# Video Object Segmentation with Dynamic Memory Networks and Adaptive Object Alignment

Shuxian Liang[1, 2]*, Xu Shen[2], Jianqiang Huang[2], Xian-Sheng Hua[2]†
[1] State Key Lab of CAD&CG, Zhejiang University, [2] DAMO Academy, Alibaba Group
shuxian.lsx@zju.edu.cn, {shenxu.sx, jianqiang.hjq, xiansheng.hxs}@alibaba-inc.com

## Abstract

*In this paper, we propose a novel solution for object-matching based semi-supervised video object segmentation, where the target object masks in the first frame are provided. Existing object-matching based methods focus on the matching between the raw object features of the current frame and the first/previous frames. However, two issues are still not solved by these object-matching based methods. As the appearance of the video object changes drastically over time, 1) unseen parts/details of the object present in the current frame, resulting in incomplete annotation in the first annotated frame (*e.g. *view/scale changes). 2) even for the seen parts/details of the object in the current frame, their positions change relatively (*e.g. *pose changes/camera motion), leading to a misalignment for the object matching. To obtain the complete information of the target object, we propose a novel object-based dynamic memory network that exploits visual contents of all the past frames. To solve the misalignment problem caused by position changes of visual contents, we propose an adaptive object alignment module by incorporating a region translation function that aligns object proposals towards templates in the feature space. Our method achieves state-of-the-art results on latest benchmark datasets DAVIS 2017 ($\mathcal{J}$ of 81.4% and $\mathcal{F}$ of 87.5% on the validation set) and YouTube-VOS (the overall score of 82.7% on the validation set) with a very efficient inference time (0.16 second/frame on DAVIS 2017 validation set). Code is available at: https://github.com/liang4sx/DMN-AOA.*

## 1. Introduction

Semi-supervised video object segmentation (VOS) is a task that distinguishes target objects from their background at the pixel level in a video, where ground-truth masks of

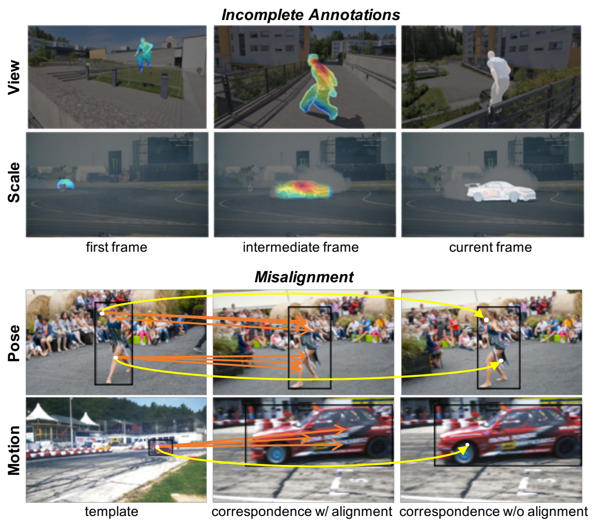

Figure 1. Two issues for the object matching based VOS. In the top two cases, corresponding parts (in red) of the query object do not present in the first frame because of view/scale changes. The bottom two cases show the misalignment problem caused by pose changes and camera motion. Our method solve these issues by exploiting object-specific memories and aligning objects adaptively.

objects are provided in the first frame. The task is then to predict the segmentation masks from the rest video frames. One of the main challenges of VOS is that the appearances of the target object can change drastically across frames due to object movements, camera movements and occlusions. For the semi-supervised video object segmentation community, the pixel matching based VOS (PVOS) and the object matching based VOS (OVOS) are investigated in parallel as two research topics. PVOS methods predict based on cross-frame pixel correlations (*e.g.* [7, 20, 28, 35]). OVOS methods predict by correlating proposal objects of the current frame with template objects of historical frames (*e.g.* [4, 11, 29, 36]).

Existing OVOS methods focus on the matching between the raw object features of the proposals in the current frame and the templates in the first/previous frames

*This work was done when the author was visiting Alibaba as a research intern.
†Corresponding author.

