Efficient Video Instance Segmentation via Tracklet Query and Proposal

Jialian Wu[1]* Sudhir Yarram[1] Hui Liang[2] Tian Lan[2]
Junsong Yuan[1] Jayan Eledath[2] Gerard Medioni[2]

[1]State University of New York at Buffalo [2]Amazon

https://jialianwu.com/projects/EfficientVIS.html

Abstract

Video Instance Segmentation (VIS) aims to simultaneously classify, segment, and track multiple object instances in videos. Recent clip-level VIS takes a short video clip as input each time showing stronger performance than frame-level VIS (tracking-by-segmentation), as more temporal context from multiple frames is utilized. Yet, most clip-level methods are neither end-to-end learnable nor real-time. These limitations are addressed by the recent VIS transformer (VisTR) [25] which performs VIS end-to-end within a clip. However, VisTR suffers from long training time due to its frame-wise dense attention. In addition, VisTR is not fully end-to-end learnable in multiple video clips as it requires a hand-crafted data association to link instance tracklets between successive clips. This paper proposes EfficientVIS, a fully end-to-end framework with efficient training and inference. At the core are tracklet query and tracklet proposal that associate and segment regions-of-interest (RoIs) across space and time by an iterative query-video interaction. We further propose a correspondence learning that makes tracklets linking between clips end-to-end learnable. Compared to VisTR, EfficientVIS requires $15\times$ *fewer training epochs while achieving state-of-the-art accuracy on the YouTube-VIS benchmark. Meanwhile, our method enables whole video instance segmentation in a single end-to-end pass without data association at all.*

1. Introduction

Video Instance Segmentation (VIS) is a challenging video task recently introduced in [28]. It aims to predict a tracklet segmentation mask with a class label for every appeared object instance in a video as illustrated in Fig. 1. Existing methods typically solve the VIS problem at either frame-level or clip-level. The frame-level methods [10, 14, 16, 24, 26, 29] follow a tracking-by-segmentation paradigm, which first performs image instance segmentation and then links the current masks with history tracklets via data association as shown in Fig. 1 (a). This paradigm typically requires complex data association algorithms and exploits limited temporal context, making it susceptible to object occlusions. By contrast, the clip-level methods [2, 3, 15, 25] jointly perform segmentation and tracking clip-by-clip. Within each clip, object information is propagated back and forth. Such a paradigm usually performs stronger than the frame-level methods thanks to larger temporal receptive field. However, most clip-level methods are not end-to-end and require elaborated inference that causes slow speed. These issues are addressed by the recent VIS transformer (VisTR) [25] which extends image object detector DETR [7] to the VIS task. VisTR generates VIS predictions within each video clip in one end-to-end pass, which greatly simplifies the clip-level paradigm and makes VIS within a clip end-to-end trainable.

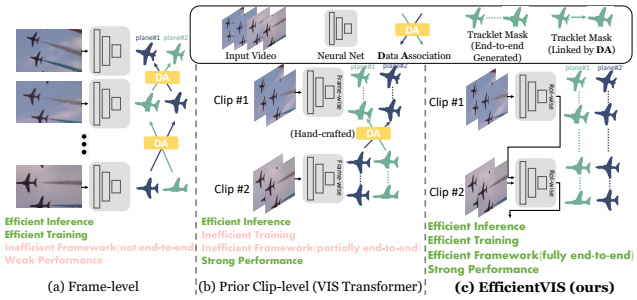

Figure 1. **Overview of real-time VIS pipelines.** Compared to (a) or (b), our method features **1) Efficient training** thanks to the RoI-wise design; **2) Efficient framework** that achieves whole video instance segmentation in a single end-to-end pass without any data association and post-processing; **3) Strong performance** thanks to the clip-by-clip processing and fully end-to-end paradigm.

However, the VIS transformer is confronted with two issues: **(i)** The convergence speed of VisTR is slow since the frame-wise dense attention weights in the transformer need long training epochs to search for properly attended regions among all video frame pixels. It is difficult for researchers to experiment with this algorithm as it requires long development cycles. **(ii)** When a video is too long to fit into GPU memory in one forward-pass, it has to be divided

*Work done during an internship at Amazon

into multiple successive clips for sequential processing. In this case, VisTR becomes partially end-to-end. As shown in Fig. 1 (b), VisTR requires a hand-crafted data association to link tracklets between clips. Such a hand-crafted scheme is not only more complicated but also less effective than an end-to-end learnable association as evidenced in Table 1d.

In this paper, we propose a new clip-level VIS framework, coined as EfficientVIS. EfficientVIS delivers a fully end-to-end framework that can achieve fast convergence and inference with strong performance. Our work is inspired by the recent success of the query-based R-CNN architecture [10, 20] in image object detection. EfficientVIS extends the spirit to the video domain to model the complex space-time interactions. Specifically, EfficientVIS uses *tracklet query* paired with *tracklet proposal* to represent an individual object instance in video. Tracklet query is latent embeddings that encode appearance information for a target instance, while tracklet proposal is a space-time tube that locates the target in the video. Our tracklet query collects the target information by interacting with video in a *clip-by-clip* fashion through a designed *temporal dynamic convolution*. This way enriches temporal object context that is an important cue for handling object occlusion and motion blur in videos. We also design a *factorised temporo-spatial self-attention* allowing tracklet queries to exchange information over not only space but also time. It enables one query to correlate a target across multiple frames so as to end-to-end generate target tracklet mask as a whole in a video clip. Compared to prior works, EfficientVIS enjoys three remarkable properties:

(i) Fast convergence: In EfficientVIS, tracklet queries interact with video CNN features only in the region of the space-time RoIs defined by the tracklet proposal. This is different from VisTR [25] that is interacting with all video pixels on the transformer [22] encoded features using dense attention. Our RoI-wise design drastically reduces video redundancies and therefore allows EfficientVIS for faster convergence than transformer as shown in Table 1g.

(ii) Fully end-to-end learnable framework: EfficientVIS goes beyond a short clip and is fully end-to-end learnable over the whole video. When there are multiple successive clips, one needs to link instance tracklets between clips. In contrast to prior clip-level works [3, 18, 23, 25] that manually stitch the tracklets, we design a *correspondence learning* that enables tracklet query to be shared among clips for *seamlessly* associating a same instance. In other words, the tracklet query output from one clip is enabled to be fed into the next clip to associate and segment the same instance. Meanwhile, the query is dynamically updated in terms of the next clip content so as to achieve a continuous tracking for the future. Such a scheme makes EfficientVIS fully end-to-end, without any explicit data association for either inner-clip or inter-clip tracking as shown in Fig. 1 (c).

(iii) Tracking in low frame rate videos: Tracking object instances that have dramatic movements is a great challenge for motion-based trackers [18, 23, 31], as they suppose instances move smoothly over time. In contrast, our method retrieves a target instance in a frame conditioned on its query representation regardless of where it is in nearby frames. Thus, EfficientVIS is robust to dramatic object movements and can track instances in low frame rate videos as shown in Fig. 5 and Table 1h.

