SRNet: Spatial Relation Network for Efficient Single-stage Instance Segmentation in Videos

Xiaowen Ying
xiy517@lehigh.edu
Lehigh University

Xin Li
xincoder@gmail.com
Lehigh University

Mooi Choo Chuah
chuah@cse.lehigh.edu
Lehigh University

ABSTRACT

The task of instance segmentation in videos aims to consistently identify objects at pixel level throughout the entire video sequence. Existing state-of-the-art methods either follow the tracking-by-detection paradigm to employ multi-stage pipelines or directly train a complex deep model to process the entire video clips as 3D volumes. However, these methods are typically slow and resource-consuming such that they are often limited to offline processing. In this paper, we propose SRNet, a simple and efficient framework for joint segmentation and tracking of object instances in videos. The key to achieving both high efficiency and accuracy in our framework is to formulate the instance segmentation and tracking problem into a unified spatial-relation learning task where each pixel in the current frame relates to its object center, and each object center relates to its location in the previous frame. This unified learning framework allows our framework to perform join instance segmentation and tracking through a single stage while maintaining low overheads among different learning tasks. Our proposed framework can handle two different task settings and demonstrates comparable performance with state-of-the-art methods on two different benchmarks while running significantly faster.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**.

KEYWORDS

Unsupervised Video Object Segmentation; Video Instance Segmentation; Efficient; Single-stage

ACM Reference Format:
Xiaowen Ying, Xin Li, and Mooi Choo Chuah. 2021. SRNet: Spatial Relation Network for Efficient Single-stage Instance Segmentation in Videos. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3474085.3475626

1 INTRODUCTION

The problem of segmenting object instances in videos is becoming a trending topic in the past years. It is closely related to a wide range of multimedia applications such as video processing, video understanding, robotics, autonomous driving, etc. Although the problem settings and task requirements may be slightly different under different use cases, the main objective remains similar, i.e., to perform segmentation and tracking on one or multiple object instance(s) throughout the entire video clip.

Comparing to static images, video sequences pose new challenges that raise the difficulty of instance segmentation and tracking, including but not limited to camera blur, fast motion, occlusions, truncation, and the changes of object poses and appearances. However, the temporal motions of objects across video frames also provide strong clues for grouping pixels with similar motions and segmenting objects from their backgrounds. Therefore, it will be helpful to leverage temporal information when designing a framework for segmenting instances from videos.

Among a variety of task configurations related to instance segmentation in videos, the Semi-supervised Video Object Segmentation (SVOS) [32] was the first task that attracted a lot of interest in the computer vision community. SVOS requires human interaction to annotate an object of interest in the first frame, and the model is required to perform segmentation of this particular object in the subsequent frames. Although this task setting has its application scenarios (e.g., video editing), it fails to apply to a wider range of realistic scenarios because manually annotating the first frame is still costly and time-consuming, and hence is usually infeasible for real-world application scenarios.

In contrast, Unsupervised Video Object Segmentation (UVOS) [32] was another relevant task setting that does not require human annotations in any frame. This task was initially formulated as a binary segmentation problem for salient objects in a video clip. It has been later extended to differentiate different object instances in [4]. The UVOS [1] task is also closely related to a more recent task setting — Video Instance Segmentation [47] (VIS). Comparing to UVOS, VIS additionally requires the model to classify the objects into a set of predefined categories, e.g., person, vehicle, etc. Practically, UVOS can also be seen as the class-agnostic version of VIS. In this paper, we focus on the UVOS and VIS problems since these two task settings are more realistic and have a wider range of application scenarios.

Existing approaches for solving these two tasks typically follow the tracking-by-detection paradigm in which the segmentation and tracking are performed by a multi-stage pipeline. For example, one can first run an object detector on every single frame and then associate these detections using a tracking module. However, these multi-stage methods involve multiple networks that are typically computationally demanding and not end-to-end trainable. Another line of recent works considers an entire video clip as a 3D volume and directly trains an end-to-end model to generate the spatial-temporal instance masks for the whole video clip at

[1] we refer UVOS to its multi-object setting in this work unless otherwise mentioned.

MM '21, October 20–24, 2021, Virtual Event, China.

ACM ISBN 978-1-4503-8651-7/21/10.
https://doi.org/10.1145/3474085.3475626

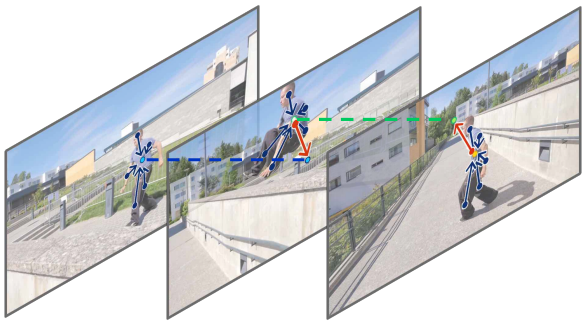

Figure 1: High-level illustration of the unified Spatial Relation Learning. Pixels belonging to an instance point to their instance center (Blue Solid Arrows) and each instance center links to its previous location (Red Solid Arrows).

once. However, these models are often inevitably complicated and resource-demanding. Moreover, when the whole video has a longer duration than the predefined window length, these approaches still need to apply an extra module for associating the tracks between two consecutive sub-clips.

In this paper, we propose a simple and efficient framework for single-stage instance segmentation in videos. Our approach solves the problem from a new perspective in which we formulate the instance segmentation and tracking problems using a unified learning objective. Concretely, we train our model to relate each pixel in the current frame to the center of the object that it belongs to and relate each object's center to its location in the previous frame, as illustrated in Figure 1. The benefit of unifying the learning objectives is that we can combine these two tasks (tracking and segmentation) into a compact and efficient multi-task model with smaller overheads for handling different tasks. The input to our framework, namely SRNet, is an RGB image of the current frame, a compact feature map from the previous frame, and a heat map indicating the locations of previously detected objects. The model can then produce all information needed for instance segmentation and tracking through a single pass. Finally, the instance mask and their track IDs can be generated via a simple post-processing algorithm.