[2, 4, 11, 26, 29, 36]. However, two issues are still not solved by these methods (shown in Figure 1). Specifically, as the appearance of the video object changes drastically over time, **first**, unseen parts/details of the object present in the current frame, resulting in incomplete information in the first frame (*e.g.* view/scale changes). For example, in the top two cases, the information about the person's back and the details of the vehicle's side (which are required by the current frame) are not provided in the first frame. And **second**, for the seen parts/details of the object in the current frame, their positions change relatively (*e.g.* pose/camera motion), leading to a misalignment for the object matching. For example, in the bottom two cases, some object parts in the first frame spatially correspond to the background (the dancer's hand *vs.* the audience's head) or to other dissimilar parts (the vehicle's body vs. the vehicle's wheel) in the current frame. Consequently, existing OVOS methods are not as strong as the state-of-the-art (SOTA) PVOS methods, especially the PVOS methods exploiting all past frames with memory networks [12, 13, 16, 20, 24].

The key challenges for OVOS to exploit all past frames and surpass PVOS lie in two folds: a) The memory module for PVOS is straight-forward, where directly storing feature maps of raw frames is enough. While for OVOS, we need a special memory module to store specific object features, which is still not solved in OVOS (no memory is used in existing OVOS methods [4, 11, 29, 36]). b) Instead of directly computing pixel-to-pixel similarities between frames as in PVOS, the matching of object features between templates and proposals is faced with the misalignment problem caused by object deformation across frames, which is still not solved by OVOS. Specifically, [4] aligns the object sizes rather than the deformed object appearances. Other works compute the object similarities by unaligned pixel-to-pixel distance [29] or global average pooled features without spatial information [4, 36].

To solve the aforementioned two challenges, we propose a novel dynamic memory network and a new adaptive object alignment module for OVOS. By reading relevant object information from all available resources, memory of object features in past frames is used to generate dynamic object templates. In this way, the past frames with object masks form a dynamic memory, and the current frame as the query is used to decode templates that represent the current appearances of objects. With the dynamic memory network, there is no restriction on the number of frames to use and new object information of a given frame can be easily accumulated in the memory. To solve the misalignment problem caused by position changes of visual contents in the target object, we propose an adaptive object alignment operation by incorporating a non-local region translation function that recomposes regions of the object templates based on the object features of proposals. Specifically, object templates and proposals are firstly projected to a shared feature space, then dense correspondences between them are utilized to translate object proposals towards the templates in the feature space.

Equipped with the proposed dynamic memory network and the adaptive object alignment module, we design an easy-to-extend framework for OVOS. Firstly, for a test frame, object proposals are generated through a pre-trained instance segmentation model. Secondly, current-frame features and object bounding boxes from the previous frame are combined as a query input to the dynamic memory network, which generates object templates for the frame incorporating object information from memory frames. Thirdly, the object matching assignment between the generated templates and proposals is produced based on the adaptive object alignment module and a differentiable matching layer [36]. Finally, the object matching results are input to a mask refinement network to make output segmentation.

Our contributions can be summarized as follows:

- We present a novel and easy-to-extend object-matching framework for the VOS task. Our proposed OVOS model outperforms all SOTA PVOS methods for the first time in the community, and we pave a new way for the development of OVOS.

- We are the first to exploit all the frames in the video for OVOS. Specifically, we introduce a dynamic memory network that computes spatio-temporal attention on object features of all past frames for each query frame, to obtain current-frame representations of target objects.

- We propose a novel adaptive object alignment module to solve the misalignment between the object proposals and templates. In detail, we design a region translation function that recomposes template-like regions with proposal object features.

## 2. Related Works

For the semi-supervised video object segmentation community, the pixel matching based VOS (PVOS) and the object matching based VOS (OVOS) are investigated in parallel as two research topics. In this section, the classic and SOTA methods of both topics are introduced. Extra discussions about the methods without matching are in our supplementary material.

**Pixel matching based VOS (PVOS).** PVOS methods provide segmentation clues based on cross-frame pixel correlations. Classic PVOS methods [3, 7, 25, 28] match current-frame pixels with the pixels from the first/previous frames. STM [20] extends the pixel matching with intermediate frames by incorporating a key-value memory network

[18]. Some recent works improve upon the memory network, making it spatial position-aware [24], developing it with a graph network [12], maintaining a fixed-size memory representation [12], and adding merging and obsoleting schemes for memory frames [13]. Other recent developments in PVOS include contrasting foreground-background features [35] and developing an efficient transductive approach [39]. Even though PVOS methods achieve good performances, they heavily rely on historical pixels and mostly make little use of object-level semantic features. As a result, these methods could be vulnerable to accumulated errors and distractor objects nearby.