We summarize our major contributions as follows:

- **EfficientVIS is the first RoI-wise clip-level VIS framework that runs in real-time.** The RoI-wise design enables a fast convergence by drastically reducing video redundancies. Fully end-to-end learnable tracking and rich temporal context of the clip-by-clip workflow together bring a strong performance. EfficientVIS ResNet-50 achieves 37.9 AP on Youtube-VIS [28] in 36 FPS by training 33 epochs, which is 15$\times$ training epochs fewer and 2.3 AP higher than VIS transformer.
- **EfficientVIS is the first fully end-to-end neural net for VIS.** Given a video as input despite its length, EfficientVIS directly produces VIS predictions without any data association or post-processing. We will demonstrate by diagnostic experiments that this fully end-to-end paradigm is not only simpler but also more effective than the previous partially/non end-to-end frameworks.

2. Related Works

Frame-level VIS: Most video instance segmentation methods work at the frame-level fashion, *a.k.a.* tracking-by-segmentation [6, 10, 11, 14, 16, 19, 24, 26, 29]. This paradigm produces instance segmentation frame-by-frame and achieves tracking by linking the current instance mask to the history tracklet. So it requires an explicit data association algorithm in either a hand-crafted or trainable way. Some attempts [11, 14, 26, 29] strive to exploit temporal context to improve VIS, yet their temporal context is limited to one or only a few frames ignoring fertile resources of videos. In contrast to the above methods, our method does not require data association algorithm at all and we take advantage of richer temporal context by performing VIS clip-by-clip.

Clip-level VIS: Recent clip-level works [2, 3, 15, 25] demonstrate promising VIS accuracy by exploiting rich temporal knowledge from multiple frames of a video clip. This paradigm propagates object information back and forth in a clip, which can well handle object occlusion and motion blur. However, most clip-level methods have a complex inference process (*e.g.* extra mask refinement, box ensemble, etc) that makes them neither end-to-end learnable nor real-time in inference. To address these limitations, the VIS transformer (VisTR) [25] extends DETR [7] from images to videos, achieving efficient and end-to-end VIS within each clip. Nevertheless, VisTR suffers from slow convergence due to the frame-wise dense attention. Besides, if there are two or

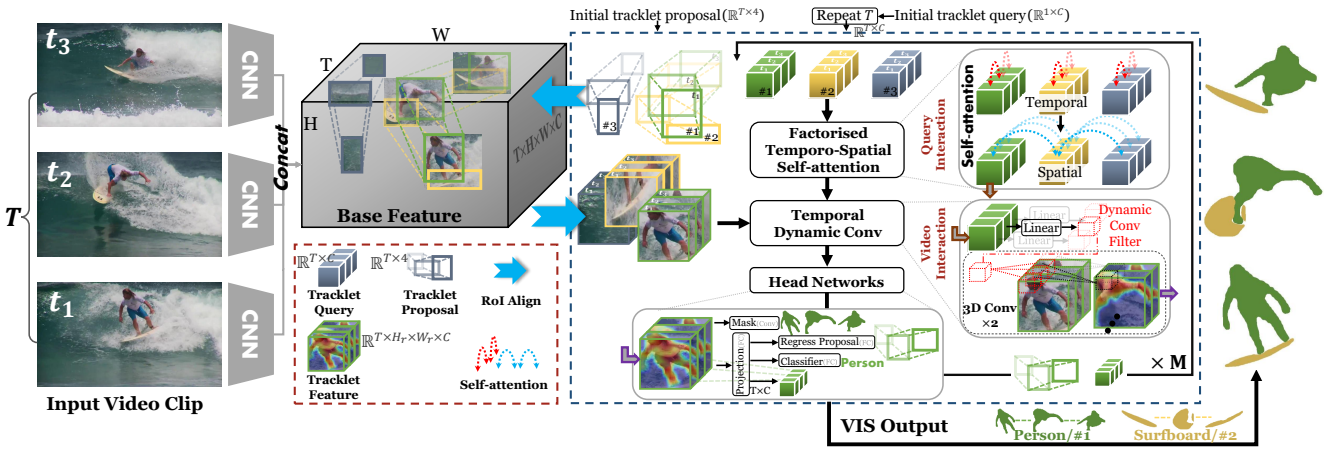

Figure 2. **EfficientVIS architecture.** EfficientVIS performs VIS clip-by-clip where the above figure illustrates how it works in one clip.

more video clips, VisTR is not fully end-to-end and requires a hand-crafted data association to link tracklets between clips. In contrast to the above methods, EfficientVIS is fully end-to-end learnable without the need of data association for either inner- or inter-clips. Moreover, EfficientVIS performs VIS in an RoI-wise manner through an efficient query-video interaction, enabling fast convergence and inference.

3. EfficientVIS

EfficientVIS aims to deliver four key features: **(i)** *Fast training and inference*; **(ii)** *Fully end-to-end learnable framework* without any data association/post-processing; **(iii)** *Tracking in low frame rate videos*; **(iv)** *Strong performance*. EfficientVIS performs VIS clip-by-clip, where we present the main VIS architecture for inner-clip in Sec. 3.1 and the inter-clip tracklets correspondence learning in Sec. 3.3.

3.1. Main Architecture

EfficientVIS is an RoI-wise clip-level VIS framework. In each forward-pass, it takes as input a video clip $\{\boldsymbol{I}_t\}_{t=1}^{T}$ and directly yields the VIS predictions, *i.e.* tracklet masks $\{\boldsymbol{m}_i\}_{i=1}^{N}$ along with tracklet classifications, where $\boldsymbol{I}_t \in \mathbb{R}^{H_I\times W_I\times 3}$ and $\boldsymbol{m}_i \in \mathbb{R}^{T\times H_I\times W_I\times 1}$. T is the number of frames in the clip, and N is the number of unique object instances. $H_I \times W_I$ is the frame spatial size. As detailed in Fig. 2, EfficientVIS starts with a CNN backbone that extracts a video base feature. Afterward, EfficientVIS iterates over the following query-video interaction M times: 1) Query interaction: tracklet queries self-interact and communicate in space and time by our *factorised temporo-spatial self-attention* so that a tracklet query consistently associates the same instance over time and different tracklet queries track different instances; 2) Video interaction: tracklet queries interact with video feature to collect the target instance information by our *temporal dynamic convolution*. At the end of each interaction, a couple of head networks are applied to update tracklet queries, proposals, masks, and classifications. Finally, EfficientVIS takes tracklet masks and classifications output from the last iteration as the VIS results.

Base Feature: We apply a standard CNN backbone frame-by-frame to the input video clip. The extracted features are then concatenated along the time dimension to form the base feature $\boldsymbol{f} \in \mathbb{R}^{T\times H\times W\times C}$, where C is the number of channels and $H \times W$ is the spatial size of feature map. The base feature contains full semantic information of the clip.

Tracklet Query and Proposal: We use tracklet queries $\{\boldsymbol{q}_i\}_{i=1}^{N}$ paired with tracklet proposals $\{\boldsymbol{b}_i\}_{i=1}^{N}$ to represent every unique object instance throughout a video. Tracklet query $\boldsymbol{q}_i$ is embedding vectors that encode exclusive appearance information for the i-th instance, indicating the instance identity. Tracklet proposal is a space-time box tube $\boldsymbol{b}_i \in \mathbb{R}^{T\times 4}$, which tracks and locates the i-th instance across the entire video clip. $\boldsymbol{b}_i^t \in \mathbb{R}^4$ determines the four corners of the box at time t. Each tracklet query has T embeddings, *i.e.* $\boldsymbol{q}_i \in \mathbb{R}^{T\times C}$, where C is the channel number of an embedding. In this way, we enforce different embeddings to focus on different frames. This is more effective than using only one embedding as evidenced in Table 1f, because one instance at different time could have significant appearance changes. It is harsh to enforce a single embedding to encode various appearance patterns of an instance.