SRNet can process videos from the first frame without buffering due to the nature of its design. Thus it is well-suited for application scenarios that require real-time processing. Although our approach intuitively trades off the ability to reconnect long-range tracks as we only consider the local temporal information from the immediately previous frame, our experimental results show that such cases have minimal impact on the performance. For example, on the YoutubeVIS benchmark, our performance is only 2.3% lower than StemSeg's 16-frames model while running 10x faster.

Our framework is originally designed for the UVOS task but can be easily extended to the VIS task by adding a lightweight head to predict the object categories. Our experimental results show that SRNet can achieve comparable performance with state-of-the-art methods in these two tasks while being significantly faster than existing methods.

We summarize our contributions as follows:

- We proposed a novel approach for efficient instance segmentation in videos.
- We tackle the problem from a novel perspective to formulate the learning of segmentation and tracking using a unified learning framework.
- Our approach can be adapted to both UVOS and VIS tasks and achieves comparable performance with state-of-the-art methods while running significantly faster.

2 RELATED WORK

Instance Segmentation on Static Images. Many existing works that tackle the instance segmentation problem are proposal-based and are typically extended from an object detection framework. These approaches first use their detection module to generate object proposals, and then refine these proposals into instance masks [8] [21] [33] [34]. A few recent works simplify the pipeline by directly predicting the masks on the proposal region via an additional branch [13] [24]. However, these methods typically suffer from relatively coarse prediction since the mask branch processes low-resolution inputs. Another line of works tackle the problem from the perspective of dense-prediction [11] [28] [18] [9]. These methods are often extended from a semantic segmentation framework and train their models to encourage pixels belonging to the same instance to be close to each other in the embedding space analogous to the learning objective of semantic segmentation. However, directly learning to map input images/videos into such an instance embedding space is difficult due to the spatial-invariant nature of CNNs. To overcome the challenges, recent works [22] [16] [29] [39] have shown that learning a more meaningful Spatial Embedding for each pixel that serves as a displacement vector from instance centers can reduce the learning difficulties and alleviate the problem of appearance ambiguities. Based on the same concept, Neven et al. [27] further proposed to learn both the spatial embedding and instance bandwidth simultaneously and incorporate the clustering step into the loss function such that directly optimizing the intersection-over-union of the final mask becomes possible. We took inspiration from [27] and followed their suggestions when designing the components for relating pixels to instance centers in our framework.

Semi-supervised Video Object Segmentation. SVOS is a task that extends the object tracking problem from bounding box level to pixel level. The task assumes the annotation for the objects of interest is given in the first frame and asks to keep tracking of these object masks in the subsequent frames at the pixel level. State-of-the-art methods for SVOS [3] [31] [44] [38] [41] [30] [46] [48] [51] [7] typically utilize the first-frame annotation to do online fine-tuning or try to directly propagate the masks to the subsequent frames. The SVOS task can be applied to some applications which require pixel-wise tracking (e.g. video editing); however, the annotation of the first frame is typically not available in real-world application scenarios.

Unsupervised Video Object Segmentation. UVOS removes the requirement of the first frame annotation. The goal is to perform segmentation and tracking for all object instances that consistently appear throughout the entire video sequence and have predominant motion in the scene. A branch of prior works [14] [15] [35] [36] [38] [49] [17] [20] [42] follow the original setting [32] which only requires a binary foreground-background segmentation. RVOS [40]

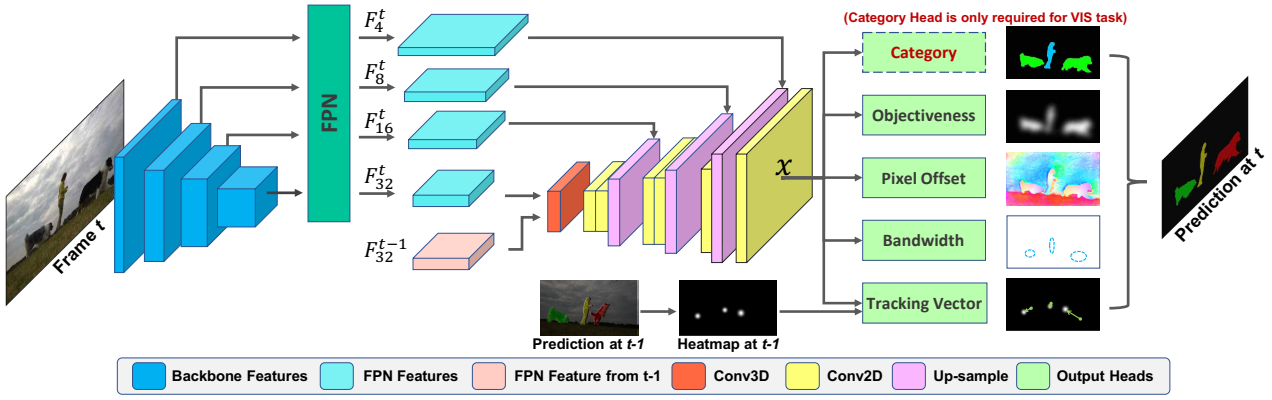

Figure 2: The architecture of our framework. Zoom in for more details. Best view in color.

is the first work that designs an end-to-end model to tackle the multi-object segmentation setting of UVOS by applying a set of RNNs to generate the mask for each instance in sequential order. Luiten et al. [25] design a proposal-detection-based framework with heuristic post-processing and obtained first place in the Davis-2019 challenge. However, [1] shows that their method is highly tailored to the challenge and does not generalize to other datasets. StemSeg [1] is another method that closely relates to our work. They extend the aforementioned instance segmentation methods [27] to handle video input as 3D volume, and train the model to group each pixel to a spatial-temporal object center in such a 3D volume. In contrast, we focus on first linking the pixels to the spatial center in the current frame, and then link each object center to its previous location. Although their approach has the ability to exploit longer temporal information (16 frames), their method is inevitably slow and resource-demanding. It is worth mentioning that Stemseg still requires an additional association process if the video is longer than its predefined input length.

Video Instance Segmentation. The task of VIS is first introduced with the YouTube-VIS dataset [47]. Compared to UVOS, VIS also requires classifying the object instances into a set of predefined categories. Apart from releasing the YTVIS benchmark, Yang et al. [47] also proposed a framework for VIS, namely MaskTrack-RCNN, by extending the MaskRCNN with an additional tracking branch. A few of the aforementioned works [1] [41] [48] are also compatible with the VIS task. Wang et al. [43] proposed a transformer-based framework for VIS which is extended from a recently proposed transformer-based object detection framework [6]. Similar to [1], their approach directly inputs an entire video clip with predefined length (36 frames) and predicts the spatial-temporal instance masks for the whole sequence. While this is a new promising direction based on Transformer, at this stage their approach is resource-demanding and cannot be applied to videos with longer duration than the predefined window length (see 4.4 for more discussion).