**Object matching based VOS (OVOS).** OVOS methods obtain clues by correlating proposal objects of the current frame with template objects of historical frames. DyeNet [11] uses template re-identification and mask propagation iteratively to estimate object masks. FAVOS [4] tracks part regions of objects and aggregates part masks into object masks through template matching. DMM-Net [36] proposes a differentiable and Hungarian-algorithm-like object matching layer. Recent works develop OVOS with approaches like state-switchable tracking [2], tracklet-based dynamic programming [29], 3D convolutions and a rule-based template bank [8] and actor-critic reinforcement learning [26]. Object-level correlations endow segmentation networks with a high-level object estimation. However, all existing OVOS methods fail to learn object templates in an optimizable and dynamic way, and overlook the misalignment between templates and proposals. Then it is not surprising that they are not as strong as the SOTA PVOS methods (refer to Section 4.2).

Our method belongs to the OVOS topic. The differences between our method and previous OVOS methods lie in two aspects. First, our template is generated in a dynamic way via a memory network, while in others, the template is fixed using the annotation in the first frame with/without the prediction of the previous frame. Second, the misalignment between the template and proposals is solved by our adaptive object alignment module for the first time.

## 3. Approach

For simplicity, in Section 3.1, 3.2 and 3.3, we introduce the proposed method in the case of single-object VOS. And in Section 3.4, we introduce how the proposed method can be used for multi-object VOS.

### 3.1. Framework Overview

The overview of our framework is shown in Figure 2. During the video processing, we consider the past frames with object masks (either given in the first frame or estimated in subsequent frames) as the memory frames and the current frame without the object mask as the query frame. Our framework is easy-to-extend because of the modular design. Each of the $5$ modules (base feature extractor, object proposal generator, object template generator, aligned matcher and mask refinement network) can be independently modified or replaced. Meanwhile, all the $4$ modules are end-to-end trained, except for the pre-trained object proposal generator. These modules are introduced as follows.

**Base Feature Extractor.** The base feature extractor takes the frames as input. We use ResNet-50 [6] as our backbone. The output features of the stage-2 to stage-4 ($res2, res3, res4$) of the backbone are used as our base features. To reduce the dimension of the features, we add a bottleneck layer over the last layer of every stage. Object features are extracted by performing ROI pooling on the base features using the corresponding bounding boxes.

**Object Proposal Generator.** Object proposals are generated by an off-the-shelf instance segmentation model for each frame independently. We typically use a COCO-pretrained Mask R-CNN [5] with ResNeXt-101-FPN [14, 32] as the backbone for this task. By default, we collect top-30 output instances (including bounding boxes and masks) as object proposals for each frame to ensure a high recall.

**Object Template Generator.** The object template generator takes object features and object masks of the past frames as memories and takes target object features of the current frame as query. Using the memories and the query, an object template is generated adaptively for the current frame, which is then used for the object matching. In this work, we use the proposed dynamic memory network as an effective and efficient object template generator. The details of the module will be introduced in Section 3.2.

**Aligned Matcher.** The matcher takes features of all object proposals and features of the object template as input. Distances from the proposals to the template are computed by the cosine similarities between their features. Taking the distances as input, a differentiable matching layer [36] is adopted to generate an assignment matrix $\mathcal{A} \in \mathbb{R}^{1\times n}$, where $n$ is the number of object proposals. A coarse mask for the target object is generated by a summation (weighted by $\mathcal{A}$) of masks of all the object proposals. In this work, to avoid misalignment, we use the proposed adaptive object alignment module to align the proposals towards the template in a shared feature space. As a result, the aforementioned distances are computed between the aligned proposals and the template. The details of the module will be introduced in Section 3.3.

**Mask Refinement Network (*RefineNet*).** Base features ($res2 - res4$) and object masks of the first frame, the previous frame and the current frame are fed into *RefineNet* as input. For the two reference frames (the first and the previous frames), we use the output (or given) probability masks. For the query frame, we use the aforementioned coarse mask. Notably, to handle the spatial drifting of objects across frames, the features of the reference frames are