Factorised Temporo-Spatial Self-Attention (FTSA): The goal of FTSA is to achieve a query-target correspondence that makes each query associate to a unique target instance in the video. We take advantage of the idea of factorised temporo-spatial self-attention in video transformer [1, 4] which is originally used for encoding video backbone features. Here, we exploit it on instance tracklet queries to achieve an instance communication. Concretely, we separately perform temporal and spatial Multi-Head Self-Attention (MSA) [22] one after the other on tracklet queries as shown in Fig. 2. Let $\boldsymbol{q}_i^t \in \mathbb{R}^{1\times C}$ denote the t-th embedding in the i-th instance tracklet query. The temporal self-attention is computed within the same tracklet query,

across the T embedding pairs as:

$$\{\boldsymbol{q}_i^t\}_{t=1}^T \leftarrow \text{MSA}(\{\boldsymbol{q}_i^t\}_{t=1}^T), \quad i = 1, ..., N. \tag{1}$$

The temporal self-attention allows the embeddings of one instance query from different frames to exchange the target instance information, such that the embeddings can jointly associate the same instance over time. Therefore, instance association/tracking in each clip is **implicitly** achieved and end-to-end **learned** by our temporal self-attention. The spatial self-attention is then computed at the same frame, across the N embedding pairs from different tracklet queries as:

$$\{\boldsymbol{q}_i^t\}_{i=1}^N \leftarrow \text{MSA}(\{\boldsymbol{q}_i^t\}_{i=1}^N), \quad t = 1, ..., T. \tag{2}$$

The spatial self-attention allows each query to acquire object context from other queries in each frame, reasoning about the relations among different object instances. We observe in experiments that the order of temporal and spatial self-attention does not cause a noticeable performance difference.

Joint temporo-spatial self-attention that computes MSA across all $T \times N$ embedding pairs for once is another alternative for query communication. Compared to this scheme, our FTSA saves more computation and is more effective as we will demonstrate in Table 1a.

Temporal Dynamic Convolution (TDC): The aim of TDC is to collect the target instance information from the video clip. Specifically, we generate a dynamic convolutional filter [13, 21] conditioned on tracklet query embedding. We then use it to perform convolution on an RoI region of the base feature specified by the corresponding tracklet proposal. Different from the still-image dynamic convolution [10, 20, 21], we perform a 3D dynamic convolution in order to collect temporal instance context from nearby frames as well. Let $\boldsymbol{w}_i^t$ denote the dynamic convolutional filter generated from tracklet query embedding $\boldsymbol{q}_i^t$. The TDC is computed as:

$$\boldsymbol{o}_i^t = \sum_{t'=t-1}^{t+1} \boldsymbol{a}_{i,(t,t')} \circ \text{conv2d}(\boldsymbol{w}_i^t, \phi(\boldsymbol{f}^{t'}, \boldsymbol{b}_i^{t'})). \tag{3}$$

ϕ denotes RoI align [12] whose output spatial size is $H_r \times W_r$ and $\circ$ is element-wise product. $\boldsymbol{a}_{i,(t,t')} \in \mathbb{R}^{H_r \times W_r}$ is a simple adaptive weight that measures the similarity of conv2d$(\cdot)$ outputs between time step t and t'. Similar to [27, 32], the adaptive weight is calculated by cosine similarity and softmax such that $\sum_{t'=t-1}^{t+1} a_{i,(t,t')}^{x,y} = 1$. $\{\boldsymbol{o}_i^t\}_{t=1}^T \in \mathbb{R}^{T \times H_r \times W_r \times C}$ is the *tracklet feature* of the i-th instance. Eq. 3 differs from regular 3D convolution, where we share the same filter over the time dimension.

Since the dynamic filter $\boldsymbol{w}_i$ is generated from $\boldsymbol{q}_i$, $\boldsymbol{w}_i$ has distinct cues and appearance semantics of the i-th instance. The TDC exploits $\boldsymbol{w}_i$ to exclusively collect the i-th instance information from the video base feature yet filter out irrelevant information. Therefore, tracklet feature $\boldsymbol{o}_i$ shall be highly activated by $\boldsymbol{w}_i$ in the region where the i-th instance appears but will be suppressed in the region of uncorrelated instances or background clutter as evidenced in Fig. 4.

Head Networks: For each instance i, we employ light-weight head networks on its tracklet feature $\boldsymbol{o}_i$ to output its VIS predictions and renewed tracklet query and proposal. Since tracklet feature $\boldsymbol{o}_i$ exclusively carries the i-th instance information across the clip, these outputs can be readily derived by applying regular convolutions or fully connected layers (FCs). Specifically, we apply convolutions to segment tracklet mask, and FCs to classify tracklet, regress tracklet proposal, and update tracklet query. The updated query and proposal are fed into the next iteration. The head network architectures are similar to [10, 12, 20]. In a nutshell, our tracklet (proposal) for a video clip is directly generated as a whole determined by tracklet query. So we do not need explicit data association to link instances/boxes across frames in a clip as frame-level VIS methods [10, 11, 14, 16, 19, 24, 29].

Initial Tracklet Query and Proposal Details: As [10, 20], the initial tracklet query and proposal at the input of the first iteration are model parameters learned from training. Each initial query has one embedding, *i.e.* $\boldsymbol{q}_i^\star \in \mathbb{R}^{1 \times C}$. As shown in Fig. 2, we repeat it T times over the time dimension before starting the first iteration[1]. Besides, we find in experiments that all the learned initial proposals tend to be the frame size which covers the whole scene. We think this is reasonable because it is easier for later stages to regress the proposal to the target region by first taking a glance at the whole image.

3.2. Training

For each video clip, EfficientVIS is trained by the one-to-one matching loss [7, 10, 20, 25], which first obtains a one-to-one assignment between predictions and ground truths by bipartite matching and then computes the training loss in terms of the assignment. Let $\{y_i\}_{i=1}^N$ denote the tracklet predictions and $\{\hat{y}_j\}_{j=1}^G$ denote the tracklet ground truths, where G ($\leq N$) is the total number of ground truths. We denote σ as a G-elements permutation of $\{1, ..., N\}$. The bipartite matching between y and $\hat{y}$ is found by searching for a σ with the lowest cost as:

$$\hat{\sigma} = \arg\min_{\sigma} \sum_{j=1}^{G} \mathcal{L}_{\text{match}} \left(y_{\sigma(j)}, \hat{y}_j\right). \tag{4}$$