3 PROPOSED METHOD

The core idea of our approach is to formulate the learning of instance segmentation and tracking into a unified spatial relation problem. The network takes inputs as an RGB image at frame t and information from its immediate previous frame $t-1$ and is trained to relate each pixel to its instance center and relate each instance center to its location in the previous frame. This is achieved by learning offset vectors for each pixel, pointing to its desired location. With this unified learning framework, our problem can be solved using a simple, compact, and efficient network architecture. Our network architecture is illustrated in Figure 2 and is explained in Section 3.1. More details about the training scheme are explained in Section 3.2 - 3.6. Our approach is an end-to-end single-stage method that predicts everything needed by the video instance segmentation and tracking association through a single pass. A simple post-processing algorithm is sufficient to aggregate the predictions and generate the final results, which is explained in Section 3.7.

3.1 Architecture

The network architecture of SRNet follows the encoder-decoder framework that has been widely used in dense-prediction problems. The encoder consists of a backbone connected to a Feature Pyramid Network (FPN) following the practice of He et al. [13]. The FPN takes as input the feature maps from different stages of the backbone and produces compressed feature maps at four different scales $\{F_s, s = 4, 8, 16, 32\}$ which are used as the input to our decoder.

Our decoder is a standard multi-head decoder with skip connections except that we modify the first convolution block to handle spatial-temporal features. Concretely, our decoder additionally takes as input the features at 1/32 scale from the previous frame F_{32}^{t-1} and stacks it with F_{32}^t along the temporal dimension. We apply a 3D convolution layer with $(2 \times 3 \times 3)$ kernel size on this stacked feature map to capture spatial-temporal features such as the object motions. Since the dimension of this feature map is very small $(256 \times \frac{H}{32} \times \frac{W}{32})$ in our case, it drastically reduces the computational cost of performing expensive 3D convolutions and has low memory usages. We experimentally analyze the impact of involving temporal information in Section 4.3.

The output of our decoder is produced by several lightweight output heads consists of only two convolution layers. All output heads operate on the feature map $\mathcal{X}$ produced by the last decoder block at 1/4 resolution except the input of the tracking head is a concatenation of $\mathcal{X}$ and an instance heatmap $\mathcal{H}^{t-1}$ rendered from

the previous prediction, using the Gaussian rendering function described in [19]. We provide this clue to the tracking head to alleviate the ambiguity of center location when predicting the tracking vector. The effectiveness of adding this clue is experimentally analyzed in Section 4.3.

3.2 Spatial Grid Construction

We start by constructing a unified Spatial Grid for our spatial learning framework. The Spatial Grid encodes the positional information for each pixel such that the predicted offset vectors are meaningful. Although it is possible to directly use the pixel coordinates as the spatial grid [52], the resulting grid becomes scale-variant, e.g., the offset vectors predicted at different scales have different magnitudes and hence cannot be directly interpolated when up-sampling. To avoid this problem we construct our Spatial Grid $\mathcal{S} \in \mathcal{R}^{H_s \times W_s \times 2}$ using Equation (1).

$$\mathcal{S}_{i,j} = [\frac{2T_x \cdot i}{W_s - 1} - T_x, \frac{2T_y \cdot j}{H_s - 1} - T_y], \tag{1}$$

where $T_x = max(1, W_s/H_s)$ and $T_y = max(1, H_s/W_s)$.

This equation creates a normalized spatial grid that is scale and size invariant. It is easy to show that bilinearly upsamples $\mathcal{S}$ to a higher resolution is equivalent to creating a new spatial grid at that target resolution. The constructed spatial grid serves as the spatial coordinates and is used for both instance segmentation and tracking task.

3.3 Learning Pixel Offsets for Segmenting Object Instances

The pixel-offset head in our framework predicts an offset vector $\mathcal{E}_{i,j} \in \mathcal{R}^2$ for each pixel that points to its instance center. Ideally, we would expect the following holds for every pixel belonging to the instance n:

$$\left\|\mathcal{E}_{i,j} + \mathcal{S}_{i,j} - c_n\right\| \leq \epsilon, \tag{2}$$

where $\mathcal{E}_{i,j}$ is the output from our Pixel-offset Head at location (i, j), c_n is the center of instance n and ϵ is the acceptable error since the prediction can never be perfectly accurate.

Setting the value of ϵ to be stricter will increase the learning difficulty, while a more relax ϵ may result in inaccurate predictions. A good practice is to set the value of ϵ to be proportional to the size of the objects since it is more difficult for large objects to relate pixels near the edges to its object center due to the Effective Receptive Field Decay [26] nature of CNNs. To alleviate this problem, we adopt the suggestions from [27] to learn the instance bandwidth $\Sigma_{i,j} \in \mathcal{R}^2$ for each pixel, and it is predicted by our Bandwidth Head.

Equation 2 defines a decision boundary where pixels (after adding the offset vectors) that fall within a certain range from an instance center should belong to this instance. Since this is a hard decision boundary that is not suitable for training, a Gaussian function can be used to model the decision boundary as a smoothed probability distribution that is differentiable. With this formulation, the probability of pixel (i, j) belongs to instance n is:

$$\mathcal{P}^n_{i,j} = exp(-\frac{\left\|\mathcal{E}_{i,j} + \mathcal{S}_{i,j} - c_n\right\|^2}{2\Sigma_n^2}), \tag{3}$$

where Σ_n is the predicted bandwidth of instance n.

During training, Σ_n is set as the average of all $\Sigma_{i,j \in M_n}$, where M_n is the set of foreground pixels in the ground truth mask of instance n. Similarly, we set c_n equals to the average of $\mathcal{E}_{i,j \in M_n} + \mathcal{S}_{i,j \in M_n}$ during training. During inference, c_n and Σ_n can be obtained by:

$$\begin{aligned} c_n &= \mathcal{E}_{c'_n} + \mathcal{S}_{c'_n} \\ \Sigma_n &= \Sigma_{c'_n}, \end{aligned} \tag{4}$$

where c'_n is the location of predicted instance center from the Objectiveness Head (described in Section 3.5).