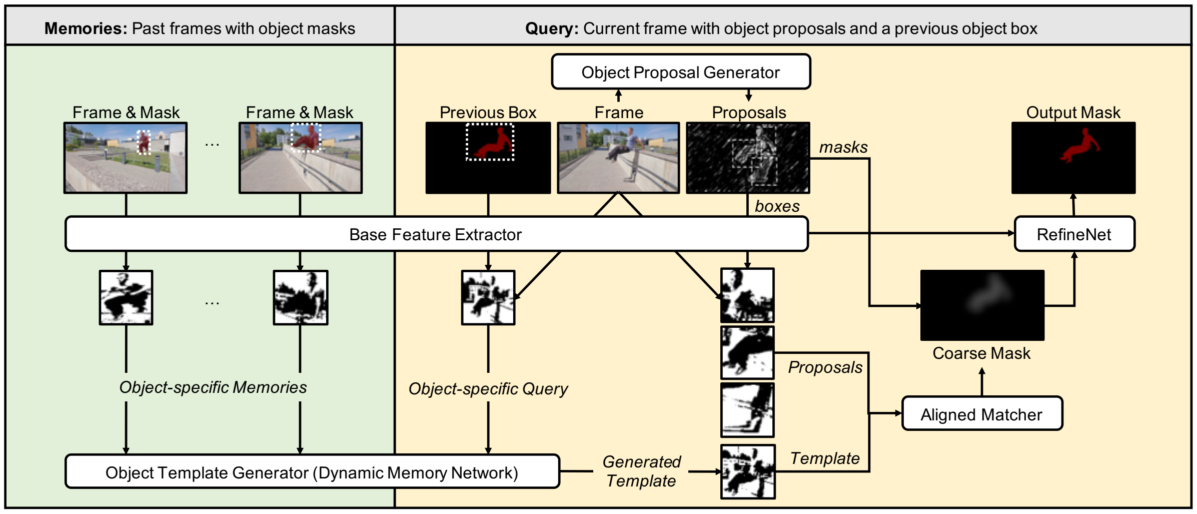

Figure 2. Overview of our framework. We use the proposed dynamic memory network (DMN) as our object template generator. Specifically, DMN takes past frames with object instances (object features and masks) to form object-specific memories, and takes the features of current frame and the object box of previous frame to construct an object-specific query. Using the memories and query, DMN dynamically generates a object template for the current frame. The template is then matched with all proposals by the proposed adaptive object alignment method. A coarse mask is produced from the matching result and then refined by a mask refinement network (*RefineNet*).

aligned towards the features of the current frame in the same manner as AOA before they are fed to *RefineNet*. The architecture of *RefineNet* is the same as in [20, 24, 31], which contains 3 blocks. Each block of this network takes as input the corresponding current features ($res$-$i$), the corresponding reference features ($res$-$i$), and the output features from the previous block (spatially upsampled by $2\times$). The output feature maps of the last block is reduced to 2-channel feature maps by a $1 \times 1$ convolutional layer. The final probability mask is obtained by applying softmax on these 2-channel feature maps.

## 3.2. Dynamic Memory Network (DMN)

As shown in Figure 3, the proposed DMN module consists of a memory encoder, a query encoder and a template decoder. They are introduced as follows.

**Memory Encoder.** The memory encoder takes the object features and the object mask of a memory frame as input. Suppose we have a past frame $m$ with its object features and its object mask as a new memory frame. The object features and the object mask are concatenated along the channel dimension and encoded into pairs of **key** and **value** feature maps through two parallel $3 \times 3$ convolutional layers. Notably, to use in later steps, the key feature maps and the value feature maps of all memory frames are stacked along the temporal axis to 4D tensors. Here we denote the key feature maps as $\mathcal{K} \in \mathbb{R}^{T\times H\times W\times C/4}$ and the value feature maps as $\mathcal{V} \in \mathbb{R}^{T\times H\times W\times C}$, where $T$ is the number of memory frames, and $H$, $W$ and $C$ are the height, the width and the number of channels of the object features.

**Query Encoder.** The query encoder takes the target object features of the current frame as input. Notably, the target object features are obtained by ROI pooling on the base features of the current frame using the object box from the previous frame. The target object features are encoded as the query feature map through a $3 \times 3$ convolutional layer. Here, we denote the query feature map as $\mathcal{Q} \in \mathbb{R}^{H\times W\times C/4}$.

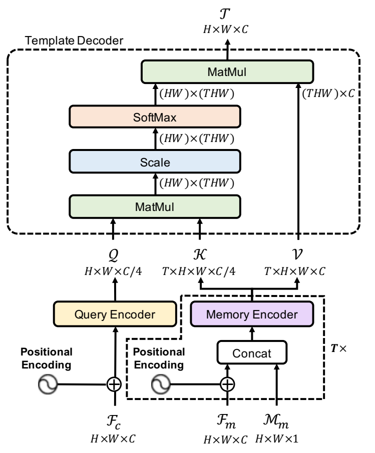

Figure 3. The dynamic memory network. $\mathcal{F}_c$ and $\mathcal{F}_m$ are object features of the current frame $c$ and the memory frame $m$, respectively. $\mathcal{M}_m$ is the object mask of $m$.