The optimal assignment $\hat{\sigma}$ is obtained by the Hungarian algorithm. Each instance tracklet has three types of annotation $\hat{y}_j = (\hat{\boldsymbol{c}}_j, \hat{\boldsymbol{b}}_j, \hat{\boldsymbol{m}}_j)$, where $\hat{\boldsymbol{c}}_j \in \mathbb{R}^T$ is the class label including background class $\emptyset$. $\hat{c}_j^t = \emptyset$ indicates the instance disappears at time step t. Let $p_{\sigma(j)}(\hat{\boldsymbol{c}}_j)$ be the predicted classification score of the $\sigma(j)$-th tracklet for the class $\hat{\boldsymbol{c}}_j$. The matching cost is computed by $\mathcal{L}_{\text{match}} = \sum_{t=1}^T -p_{\sigma(j)}(\hat{c}_j^t) + \mathbb{1}_{\{\hat{c}_j^t \neq \emptyset\}}(\mathcal{L}_{\text{box}}(\boldsymbol{b}_{\sigma(j)}^t, \hat{\boldsymbol{b}}_j^t) - \mathcal{L}_{\text{maskIoU}}(\boldsymbol{m}_{\sigma(j)}^t, \hat{\boldsymbol{m}}_j^t))$, where $\boldsymbol{b}$ and $\boldsymbol{m}$ are the predicted tracklet proposal and mask respectively. $\mathcal{L}_{\text{box}}$ is calculated by box IoU and L1 distance

[1] We repeat the parameter $\boldsymbol{q}^\star$ T times rather than directly setting the parameter to be $\boldsymbol{q}^\star \in \mathbb{R}^{T \times C}$, because it gives an initial prior that all the T embeddings shall belong to a same instance.

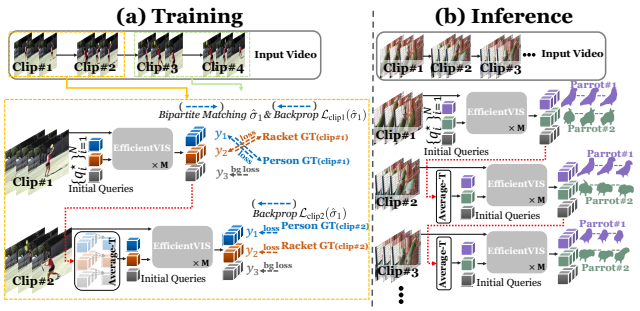

Figure 3. **(a) Correspondence learning**. **(b) Fully end-to-end inference in multiple clips**.

as [7]. After finding the optimal assignment $\hat{\sigma}$ that one-to-one matches between tracklet predictions and ground truths, the training loss for the video clip is computed as:

$$\mathcal{L}_{\text{clip}}(\hat{\sigma}) = \sum_{j=1}^{G}\sum_{t=1}^{T}[-p_{\hat{\sigma}(j)}(\hat{c}_j^t) + \mathcal{L}_{\text{CE}}(\boldsymbol{m}_{\hat{\sigma}(j)}^t, \hat{\boldsymbol{m}}_j^t) + \mathbb{1}_{\{\hat{c}_j^t \neq \emptyset\}}(\mathcal{L}_{\text{box}}(\boldsymbol{b}_{\hat{\sigma}(j)}^t, \hat{\boldsymbol{b}}_j^t) + \mathcal{L}_{\text{dice}}(\boldsymbol{m}_{\hat{\sigma}(j)}^t, \hat{\boldsymbol{m}}_j^t))], \tag{5}$$

where $\mathcal{L}_{\text{CE}}$ is the binary cross-entropy loss and $\mathcal{L}_{\text{dice}}$ is the dice loss [17]. All the losses and balancing weights are similar to [7, 10, 20, 25], except that we add a $\mathcal{L}_{\text{CE}}$ to enforce predicted mask to be zero if the instance disappears. Eq. 5 is averaged by the total number of entries. For those unmatched tracklet predictions, we only impose classification loss to supervise them to be background class.

3.3. Correspondence Learning

If a long video cannot fit into GPU memory in one forward-pass, it needs to be split into multiple successive clips for sequential processing. To stitch tracklets between two clips, prior works [3, 18, 23, 25] turn to hand-crafted data association. However, human-designed rules are cumbersome and not end-to-end learnable. Therefore, we design a correspondence learning that makes the tracklet linking between clips end-to-end learnable without data association, which is not only simpler but also more effective.

Correspondence learning: As shown in Fig. 3, every two clips from the input video are paired[2] in training. For the first clip of a pair, the initial tracklet queries are the model parameters $\boldsymbol{q}^\star$ as described in Sec. 3.1, and the training procedure is the same as Sec. 3.2, *i.e*. finding the optimal assignment $\hat{\sigma}$ and backpropagating $\mathcal{L}_{\text{clip}}(\hat{\sigma})$. For the initial queries of the second clip, we use the output tracklet queries from the first clip averaged along the time dimension. The average operation can extract a comprehensive abstract for the instance representation. As for the assignment of the second clip, we do not perform bipartite matching. Instead, we use the same assignment $\hat{\sigma}$ that is already obtained by the first clip to train the second clip. Such a training manner enforces the tracklet queries output from one clip to segment and correspond to the same object instances in the next clip. In this way, tracklet query is able to serve as a generic video-level representation to *seamlessly* associate and correlate an instance across a whole video instead of only a clip.

Inference: EfficientVIS simply takes output queries from the last clip as the initial queries to correlate and segment the same instances in the current clip without data association. Meanwhile, the queries are continuously updated by our query-video interaction to collect the current clip information so as to achieve a continuous tracking for the future. If a tracklet query has been classified as background for many clips, one may re-initialize it with $\boldsymbol{q}^\star$ in order to allocate slots. Since instance positions vary in different clips, we use the same initial proposals for every clip, *i.e*. the trained model parameters rather than the last clip output proposals.

4. Experiments

4.1. Experimental Setup

Datasets: Our experiments are conducted on the YouTube-VIS [28] benchmark in two versions. YouTube-VIS 2019 is the first dataset for video instance segmentation, which contains 2,883 videos labeled at 6 FPS, 131K instance masks, and 40 object classes. YouTube-VIS 2021 is an increased version that contains 3859 videos.

Evaluation Metric: The video-level average precision (AP) and average recall (AR) are the metrics. Different from the image domain, the video intersection over union (IoU) is computed between the predicted tracklet mask and ground truth tracklet mask. So, in order to achieve high performance, a model needs to not only correctly classify and segment the target instance but also accurately track the target over time.

Architecture Settings: Following [25], the default video clip length and the number of tracklet queries are set to $T = 36$ and $N = 10$, respectively. Following [10, 20, 25], the number of iterations is $M = 6$ and the default CNN backbone is ResNet-50. We use the above default settings for all experiments unless otherwise specified. Image frame size and pretrained model are the same as [6, 10, 25, 29].

Training: EfficientVIS is trained by AdamW with an initial learning rate of 2.5×10^{-5}. We train the model for 33 epochs where the learning rate is dropped by a factor of 10 at the 27-th epoch. For example, the model can be trained in 12 hours with 4 RTX 3090 GPUs on YouTube-VIS 2019. The correspondence learning is enabled if the maximum frame number of videos in a dataset is larger than T. No training data augmentation is used unless otherwise specified.

Fully end-to-end Inference: EfficientVIS does not include any data association or post-processing. **Reason:** Target tracklet **within** a clip is generated as a whole in a single forward-pass thanks to the implicit association of our FTSA (Sec. 3.1). Target tracklets **between** two clips are automatically correlated owing to our correspondence learning

[2]The order of the two clips in a pair is randomized.