Finally, the learning of pixel offset $\mathcal{E}$ can be optimized along with the predicted instance bandwidth Σ using any dense-prediction loss that maximizes the Intersection-Over-Union (IOU) between the prediction and ground truth instance mask.

$$\mathcal{L}_{seg} = \frac{1}{N}\sum_{n}^{N}(\mathcal{L}_{IOU}(\mathcal{P}^n, M_n) + \frac{1}{|M_n|}\sum_{i,j \in M_n}\left\|\Sigma_{i,j} - \Sigma_n\right\|), \tag{5}$$

where M_n is the ground truth binary mask for instance n. $|M_n|$ denotes the number of foreground pixels of the instance mask. The second term is a smooth loss that encourages the predicted bandwidth to have similar values within an instance. For more detail about the smooth term, we refer to [27].

We have experimentally tested different choices of $\mathcal{L}_{IOU}$ including two popular surrogate functions to approximate the IOU — Dice Loss [37] and Lovást Hinge Loss [50]. The impact of these two choices [2] is further discussed in Section 4.3.

Extra Dimensions. The above formulation describes process of learning 2D pixel offsets for grouping pixels of instances. Athar et al. [1] found that allowing the network to predict extra dimensions is beneficial. Such extra dimensions provide free representation space for network to learn and do not have a geometric interpretation. While they validated the effectiveness of this extension in the case of 3D space-time volume, we found that adding such extra dimensions is also helpful in our spatial-relation learning. For example, to extend the current formulation with two extra dimensions, we change $\mathcal{E}_{i,j}$ from $\mathcal{R}^2$ to $\mathcal{R}^4$, and extend $\mathcal{S}_{i,j}$ to $\left[\begin{array}{c|cc} \mathcal{S}_{i,j} & 0 & 0 \end{array}\right]$. The bandwidth $\Sigma_{i,j}$ is extended to $\left[\begin{array}{c|cc} \Sigma_{i,j} & \phi_1 & \phi_2 \end{array}\right]$, where ϕ_1 and ϕ_2 are the predefined bandwidths in the two extra dimensions and are part of the hyper-parameters. We choose to use two extra dimensions in our final model. The effectiveness of extra dimensions is studied in Section 4.3.

3.4 Learning Tracking Vector

The tracking offset is learned in a similar manner as the pixel-offset under the same spatial coordinate defined by the Spatial Grid $\mathcal{S}$. We train the network to predict a tracking vector $\mathcal{T}$ for each pixel, and expect the predicted vector links each center c^t_n to c^{t-1}_n:

$$\mathcal{T}_{c^t_n} + \mathcal{S}_{c^t_n} = \mathcal{S}_{c^{t-1}_n} \tag{6}$$

Compared to Equation 2, we did not consider the error term ϵ in Equation 6. This is because we consider tracking as a one-to-one association problem and we always associate the best match.

$$\mathcal{L}_{track} = \frac{1}{N}\sum_{n}^{N}\left\|\mathcal{T}_{c_n^t} + \mathcal{S}_{c_n^t} - \mathcal{S}_{c_n^{t-1}}\right\| \quad (7)$$

During both training and testing, c_n^t and c_n^{t-1} in Equation 7 is set as the mass center of M_n^t and M_n^{t-1}, respectively. Note that this is different from Equation 3 as we found that using the mass center is more stable for tracking. We discuss the impact of using different centers in our supplementary material.

3.5 Objectiveness

Our network also predicts an objectiveness map $\mathcal{O}$ via the Objectiveness Head to provides clues for locating the potential instances. During training, we encourage pixels that are close to the center of an instance to have higher objectiveness scores and pixels that do not belong to any object to have lower scores.

$$\mathcal{L}_{score} = \frac{1}{|S|}\sum_{i,j}\mathbb{1}_{\{i,j\in M\}}\left\|\mathcal{O}_{i,j} - \mathcal{P}_{i,j}\right\|^2 + \mathbb{1}_{\{i,j\notin M\}}\left\|\mathcal{O}_{i,j}\right\|^2 \quad (8)$$

where $M = \bigcup_{n=1}^{N} M_n$ is the set of all foreground pixels.

Unlike the object heatmap in [52] or the seed map in [27] that splits the objects into different channels based on their categories, our objectiveness map is learned in a class-agnostic fashion. Therefore the model is more flexible and can be reused for different datasets without having to retrain the model (or only finetune the category head if doing VIS task).

3.6 Extension to Video Instance Segmentation

So far the framework is already capable of doing the UVOS task, and it can be further extended to the VIS task by adding a lightweight category head that predicts the category for each pixel $\mathcal{Y}_{i,j}$. Our category head is trained using a standard Cross-entropy Loss.

$$\mathcal{L}_{cat} = CE(\mathcal{Y}^{Pred}, \mathcal{Y}^{GT}) \quad (9)$$

3.7 Inference

During inference, we first feed the inputs (frame t, previous feature F_{32}^{t-1} and instance heatmap $\mathcal{H}^{t-1}$) to the network and collect the outputs from each head. For the first frame ($t = 0$) in the sequence when no previous frame is available, we let $F_{32}^{t-1} = F_{32}^{t}$ and set $\mathcal{H}^{t-1}$ to be all zeros. Then, a simple and efficient post-processing algorithm is used to produce the final output. The post-processing algorithm in our framework is modified from the Sequential Clustering in [27] where we extend the algorithm to involve the components for tracking association and category classification. The detail of our post-processing algorithm is described in Algorithm 1.

In brief, our post-processing algorithm first locates the center of potential instances using the information from Objectiveness Head. Then, the algorithm clusters all pixels belonging to each instance based on the predicted pixel offsets and bandwidths. Finally, the information from the tracking head is used to determine the instance IDs by linking new instances with existing tracks. In the VIS task, the information from the category head is used to classify the objects.