**Template Decoder.** The template decoder takes $\mathcal{K}$ and $\mathcal{V}$ of all memory frames and $\mathcal{Q}$ of the current frame as input. Since $\mathcal{Q}$ is a coarse representation of the object's appearance in the current frame, the module learns to use $\mathcal{K}$ to address target object-specific memories $\mathcal{V}$ with respect to $\mathcal{Q}$. Specifically, the relevance score between every pixel of $\mathcal{Q}$

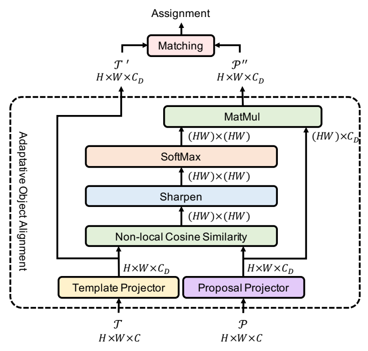
Assignment
Matching
$\mathcal{T}'$ $H\times W\times C_D$
$\mathcal{P}''$ $H\times W\times C_D$
MatMul
$(HW)\times(HW)$
$(HW)\times C_D$
SoftMax
$(HW)\times(HW)$
Sharpen
$(HW)\times(HW)$
Non-local Cosine Similarity
$H\times W\times C_D$
$H\times W\times C_D$
Template Projector
Proposal Projector
Adaptative Object Alignment
$\mathcal{T}$ $H\times W\times C$
$\mathcal{P}$ $H\times W\times C$

Figure 4. The adaptive object alignment module. The template feature map $\mathcal{T}$ and the proposal feature map $\mathcal{P}$ are projected to a shared feature space with $C_D$ dimensions. In the feature space, the proposal is translated to align with the template using the non-local correspondences between their projected feature maps.

and $\mathcal{K}$ is computed densely to determine when-and-where to retrieve $\mathcal{V}$ from. By a summation weighted by the relevance scores, values of the corresponding space-time locations are combined as the template feature map $\mathcal{T} \in \mathbb{R}^{H\times W\times C}$ as follows:

$$\mathcal{T} = \mathrm{softmax}(\frac{\mathcal{Q} \circ \mathcal{K}}{\sqrt{C/4}}) \circ \mathcal{V}, \tag{1}$$

where $\circ$ indicates the matrix multiplication. Notably, all 4D tensors are transformed to 2D matrix before performing the matrix multiplication. The decoding process can be seen as an instantiation of the scaled dot-product attention [27]. This operation can be efficiently implemented using tensor operations in modern deep learning libraries [21].

Notably, to endow DMN with temporal perception, we add positional embeddings [27] to the object features before the memory/query encoding. Specifically, the positional embeddings are encoded using the relative positions (*i.e.* offsets) of the memory/query frame w.r.t. the first frame.

There are two significant differences between the proposed DMN module for objects and the memory networks in PVOS (*e.g.* [20, 24]). Firstly, our memory encoding and query encoding are conducted on the object-centric features other than the features of a complete memory frame. As a result, we eliminate a great deal of redundant information while retaining high-resolution object semantics. Secondly, the base features in our work are computed only once for a frame and all object features of this frame are extracted through ROI pooling on the shared base features. This design makes our model much more efficient than previous works, where the base features for $n$ different objects are computed repeatedly for $n$ times.

## 3.3. Adaptive Object Alignment (AOA)

A main challenge of the object alignment is that the objects in the VOS task come with weak prior knowledge. As a result, it is difficult to apply the alignment methods that require strong prior knowledge, such as the part-part alignment in person re-identification [40]. Furthermore, objects in videos suffer from huge appearance variations due to object movements, camera movements and occlusions. These variations could involve complex non-rigid deformation so that the conventional alignment methods like affine transformation are inapplicable.