Self-attention	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Spatial (S)	32.8	56.3	34.9	36.1	42.2
Temporal (T)	30.4	49.0	32.1	34.2	39.1
Joint T-S	33.7	56.8	36.1	36.6	44.6
Factorised T-S (FTSA)	**37.0**	59.6	40.0	39.3	46.3

(a) **Self-attention schemes.** We perform different multi-head self-attention schemes on tracklet queries.

Length	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
$T = 9$	35.3	57.6	38.5	37.7	43.5
$T = 18$	36.4	58.1	39.5	39.1	46.8
$T = 36$	**37.0**	59.6	40.0	39.3	46.3

(c) **Video clip length T.** We experiment with different number of frames for each video clip. Larger temporal receptive filed provides richer temporal context and therefore yields better performance.

Scheme	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
w/o CL	7.4	19.0	5.9	9.8	13.5
w/ CL	**35.3**	57.6	38.5	37.7	43.5

(e) **Correspondence learning (CL).** We train EfficientVIS with and without CL. After training, we test both using our fully end-to-end inference paradigm. $T = 9$ in this study.

Method	Train Aug.	Epopchs	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
VisTR [25]	random crop	∼500	35.6	56.8	37.0	35.2	40.2
EfficientVIS	✗	33	**37.0**	59.6	40.0	39.3	46.3
EfficientVIS	multi-scale	33	**37.9**	59.7	43.0	40.3	46.6

(g) **Convergence speed. (EfficientVIS *vs.* VIS Transformer).** $T = 36$ for both VisTR and EfficientVIS. VisTR is equipped with random cropping training augmentation by default.

Dynamic Conv	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Still-image	36.0	59.5	39.5	38.4	45.3
Temporal	**37.0**	59.6	40.0	39.3	46.3

(b) **Still-image *vs.* Temporal - dynamic convolution**. Temporal dynamic convolution is more effective by taking into account temporal object context from nearby frames.

Scheme		AP	AP_{50}	AP_{75}	AR_1	AR_{10}
$T = 9$	Hand-craft	33.7(−1.6)	55.5	36.4	33.9	40.3
	Fully e2e (ours)	**35.3**	57.6	38.5	37.7	43.5
$T = 18$	VisTR [25]	29.7(−6.7)	50.4	31.1	29.5	34.4
	Hand-craft	34.6(−1.8)	55.3	37.4	36.6	44.6
	Fully e2e (ours)	**36.4**	58.1	39.5	39.1	46.8

(d) **Fully end-to-end (e2e) *vs.* Partially e2e**. Both "Hand-craft" and VisTR are partially e2e, where tracklet association within each clip is e2e but that between clips requires a hand-crafted linking. For "Hand-craft", we report the best results by varying matching score thresholds.

Query	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Time shared	35.5	57.1	38.5	38.7	43.9
Time disentangled	**37.0**	59.6	40.0	39.3	46.3

(f) **Time disentangled *vs.* Time shared - query.** For each tracklet query, the time disentangled scheme uses T embeddings, while the time shared scheme only uses one embedding.

Video Frame Rate	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Original FPS	**35.3**	57.6	38.5	37.7	43.5
1.5 FPS	**35.3**	57.4	39.1	37.5	42.8

(h) **Tracking in low frame rate videos ($T = 9$).** We downsample the frame rate of the original YouTube-VIS videos to 1.5 FPS. EfficientVIS is not affected by low video frame rate or dramatic object motions.

Table 1. **Ablation studies** on the YouTube-VIS 2019 `val` set.

(Sec. 3.3). The one-to-one matching loss (Sec. 3.2) further enables our method to get rid of post-processing like NMS.

4.2. Ablation Studies

Time Disentangled Query: As described in Sec. 3.1, our tracklet query is designed to be disentangled in time, *i.e.* each query contains T embeddings instead of one embedding. As shown in Table 1f, we compare it with the time shared query scheme that only uses one single embedding for each tracklet query. Our time disentangled query achieves 1.5 AP higher than the time shared one. We argue that this is because objects in different time may have dramatic appearance changes like heavy occlusion or motion blur. It is more reasonable to use multiple embeddings to separately encode different appearance patterns of an instance.

Factorised Temporo-Spatial Self-Attention: To evaluate our factorised temporo-spatial self-attention (FTSA), we experiment with different self-attention schemes as shown in Table 1a. "S" indicates the spatial self-attention that only performs multi-head self-attention (MSA) within the same frame across the embeddings of different tracklet queries. "T" denotes the temporal self-attention that only performs MSA within the same tracklet query across the embeddings of different time. "Joint T-S" is the joint temporo-spatial self-attention that performs MSA across all $T \times N$ embeddings as described in Sec. 3.1. We see from the table that the FTSA and "Joint T-S" achieve better performance than using either "S" or "T" standalone. This is because the temporal self-attention can let a query in different time exchange information so as to associate the same instance, while the spatial self-attention models different instances relations in space. Such a query communication in both space and time is necessary for the VIS task. Compared to "Joint T-S", our FTSA yields better VIS results. We think the reason is that all attendees in FTSA are strongly related to each other. For example, the embeddings in temporal self-attention are all from the same tracklet query, while those in spatial self-attention are all from the same frame. However, by mixing up all $T \times N$ embeddings in a single self-attention pass, much less relevant information from other tracklets at far temporal positions is directly involved during "Joint T-S".

Temporal Dynamic Convolution: To assess our temporal dynamic convolution (TDC), we compare it with the still-image dynamic convolution that performs 2D convolution on the current frame only. As shown in Table 1b, our temporal dynamic convolution improves 1 AP over the still-image version. It suggests that the temporal object context from nearby frames is also an informative cue. Moreover, we visualize the tracklet feature. As shown in Fig. 4, tracklet feature is highly activated by our TDC in the region where the target instance appears. We also see that tracklet fea-

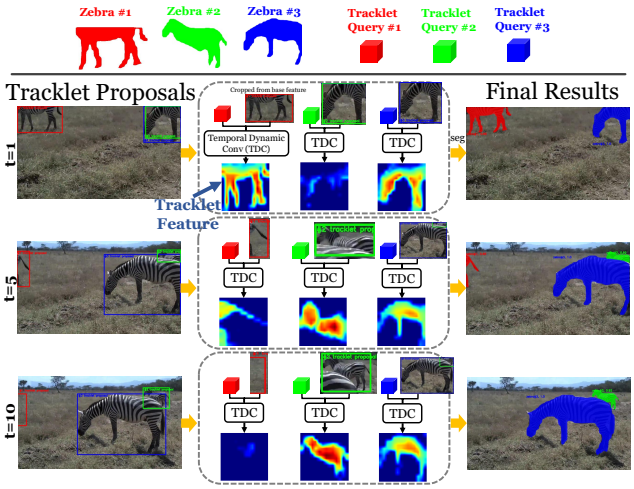

Figure 4. **Tracklet features carry exclusive target instance information.** The dynamic conv filter conditioned on tracklet query activates tracklet feature only in the region where the target instance appears. The #2 tracklet feature at t=1 is suppressed by the filter of **#2 zebra** query even the proposal has drifted to the **#3 zebra**.

ture is suppressed when the target instance disappears, even if its tracklet proposal has drifted to background clutter or uncorrelated instance. This demonstrates the dynamic filter conditioned on tracklet query is target-specific, which exclusively collects the query's target information from the video base feature yet discards irrelevant information.