Algorithm 1: Post-processing

Input: Spatial Grid: $\mathcal{S} \in \mathcal{R}^{H_s \times W_s \times 2}$
Pixel Offsets: $\mathcal{E} \in \mathcal{R}^{H_s \times W_s \times 2}$
Bandwidth: $\Sigma \in \mathcal{R}^{H_s \times W_s \times 2}$
Tracking Vectors: $\mathcal{T} \in \mathcal{R}^{H_s \times W_s \times 2}$
Objectiveness: $\mathcal{O} \in \mathcal{R}^{H_s \times W_s \times 1}$
Classification: $\mathcal{Y} \in \mathcal{R}^{H_s \times W_s \times C}$
$Dets^{t-1}$: List of instance in previous frame.

$Dets^t \leftarrow []$;
$available\_pixels \leftarrow$ ones_like($\mathcal{O}$);
while $max(\mathcal{O}$ & $available\_pixels) > min\_score$ **do**
  $I \leftarrow$ new Instance;
  $c_n' \leftarrow argmax(\mathcal{O}$ & $available\_pixels)$;
  $c_n, \Sigma_n \leftarrow$ Solving equation (4) ;
  $\mathcal{P}_n \leftarrow$ Solving equation (3) ;
  $I.mask \leftarrow \mathcal{P}_n > 0.5$;
  $I.center \leftarrow$ mass_center(I.mask) ;
  $I.tracking \leftarrow I.center + \mathcal{T}_{I.center}$;
  $I.class \leftarrow argmax(\sum_{i,j\in I.mask} \mathcal{Y}_{i,j})$;
  $Dets^t$.append(I) ;
  $available\_pixels[I.mask] \leftarrow 0$;
end
$distances \leftarrow$ pairwise distance between each of $\{I.center \in Dets^{t-1}\}$ and each of $\{I.tracking \in Dets^t\}$;
$assignments \leftarrow$ GreedyAssignment(distances) ;
for (a_1, a_2) *in assignments* **do**
  $Dets^t[a_2].ID \leftarrow Dets^{t-1}[a_1].ID$;
end
return $Dets^t$

4 EXPERIMENTS

4.1 Implementation Details

Network Details. We use the ResNet-101 architecture as the backbone in our encoder. We initialize the backbone and the FPN with weights from Mask R-CNN [13] pretrained on MS-COCO dataset [23]. The output channel of FPN is set to 256. The kernel size of the 3D convolution in decoder block 1 is set as $(2 \times 3 \times 3)$. All output heads use the same structure which consists of a 3×3 Conv2D layer followed by a ReLU activation and a final 1×1 Conv2D layer.

Training. Our method is implemented using the PyTorch framework. For training, we use a momentum optimizer with an initial learning rate of 1e-3. The learning rate decays exponentially starting from the 60K iteration and finally reduced to 1e-5. Our network is trained on a single NVIDIA GTX 1080Ti GPU for 160K iterations with a batch size of 4. The final optimization loss is the summation of all the aforementioned loss terms with each term being equally weighted. Since our framework only considers 2 consecutive frames, we augment our training data with synthesized data generated from static images for better generalization following the same practice of [1]. These synthesized data were produced by applying random affine transformations on static images from

$\mathcal{L}_{IOU}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$-Mean	$\mathcal{F}$-Mean
Dice	58.1	56.5	59.7
Lovast Hinge	59.7	58.2	61.3

Table 1: Impact of different choice for $\mathcal{L}_{IOU}$

Experiment	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$-Mean	$\mathcal{F}$-Mean
Baseline	59.7	58.2	61.3
w/o Instance Heatmaps	58.6 (-1.1)	57.1	60.1
w/o Temporal Features	56.9 (-2.8)	55.3	58.5
w/o Extra Dimensions	58.4 (-1.3)	56.7	60.0
w/o Synthesized Data	54.5 (-5.2)	52.7	56.3

Table 2: Effectiveness of different components.

Backbone	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$-Mean	$\mathcal{F}$-Mean	FPS
ResNet-50	58.1	56.3	59.9	44
ResNet-101	59.7	58.2	61.3	35

Table 3: Performance and speed using different backbone.

MS-COCO[23] and Pascal-VOC [10] dataset. We study the impact of adding these synthesized data in Section 4.3.

4.2 Datasets

DAVIS: DAVIS is a popular benchmark for video object segmentation. It consists of 60 video sequences for training and 30 for validation. The dataset has many versions and different task settings, and we use the DAVIS-2019 Unsupervised Multi-object Segmentation [4] benchmark for evaluating our approach. This benchmark is provided for the UVOS task where the goal is to perform instance segmentation on tracking for all foreground objects. The evaluation metrics includes $\mathcal{J}$-score (IOU) and $\mathcal{F}$-score (boundary precision). The mean of these scores, namely $\mathcal{J}\&\mathcal{F}$ is used as the main score for comparison.

YouTube-VIS: The Youtube-VIS dataset is a large-scale dataset for video instance segmentation. This benchmark extends the image instance segmentation task from the image domain to the video domain. The dataset includes 2,238 video sequences for training and 302 video sequence for validation. The number of video sequences is orders of magnitude greater than the DAVIS dataset but the video sequence is shorter (the longest video in YTVIS only has 36 frames). Youtube-VIS employs the Average Precision (AP) and Average Recall (AR) as the evaluation metrics.

4.3 Ablation Study

In this section, we perform ablation studies to analyze the impact of alternating different components in our framework. We conduct experiments on the DAVIS-19 benchmark for its validation set is available for offline evaluation.

Choice of $\mathcal{L}_{IOU}$: We study the impact of choosing different losses for $\mathcal{L}_{IOU}$. We experiments with Dice Loss [37] and Lovasz Hinge loss [50] [2], both are widely used surrogate functions to approximate the Intersection-Over-Union. Table 1 compare the results of using these two losses. We can see that training with the Lovasz Hinge loss achieves better results under both $\mathcal{J}$ measure and $\mathcal{F}$ measure. We suspect that this is due to the inherent nature of the Lovasz Hinge Loss which focuses more on optimizing the samples outside the boundary and stops optimizing them once they are within the decision boundary. This characteristic is very suitable for our learning task since the predicted bandwidth defines a tolerant range for clustering the instance pixels, and there is no need to keep pushing them towards the center if they have fallen within the bandwidth.

