS2016 and use a CRF [28] post-processing step. We outperform all methods in terms of $\mathcal{F}$-measure and all methods but LVO on $\mathcal{J}$-measure. Note that unlike LVO, we do not utilize an RNN for video foreground segmentation, yet we still achieve performance comparable to the state-of-the-art. Also, LVO [45] reports a $\mathcal{J}$-measure of 70.1 without using a CRF, while our method attains a $\mathcal{J}$-measure of 71.4 without using a CRF. This demonstrates the efficacy of the Y-Net architecture.

Multi-object Motion Segmentation. We compare our method with state-of-the-art methods CCG [7], CUT [24], and CVOS [43]. We report our results in Table 3. We outperform all models on F-score on the FBMS and Camou-flagedAnimal datasets. On FBMS, we dominate on precision, recall, and F-score with a relative gain of 3.5% on F-score compared to the second best method CCG. Our performance on ΔObj is comparable to the other methods. On Camou-flagedAnimal, we show higher recall with lower precision, leading to a 4.4% relative gain in F-score. Again, our result on ΔObj is comparable. However, our method places third on the ComplexBackground dataset. This small 5-sequence dataset exhibits backgrounds with varying depths, which is hard for our network to correctly segment. However, we still outperform all other methods on F-score when looking at all videos. Similarly to the binary case, this is due to our high recall. Because we are the first work to use FT3D for motion segmentation, we report results on FT3D in the supplement for the interested readers.

To illustrate our method, we show qualitative results in

Figure 5. We plot RGB, optical flow [20], groundtruth, results from the state-of-the-art CCG [7], and our results on 4 sequences (*goats01*, *horses02*, and *cars10* from FBMS, and *forest* from ComplexBackground). On *goats01*, our results illustrate that due to our predicted foreground mask, our method is able to correctly segment objects that do not have instantaneous flow. CCG struggles in this setting. On *horses02*, we show a similar story, while CCG struggles to estimate rigid motions for the objects. Note that our method provides accurate segmentations without the use of a CRF post-processing step. We show two failure modes for our algorithm: 1) if the foreground mask is poor, the performance suffers as shown on *cars10* and *forest*, and 2) cluster collapse can cause multiple objects to be segmented as a single object as shown in *cars10*.

5. Conclusion

We proposed a novel deep network architecture for solving the problem of object discovery using object motion cues. We introduced an encoder-decoder network that learns representations of video frames and optical flow, and a novel recurrent neural network that learns feature embeddings of pixel trajectories inside foreground masks. By clustering these embeddings, we are able to discover and segment potentially unseen objects in videos. We demonstrated the efficacy of our approach on several motion segmentation datasets for object discovery.

Acknowledgements

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 2011. 1

[2] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *International Conference on Learning Representations (ICLR)*, 2016. 4

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994. 1

[4] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 2011. 1

[5] P. Bideau and E. Learned-Miller. A detailed rubric for motion segmentation. *arXiv preprint arXiv:1610.10033*, 2016. 1, 6

[6] P. Bideau and E. Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In *European Conference on Computer Vision (ECCV)*, pages 433–449. Springer, 2016. 2, 6, 7

[7] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. Learned-Miller. The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6, 7, 8

[8] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017. 1

[9] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision (ECCV)*, 2010. 1, 2, 3

[10] J. Cheng, J. Yang, Y. Zhou, and Y. Cui. Flexible background mixture models for foreground segmentation. *Image and Vision Computing*, 2006. 1

[11] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017. 2, 5

[12] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *British Machine Vision Conference (BMVC)*, 2014. 7

[13] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017. 2

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[15] K. Fragkiadaki, G. Zhang, and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[16] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 2016. 1

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[18] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 1

[19] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *International Journal of Computer Vision*, 2011. 1

[20] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 8

[21] S. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2017. 2, 7

[22] Z. Kalal, K. Mikolajczyk, J. Matas, et al. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 2012. 1

[23] M. Keuper. Higher-order minimum cost lifted multicuts for motion segmentation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4252–4260, 2017. 2

[24] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 7, 8

[25] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[26] T. Kobayashi and N. Otsu. Von mises-fisher mean shift for clustering on a hypersphere. In *International Conference on Pattern Recognition (ICPR)*, 2010. 6

[27] S. Kong and C. Fowlkes. Recurrent pixel embedding for instance grouping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[28] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*, 2011. 8

[29] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 6

[30] M. Narayana, A. Hanson, and E. Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2, 6

[31] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. Semi-convolutional operators for instance segmentation. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[32] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 2014. 2, 5, 6, 7

[33] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2, 7

[34] D. Pathak, R. B. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[35] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 6

[36] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[37] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 2, 6

[38] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 3, 4

[39] L. Shao, P. Shah, V. Dwaracherla, and J. Bohg. Motion-based object segmentation based on dense rgb-d scene flow. In *IEEE Robotics and Automation Letters*, volume 3, pages 3797–3804. IEEE, Oct. 2018. 2

[40] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems (NIPS)*, 2014. 3

[41] J. Straub. *Nonparametric Directional Perception*. PhD thesis, Massachusetts Institute of Technology, 2017. 6

[42] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision (ECCV)*, 2010. 2, 3, 4, 5

[43] B. Taylor, V. Karasev, and S. Soatto. Causal video object segmentation from persistence of occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 7, 8

[44] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 6, 7

[45] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3, 7, 8

[46] W. Wang, J. Shen, J. Xie, and F. Porikli. Super-trajectory for video segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1671–1679, 2017. 2

[47] Y. Wu and K. He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018. 3

[48] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[49] Y. Xiang and D. Fox. Da-rnn: Semantic mapping with data associated recurrent neural networks. *Robotics: Science and Systems*, 2017. 5

[50] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 1