Video Clip Length: As shown in Table 1c, we experiment with different video clip lengths. EfficientVIS improves 1.7 AP by increasing the number of frames from 9 to 36. It is reasonable because more frames can provide richer temporal context which is important to video tasks. Moreover, since our method runs clip-by-clip in 36 FPS as shown in Table 2, it can be regarded as a near-online fashion with a delay of $\frac{T}{36}$. For example, there is a 0.25s delay when $T = 9$.

Correspondence Learning: To validate the effectiveness of our correspondence learning, we train an EfficientVIS without the correspondence learning. Concretely, we treat every clip as an individual video and do not use tracklet queries from other clips as input during training. In inference, we keep our fully end-to-end paradigm, *i.e*. the output tracklet queries from one clip are used as the initial queries of the next clip. As shown in Table 1e, the performance significantly drops if EfficientVIS is trained without the correspondence learning. It suggests that our correspondence learning is the key to enabling tracklet query to be a video-level instance representation that can be shared among different video clips.

Fully End-to-end Learnable Framework: Thanks to the correspondence learning, EfficientVIS fully end-to-end performs VIS by taking tracklet queries from one clip as the initial queries of the next clip. To evaluate this fully end-to-end framework, we compare it with two partially end-to-end schemes, "hand-craft" and VisTR [25] in Table 1d. These two methods perform VIS end-to-end within each clip but require a hand-crafted data association to link tracklets between clips. For the "hand-craft" scheme, all model architectures keep the same as EfficientVIS except that tracklets between two clips are linked by a human-designed rule: 1) We first construct an affinity matrix by taking into account box IoU, box L1 distance, mask IoU, and class label matching like prior tracking works [26, 31]; 2) We then solve the tracklet association by the Hungarian algorithm. As shown in Table 1d, our fully end-to-end scheme performs better than the partially end-to-end framework in different T settings while greatly simplifying the previous VIS paradigms. The reason for this performance gap is that the tracklet linking between clips in hand-crafted scheme tends to be sub-optimal, while that in our fully end-to-end framework is learned and optimized by ground truths during training.

Tracking in Low Frame Rate Videos: Tracking object instances in low frame rate videos is a hard problem, where motion-based trackers usually fail. Motion-based trackers suppose instances move smoothly over time and impose spatial movement constraints to prevent trackers from associating very distant instances. In contrast, our tracking is determined by instance appearance rather than motion, as we retrieve a target instance solely conditioned on its query representation regardless of the target spatial distance among different frames. Therefore, our tracking is not limited by dramatic instance movements. To demonstrate this property, we downsample the frame rate of the YouTube-VIS dataset to 1.5 FPS and test our method. As shown in Table 1h, EfficientVIS successfully maintains its VIS performance in the low frame rate video setting. As shown in Fig. 5, we also visualize the results of EfficientVIS on a very low frame rate video whose frame is sampled every 5 seconds, *i.e*. 0.2 FPS.

Fast Convergence: In our method, we crop video using tracklet proposal and collect target information from the proposal. Compared to VIS transformer (VisTR) [25] that uses frame-wise dense attention, this RoI-wise design eliminates much background clutter and redundancies in videos and enforces EfficientVIS to focus more on informative regions, making our convergence 15× faster as shown in Table 1g. This RoI-wise pipeline also leads to better accuracy than VisTR. This is because tracklet proposal can avoid regions outside the target being segmented (second figure in appendix), which results in more precise masks and significantly higher AP_{75} as evidenced in Table 1g.

4.3. Comparison to State of the Art

YouTube-VIS 2019: In Table 2, we compare EfficientVIS with the real-time state-of-the-art VIS models. EfficientVIS is the only fully end-to-end framework while achieving superior accuracy. We attribute the strong performance to two main aspects: 1) Our object tracking is learned via the end-to-end framework; 2) Our clip-by-clip processing and

Method	Publication	Augmentations	Backbone	FPS	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
MaskTrack R-CNN [28]	ICCV'19	✗	ResNet-50	33	30.3	51.1	32.6	31.0	35.5
SipMask [6]	ECCV'20	✗	ResNet-50	34	32.5	53.0	33.3	33.5	38.9
CompFeat [11]	AAAI'21	✗	ResNet-50	<33	35.3	56.0	38.6	33.1	40.3
TraDeS [26]	CVPR'21	✗	ResNet-50	26	32.6	52.6	32.8	29.1	36.6
QueryInst [10]	ICCV'21	✗	ResNet-50	32	34.6	55.8	36.5	35.4	42.4
CrossVIS [29]	ICCV'21	✗	ResNet-50	40	34.8	54.6	37.9	34.0	39.0
VisSTG [24]	ICCV'21	✗	ResNet-50	22	35.2	55.7	38.0	33.6	38.5
EfficientVIS (Ours)		✗	ResNet-50	36	**37.0**	**59.6**	**40.0**	**39.3**	**46.3**
STMask [14]	CVPR'21	DCN backbone [9]	ResNet-50	29	33.5	52.1	36.9	31.1	39.2
SG-Net [16]	CVPR'21	multi-scale training	ResNet-50	23	34.8	56.1	36.8	35.8	40.8
VisTR [25]	CVPR'21	random crop training	ResNet-50	30	35.6	56.8	37.0	35.2	40.2
QueryInst [10]	ICCV'21	multi-scale training	ResNet-50	32	36.2	56.7	39.7	36.1	42.9
CrossVIS [29]	ICCV'21	multi-scale training	ResNet-50	40	36.3	56.8	38.9	35.6	40.7
VisSTG [24]	ICCV'21	multi-scale training	ResNet-50	22	36.5	58.6	39.0	35.5	40.8
EfficientVIS (Ours)		multi-scale training	ResNet-50	36	**37.9**	**59.7**	**43.0**	**40.3**	**46.6**
MaskTrack R-CNN [28]	ICCV'19	✗	ResNet-101	29	31.9	53.7	32.3	32.5	37.7
SRNet [30]	ACMMM'21	✗	ResNet-101	35	32.3	50.2	34.8	32.3	40.1
CrossVIS [29]	ICCV'21	✗	ResNet-101	36	36.6	57.3	39.7	36.0	42.0
EfficientVIS (Ours)		✗	ResNet-101	32	**38.7**	**61.3**	**44.0**	**40.6**	**47.7**
SipMask [6]	ECCV'20	multi-scale training	ResNet-101	24	35.8	56.0	39.0	35.4	42.4
STMask [14]	CVPR'21	DCN backbone [9]	ResNet-101	23	36.8	56.8	38.0	34.8	41.8
SG-Net [16]	CVPR'21	multi-scale training	ResNet-101	20	36.3	57.1	39.6	35.9	43.0
VisTR [25]	CVPR'21	random crop training	ResNet-101	28	38.6	61.3	42.3	37.6	44.2
EfficientVIS (Ours)		multi-scale training	ResNet-101	32	**39.8**	**61.8**	**44.7**	**42.1**	**49.8**

Table 2. **Comparison with real-time state-of-the-art methods on the YouTube-VIS 2019 `val` set.**

Figure 5. **Video instance segmentation in a 0.2 FPS video.** EfficientVIS successfully tracks the instances with dramatic movements.