Instance Heatmap. To validate the effectiveness of providing instance heatmap $\mathcal{H}^{t-1}$ to the tracking head, we conduct an ablation study where we set the values in $\mathcal{H}^{t-1}$ to be all zeros before feeding into the model. Note that modifying $\mathcal{H}^{t-1}$ will only affect the tracking prediction and does not affect the predicted segmentation mask since there is no dependency between them (as shown in Figure 2). As we can see in Table 2, if we remove the information in $\mathcal{H}^{t-1}$, the performance drops 1.1% in terms of the $\mathcal{J}\&\mathcal{F}$ score. This shows that the tracking head is not completely relied on the instance heatmap $\mathcal{H}^{t-1}$ but providing such clues allows the network to predict more precise tracking vectors.

Temporal Features. Temporal information provides strong clues to perform segmentation on moving objects which is helpful especially for the UVOS task. To study the impact of using temporal features, we run the inference process using our best model but let $F_{32}^{t-1} = F_{32}^{t}$. This is equivalent to repeating the current frame so that the previous frame is the same as the current frame. It is worth mentioning that this strategy is already used during our normal inference process for the first frame when no previous frame is available, thus doing so will not break the model's prediction. As we can see in Table 2, involving temporal features brings 2.8% overall improvements on the $\mathcal{J}\&\mathcal{F}$ score.

Extra Dimensions. We found that involving extra dimensions in the predicted pixel offsets $\mathcal{E}$ is beneficial even our problem formulation is purely 2D. Our final model involves two extra dimensions following the suggestion from [1] such that $\mathcal{E} \in \mathcal{R}^4$. As we can see in Table 2, it brings 1.3% improvements compared to the model without extra dimensions.

Synthesized Data. Our model is trained using a combination of video data and synthesized data augmented from static images as described in 4.1. This brings 5.2% improvements compared to the model trained without synthesized data. Since our model performs instance segmentation in a class-agnostic way and only uses short-term temporal information, combining more synthesized training data helps our model to generalize better without heavily relying on temporal information.

Backbone: Our main model uses ResNet-101 as our backbone but we also evaluate a variant of our model by changing the backbone to ResNet-50. The comparison is summarized in Table 3. We can see that switching to a lighter backbone only has a minor performance drop (1.6% on the $\mathcal{J}\&\mathcal{F}$) which validates the effectiveness of our framework that is compatible with different backbones. In addition, using a lighter backbone increases the inference speed to 56 FPS. This shows that the time spent on backbone feature extraction is a bottleneck in our framework and the speed could potentially be further increases by using a faster backbone.

4.4 Comparison with state of the art

Since our approach is compatible with both UVOS and VIS tasks, we compare our results with other state-of-the-art using both

Methods	#Frames	Proposal	Flow	Re-ID	FPS	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$-Mean	$\mathcal{J}$-Recall	$\mathcal{J}$-Decay	$\mathcal{F}$-Mean	$\mathcal{F}$-Recall	$\mathcal{F}$-Decay
UnOVOST † [25]	1	✓	✓	✓	<1	67.0	65.6	75.5	0.3	68.4	75.9	3.7
STEm-Seg [1]	16				7	64.7	61.5	70.4	-4.0	67.8	75.5	1.2
OF-Tracker [1]	1	✓	✓		1	54.6	53.4	60.9	-1.3	55.9	63.0	1.1
RI-Tracker [1]	1	✓		✓	<1	56.9	55.5	63.3	2.7	58.2	64.4	6.4
ALBA [12]	1	✓	✓		<1	58.4	56.6	63.4	7.7	60.2	63.1	7.9
AGNN [42]	1	✓	✓		<1	61.1	58.9	65.7	11.7	63.2	67.1	14.3
RVOS [40]	1				17	41.2	36.8	40.2	0.5	45.7	46.4	1.7
Ours	1				35	59.7	58.2	66.6	-3.7	61.3	68.2	-0.9

Table 4: Results on the Validation Set of DAVIS-2019 Unsupervised Track. Proposal: use proposals generated by separate proposal network. Flow: use optical flow. Re-ID: perform additional Re-ID processing. †: Approaches that uses heuristic post-processing (see 4.4 for more discussion).

Methods	#Frames	Proposal	FPS	mAP	AP@50	AP@75	AR@1	AR@10
STEm-Seg [1]	16		3	34.6	55.8	37.9	34.4	41.6
VisTR [43]	36		12	38.6	61.3	42.3	37.6	44.2
IoUTracker+ [47]	1	✓	-	23.6	39.2	25.5	26.2	30.9
DeepSORT [45]	1	✓	-	26.1	42.9	26.1	27.8	31.3
OSMN [48]	1	✓	-	27.5	45.1	29.1	28.6	33.1
SeqTracker [47]	1		-	27.5	45.7	28.7	29.7	32.5
MaskTrack R-CNN [47]	1		17	30.3	51.1	32.6	31	35.5
SipMask [5]	1		14	33.7	54.1	35.8	35.4	40.1
Ours	1		35	32.3	50.2	34.8	32.3	40.1

Table 5: Results on the Validation Set of Youtube-VIS Video Instance Segmentation task.

DAVIS-2019 Unsupervised benchmark and **Youtube-VIS** benchmark (2019 version).

Unsupervised Video Object Segmentation on DAVIS-2019. Our results for the UVOS task compared with other state-of-the-art are summarized in Table 4. We can see that our simple and efficient framework can achieve comparable performance with other state-of-the-art methods while being significantly faster. Most of the existing methods on this table such as UnOVOST, ALBA, and AGNN uses proposals generated by Mask-RCNN and information from Optical Flow (typically also predicted by another network) apart from their model. Such a multi-stage pipeline with stacked networks is usually beneficial to the performance but also greatly increases the computation cost. As a result, we can see that most of these approaches are running under 1 FPS. UnOVOST[25] is the winning solution for the DAVIS challenge that achieves the best score on the table by stacking several networks with heuristic-based post-processing. Experimental results in [1] suggest that UnOVOST is highly tailored to the DAVIS dataset and does not perform equally well on other datasets.

Compared to other approaches that are not proposal-based or using optical flow, STEm-Seg achieves the best performance by using a 3D-volume-based model to process an entire video clip with 16 frames. For videos longer than 16 frames, they perform an additional step to associate the objects by setting an 8-frames overlap between two temporal windows. While their approach achieves better performance by leveraging more temporal information, it inevitably runs slower and is not suitable for online processing.

Video Instance Segmentation on Youtube-VIS. Table 5 summarizes our results on the Youtube-VIS benchmark for the Video Instance Segmentation Task. Our approach achieves the best speed-accuracy trade-off among all methods on the table. Note that the inference speed of STEm-Seg dropped to 3FPS compared to Table 4, this is because they use an additional full-scale decoder to predict the category, while our framework efficiently uses a lightweight category head. VisTR[43] is a recently proposed VIS method based on the powerful Transformer framework. It directly inputs an entire video with a maximum of 36 frames and generates the results in an end-to-end fashion. Although being a new promising direction for this task, it still suffers from several weaknesses at this point: (1) once the model is trained, it cannot process video longer than the predefined window length (they set it as 36 the maximum video length in YTVIS dataset); (2) it is not flexible and inefficient for online processing as the computation cost for processing 1 frame is equivalent to 36 frames because input clips shorter than 36 frames is padded to 36 frames; (3) it is not scalable for a longer predefined window due to the exponential growth of the computational cost in the Transformer; (4) the training cost is expensive as it requires multiple GPUs with at least 32GB memory for each GPU for training, while our model can be trained on a single NVIDIA 1080-Ti GPU.

Among all the online methods in Table 5, our approach achieves comparable results with top methods, while being at least 2X faster. It is worth mentioning that these methods are typically designed specifically for the VIS task, while our approach is compatible with both UVOS and VIS tasks.

Speed Comparison. All the FPS number reported in Table 4 and Table 5 are evaluated on a single NVIDIA 1080Ti GPU. We tried our best to fairly compare the inference speed for all these methods by

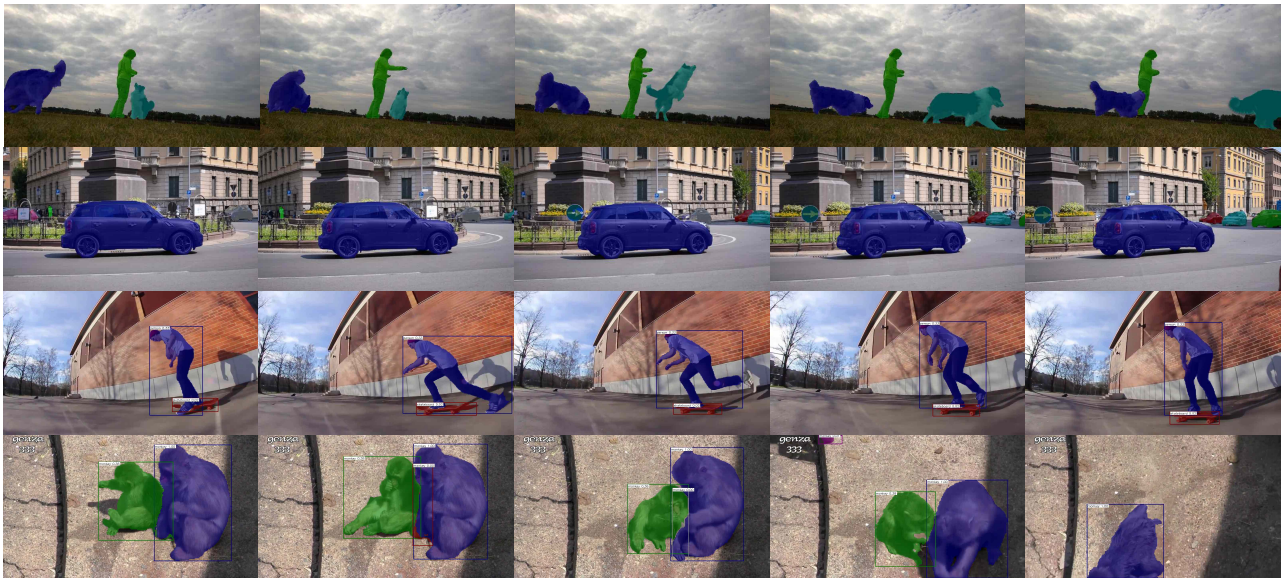

Figure 3: Qualitative results generated by our model. First two rows are sequences from DAVIS-2019 validation set while the last two rows are from Youtube-VIS validation set. Zoom in for more details. Best view in color.

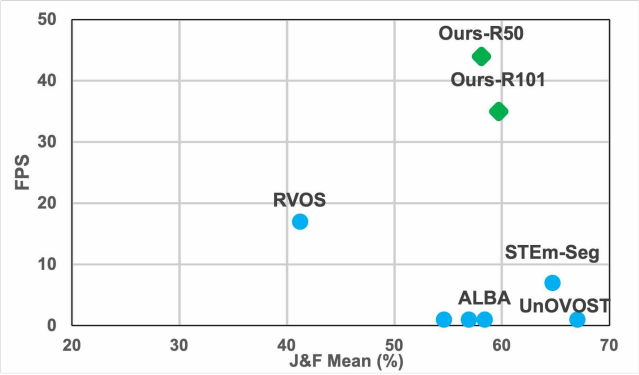

Figure 4: Speed & Accuracy Trade-off on DAVIS-2019 UVOS benchmark.

downloading their source code and running their inference under the same hardware environments on our server. OF-Tracker and RI-Tracker are two exceptions since we do not have access to their source code but the speed reported in their paper was also tested on an NVIDIA 1080Ti GPU so we quoted the numbers from their paper [1].

Since several methods on the tables are not capable of performing online processing, we conduct all speed evaluations under the offline setting for fair comparisons. For each approach, we use their code to perform inference on at least 500 frames and divide it by the total processing time. The time spent on data preparation and results saving are excluded since they can usually be further optimized. Apart from the GPUs, our evaluation server is equipped with an Intel Xeon E5-2620 v4 CPU running at 2.10GHz and 64GB Memory. More detailed analysis and discussion of our inference speed can be found in our supplementary materials.

4.5 Qualitative Analysis

We provide qualitative results generated from our model in Figure 3. Although being simple and efficient, our approach can generate satisfying results in many challenging cases. The example in the first row illustrates our framework's ability to perform instance segmentation and tracking for multiple objects simultaneously. The second row shows a similar multi-object situation but is more challenging since new objects are joining the scene as the video proceeds. The sequence in the third row shows our model can keep track of both large and small objects even when they are connected. In the fourth row, two monkeys are staying together with similar appearances while our model succeeds in separating them with a clean and accurate boundary. For more qualitative results we refer readers to our supplementary materials.