Method	Publication	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
MaskTrack R-CNN [28]	ICCV'19	28.6	48.9	29.6	26.5	33.8
SipMask⋆ [6]	ECCV'20	31.7	52.5	34.0	30.8	37.8
CrossVIS [29]	ICCV'21	33.3	53.8	37.0	30.1	37.6
EfficientVIS (Ours)		**34.0**	**57.5**	**37.3**	**33.8**	**42.5**

Table 3. **Comparison with real-time state-of-the-art methods on the YouTube-VIS 2021 `val` set.** ⋆ denotes multi-scale training.

temporal dynamic convolution enrich temporal object context that is helpful for handling occlusion or motion blur. As shown in Table 2, we *achieve notably higher recall* (AR) than other competitors. We think the reason for the high recall is that we recognize more heavily occluded or blurred objects, which are usually missed by common methods.

For a fair comparison, Table 2 only lists real-time methods using single model without extra data. There are two works with 40+ AP, MaskProp [3] (<5.6 FPS) and Propose-Reduce [15] (1.8 FPS), which however are not real-time and adopt multiple models or extra training data. MaskProp uses DCN backbone [9], HTC detector [8], extra High-Resolution Mask Refinement network, etc. Propose-Reduce is trained with additional DAVIS-UVOS [5] and COCO pseudo videos datasets. These elaborated implementations are beyond the scope of our work, as our main goal is to present a simple end-to-end model with real-time inference and fast training.

YouTube-VIS 2021: We experiment with EfficientVIS on YouTube-VIS 2021 in Table 3. EfficientVIS achieves state-of-the-art AP without bells and whistles. Similar to the findings on YouTube-VIS 2019, EfficientVIS significantly improves AR over other state of the arts.

5. Conclusion

This work presents a new VIS model, EfficientVIS, which simultaneously classifies, segments, and tracks multiple object instances in a single end-to-end pass and clip-by-clip fashion. EfficientVIS adopts tracklet query and proposal to respectively represent instance appearance and position in videos. An efficient query-video interaction is proposed for associating and segmenting instances in each clip. A correspondence learning is designed to correlate instance tracklets between clips without data association. The above designs enable a fully end-to-end framework achieving state-of-the-art VIS accuracy with fast training and inference.

Acknowledgement. This material is in part based upon work supported by a gift grant from Amazon Go and National Science Foundation Grant CNS1951952.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, October 2021. 3

[2] Ali Athar, Sabarinath Mahadevan, Aljosa Osep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *European Conference on Computer Vision*, pages 158–177. Springer, 2020. 1, 2

[3] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9739–9748, 2020. 1, 2, 5, 8

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021. 3

[5] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv preprint arXiv:1905.00737*, 2019. 8

[6] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 5, 8

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2, 4, 5

[8] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 8

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 8

[10] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6910–6919, 2021. 1, 2, 4, 5, 8

[11] Yang Fu, Linjie Yang, Ding Liu, Thomas S Huang, and Humphrey Shi. Compfeat: Comprehensive feature aggregation for video instance segmentation. *arXiv preprint arXiv:2012.03400*, 2020. 2, 4, 8

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 4

[13] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016. 4

[14] Minghan Li, Shuai Li, Lida Li, and Lei Zhang. Spatial feature calibration and temporal fusion for effective one-stage video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11215–11224, 2021. 1, 2, 4, 8

[15] Huaijia Lin, Ruizheng Wu, Shu Liu, Jiangbo Lu, and Jiaya Jia. Video instance segmentation with a propose-reduce paradigm. *arXiv preprint arXiv:2103.13746*, 2021. 1, 2, 8

[16] Dongfang Liu, Yiming Cui, Wenbo Tan, and Yingjie Chen. Sg-net: Spatial granularity network for one-stage video instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9816–9825, 2021. 1, 2, 4, 8

[17] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016. 5

[18] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6308–6318, 2020. 2, 5

[19] Zheyun Qin, Xiankai Lu, Xiushan Nie, Xiantong Zhen, and Yilong Yin. Learning hierarchical embedding for video instance segmentation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1884–1892, 2021. 2, 4

[20] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14454–14463, 2021. 2, 4, 5

[21] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 282–298. Springer, 2020. 4

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3

[23] Manchen Wang, Joseph Tighe, and Davide Modolo. Combining detection and tracking for human pose estimation in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 5

[24] Tao Wang, Ning Xu, Kean Chen, and Weiyao Lin. End-to-end video instance segmentation via spatial-temporal graph neural networks. In *Proceedings of the IEEE/CVF International*

Conference on Computer Vision, pages 10797–10806, 2021. 1, 2, 4, 8

[25] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021. 1, 2, 4, 5, 6, 7, 8, 11, 12

[26] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12352–12361, 2021. 1, 2, 7, 8

[27] Jialian Wu, Chunluan Zhou, Ming Yang, Qian Zhang, Yuan Li, and Junsong Yuan. Temporal-context enhanced detection of heavily occluded pedestrians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13430–13439, 2020. 4

[28] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019. 1, 2, 5, 8

[29] Shusheng Yang, Yuxin Fang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Crossover learning for fast online video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8043–8052, 2021. 1, 2, 4, 5, 8

[30] Xiaowen Ying, Xin Li, and Mooi Choo Chuah. Srnet: Spatial relation network for efficient single-stage instance segmentation in videos. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 347–356, 2021. 8

[31] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020. 2, 7

[32] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017. 4

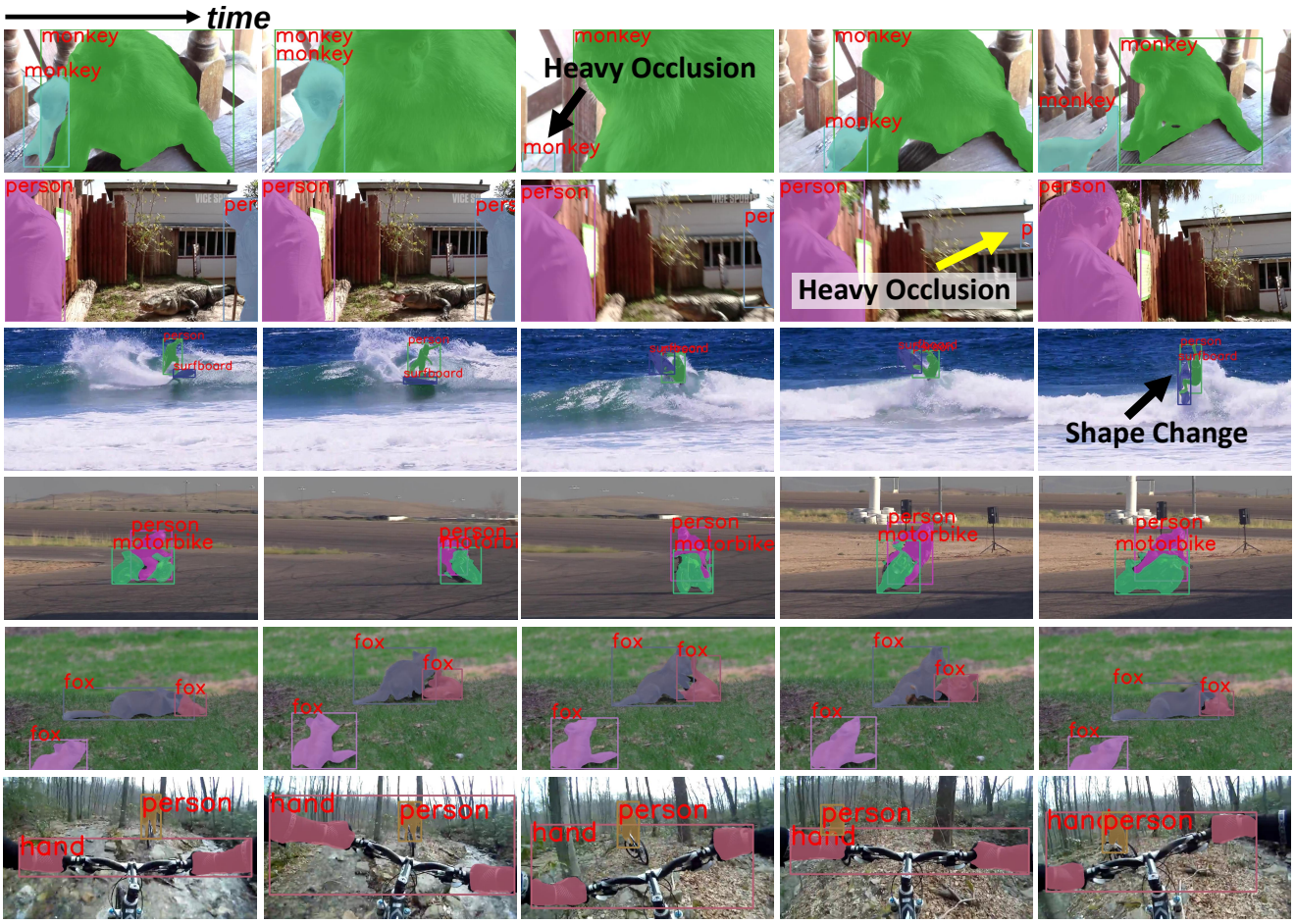

Figure 6. **Visualization on the YouTube-VIS dataset.** One color indicates one identity. The boxes denote our tracklet proposals.

Appendix

A. Qualitative Results on YouTube-VIS

As shown in Fig. 6, we visualize our video instance segmentation results on the YouTube-VIS 2019 dataset. We see from the figure that EfficientVIS can well recognize heavily occluded object instances. The reason is that the target instance information is propagated back and forth in a clip thanks to our clip-by-clip processing and *temporal dynamic convolution*. In this way, the non-occluded instance appearances from nearby frames are propagated to provide strong cues to recognize those heavily occluded instances. We also see in the figure that our tracklet proposal can successfully track target instance even its shape dramatically changes over time. This is because tracklet proposal is regressed conditioned on the target query representation, and we do not impose space-time constraints or smoothness. Thus, tracklet proposal is not limited by the target positions or shapes in nearby frames.

B. Qualitative Comparison

As shown in Fig. 7, we compare EfficientVIS with the VIS transformer (VisTR) [25]. Since VisTR produces an instance mask by segmenting the whole frame, one instance mask may easily contaminate other instances or background regions as shown in the figure. This suggests that it is hard to enforce the query representation in VisTR to be very discriminative to distinguish target object instance from the whole scene. In contrast to this frame-wise scheme, EfficientVIS filters out many irrelevant instances and regions by our tracklet proposals. Our method only needs to enforce tracklet query to distinguish target instance from the proposal region, which is easier for the model to achieve.

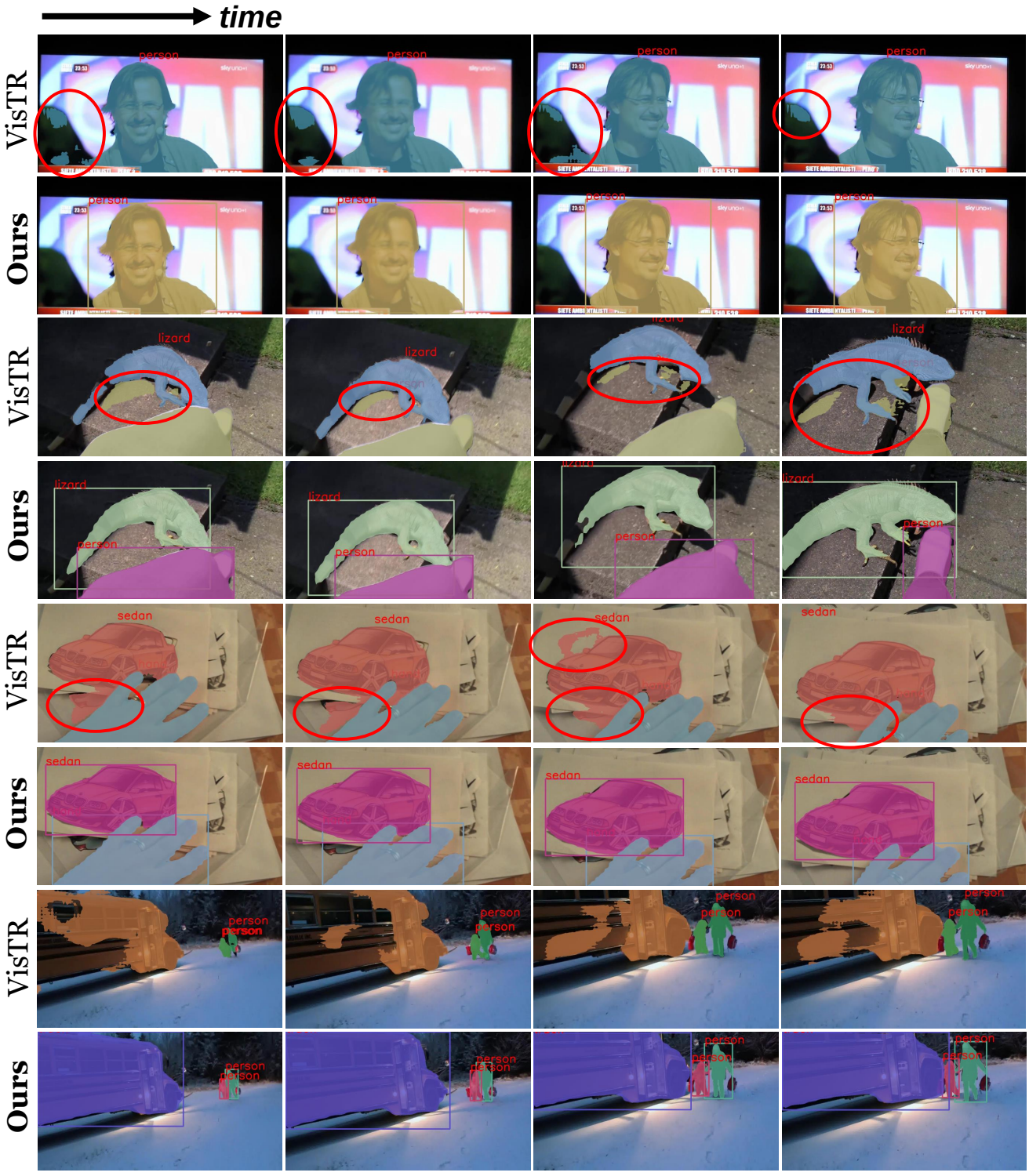

Figure 7. **Qualitative comparison** with the VIS transformer (VisTR) [25].