5 CONCLUSION

In this paper, we proposed an end-to-end single-stage framework for instance segmentation in videos, targeting applications that require efficient and high-speed processing. We achieved this by formulating the problem of instance segmentation and tracking from a novel perspective where these two tasks are learned under a unified learning framework. This unified formulation allows us to learn a multi-task network using a simple and efficient network with minimum overheads among different tasks. We validate the effectiveness of this framework on both the Unsupervised Video Object Segmentation Task and the Video Instance Segmentation Task and achieved comparable results with state-of-the-art methods while being significantly faster.

Acknowledgement. This work was partially supported by National Science Foundation Grant CPS 1931867, and a gift from Qualcomm Inc.

REFERENCES

[1] Ali Athar, Sabarinath Mahadevan, Aljosa Osep, Laura Leal-Taixé, and Bastian Leibe. 2020. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *European Conference on Computer Vision*. Springer, 158–177.

[2] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. 2018. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4413–4421.

[3] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. 2017. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 221–230.

[4] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. 2019. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737* (2019).

[5] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. 2020. SipMask: Spatial Information Preservation for Fast Image and Video Instance Segmentation. *arXiv preprint arXiv:2007.14772* (2020).

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 213–229.

[7] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. 2018. Blazingly fast video object segmentation with pixel-wise metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1189–1198.

[8] Jifeng Dai, Kaiming He, and Jian Sun. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3150–3158.

[9] Bert De Brabandere, Davy Neven, and Luc Van Gool. 2017. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551* (2017).

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[11] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. 2017. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277* (2017).

[12] Shreyank N Gowda, Panagiotis Eustratiadis, Timothy Hospedales, and Laura Sevilla-Lara. 2020. ALBA: Reinforcement Learning for Video Object Segmentation. *arXiv preprint arXiv:2005.13039* (2020).

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[14] Rui Hou, Chen Chen, Rahul Sukthankar, and Mubarak Shah. 2019. An efficient 3D CNN for action/object segmentation in video. *arXiv preprint arXiv:1907.08895* (2019).

[15] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. 2017. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 2117–2126.

[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

[17] Yeong Jun Koh and Chang-Su Kim. 2017. Primary object segmentation in videos based on region augmentation and reduction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 7417–7425.

[18] Shu Kong and Charless C Fowlkes. 2018. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9018–9028.

[19] Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*. 734–750.

[20] Siyang Li, Bryan Seybold, Alexey Vorobyov, Alireza Fathi, Qin Huang, and C-C Jay Kuo. 2018. Instance embedding transfer to unsupervised video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6526–6535.

[21] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. 2017. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2359–2367.

[22] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. 2017. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2978–2991.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[24] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. 2018. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8759–8768.

[25] Jonathon Luiten, Idil Esen Zulfikar, and Bastian Leibe. 2020. Unovost: Unsupervised offline video object segmentation and tracking. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2000–2009.

[26] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. 2017. Understanding the effective receptive field in deep convolutional neural networks. *arXiv preprint arXiv:1701.04128* (2017).

[27] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. 2019. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8837–8845.

[28] Alejandro Newell, Zhiao Huang, and Jia Deng. 2016. Associative embedding: End-to-end learning for joint detection and grouping. *arXiv preprint arXiv:1611.05424* (2016).

[29] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. 2018. Semi-convolutional operators for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 86–102.

[30] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. 2018. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7376–7385.

[31] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. 2019. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9226–9235.

[32] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. 2016. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 724–732.

[33] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. 2015. Learning to segment object candidates. *arXiv preprint arXiv:1506.06204* (2015).

[34] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. 2016. Learning to refine object segments. In *European conference on computer vision*. Springer, 75–91.

[35] Mennatullah Siam, Chen Jiang, Steven Lu, Laura Petrich, Mahmoud Gamal, Mohamed Elhoseiny, and Martin Jagersand. 2019. Video object segmentation using teacher-student adaptation in a human robot interaction (hri) setting. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 50–56.

[36] Hongmei Song, Wenguan Wang, Sanyuan Zhao, Jianbing Shen, and Kin-Man Lam. 2018. Pyramid dilated deeper convlstm for video salient object detection. In *Proceedings of the European conference on computer vision (ECCV)*. 715–731.

[37] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 240–248.

[38] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. 2017. Learning video object segmentation with visual memory. In *Proceedings of the IEEE International Conference on Computer Vision*. 4481–4490.

[39] Jonas Uhrig, Eike Rehder, Björn Fröhlich, Uwe Franke, and Thomas Brox. 2018. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 292–299.

[40] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro-i Nieto. 2019. Rvos: End-to-end recurrent network for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5277–5286.

[41] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. 2019. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9481–9490.

[42] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J Crandall, and Ling Shao. 2019. Zero-shot video object segmentation via attentive graph neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9236–9245.

[43] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. 2020. End-to-End Video Instance Segmentation with Transformers. *arXiv preprint arXiv:2011.14503* (2020).

[44] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. 2019. Ranet: Ranking attention network for fast video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3978–3987.

[45] Nicolai Wojke and Alex Bewley. 2018. Deep Cosine Metric Learning for Person Re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 748–756.

[46] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. 2018. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 585–601.

[47] Linjie Yang, Yuchen Fan, and Ning Xu. 2019. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5188–5197.

[48] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. 2018. Efficient video object segmentation via network modulation. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 6499–6507.

[49] Zhao Yang, Qiang Wang, Luca Bertinetto, Weiming Hu, Song Bai, and Philip HS Torr. 2019. Anchor diffusion for unsupervised video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 931–940.

[50] Jiaqian Yu and Matthew Blaschko. 2015. Learning submodular losses with the Lovász hinge. In *International Conference on Machine Learning*. PMLR, 1623–1631.

[51] Xiaohui Zeng, Renjie Liao, Li Gu, Yuwen Xiong, Sanja Fidler, and Raquel Urtasun. 2019. Dmm-net: Differentiable mask-matching network for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3929–3938.

[52] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking objects as points. In *European Conference on Computer Vision*. Springer, 474–490.