The proposed adaptive object alignment method works by translating an object proposal to align with a given template in the feature space (shown in Figure 4). Suppose we have a template feature map $\mathcal{T} \in \mathbb{R}^{H\times W\times C}$ and a proposal feature map $\mathcal{P} \in \mathbb{R}^{H\times W\times C}$. Firstly, both of the feature maps are projected to a shared feature space by two parallel convolutional blocks (*i.e.* projectors). Each projector consists of two $3\times 3$ convolutional layers and a nonlinear layer between them. The output feature maps of the two projectors are denoted as $\mathcal{T}' \in \mathbb{R}^{H\times W\times C_D}$ and $\mathcal{P}' \in \mathbb{R}^{H\times W\times C_D}$, where $C_D$ is the number of channels of the projected feature maps. Secondly, we use a region translation function to align the proposal to appear like the template. Specifically, the non-local cosine similarity is adopted to compute the dense correspondences between all positions of $\mathcal{T}'$ and $\mathcal{P}'$. Inspired by [37, 38], the correspondence map $\mathcal{S} \in \mathbb{R}^{HW\times HW}$ is sharpened by a parameter $\alpha$ and a softmax operation:

$$\mathcal{S}_{i,j} = \frac{\exp\left(\alpha \cdot cos(i,j)\right)}{\sum_{\forall k} \exp\left(\alpha \cdot cos(i,k)\right)}, \tag{2}$$

where $i$ is the position index of the template feature map, $j$ is the position index of the proposal feature map and $cos(i,j)$ is the cosine similarity score between features of location $i$ in $\mathcal{T}'$ and location $j$ in $\mathcal{P}'$. The feature vector in position $i$ of the aligned proposal feature map $\mathcal{P}'' \in \mathbb{R}^{H\times W\times C_D}$ is finally obtained by:

$$\mathcal{P}''_i = \sum_{\forall j} \mathcal{S}_{i,j} \cdot \mathcal{P}'_j \tag{3}$$

The final matching is conducted between the feature maps $\mathcal{T}'$ and $\mathcal{P}''$. Additionally, the region translation function (Equations (2) and (3)) could be computed using the matrix multiplication in modern deep learning libraries with high efficiency [21].

## 3.4. Multi-object Segmentation

The description of our framework is based on having one target object in the video. However, on VOS benchmarks [22, 33], models are required to process multiple objects. In our framework, all steps before *RefineNet* support multiple objects in a single forward pass. Thus, the only extra operation is to perfrom the mask refinement individually for each object. Once we obatin the refined masks, we adopt the soft aggregation method in [20, 31] to merge the predicted probability masks for all objects.

## 4. Experiments

In this section, we evaluate our method on DAVIS 2017 [22] and YouTube-VOS 2018 (YTVOS) [33] benchmarks. DAVIS has 60 videos for training, 30 videos for validation and 60 videos for testing. Each video in DAVIS is at $24fps$ and per-frame annotated, containing either single or multiple objects. YTVOS is a relatively large-scale VOS dataset which has 3471 training videos and 474 validation videos. Each video in YTVOS is at $30fps$ and annotated every 5 frames, containing single or multiple objects. For each benchmark, we train an individual model on its training set. And for evaluation, we use the classic VOS metrics: the region similarity $\mathcal{J}$, the contour accuracy $\mathcal{F}$ and their average $\mathcal{G}$.

To make a fair comparison, except the necessary modifications introduced by the proposed modules, all other modules of our framework use the same architectures as in previous works. Specifically, *Base Feature Extractor* uses the same backbone as STM (ResNet-50) [20] or optionally as CFBI (ResNet-101) [35]. *Object Proposal Generator* is from DMM-Net [36]. The memory encoder and the query encoder of DMN use the same architectures as the two encoders in STM. The template decoder works in the same manner as the memory read module in STM . The matching layer of *Aligned Matcher* is from DMM-Net. The structure of *RefineNet* is from STM.

### 4.1. Implementation Details

**Pre-training on images.** For a fair comparison, we adopt a similar pre-training strategy on images as previous works [12, 13, 16, 20, 24, 29, 34]. Since our method does not necessarily require long videos for training, we simulate video clips by applying random affine transformations on static images from the COCO dataset [15]. Each pre-training clip contains an annotated first frame for reference and three following frames for segmentation.

**Main training on videos.** Our model is firstly initialized from pre-trained weights, and then trained on the VOS benchmarks. On both benchmarks, a training clip is generated by sampling 4 temporally ordered frames from each training video with a random skipping stride from 1 to 5. And the first frame is an annotated reference frame and the following frames are to be segmented.

**Training Details.** We set the batch size to 12 for the pre-training and to 4 for the main training. To avoid the performance degradation caused by the small batch size, we keep the batch normalization layers [9] of the backbone frozen and use group normalization ($G = 32$) [30] for all other modules. The cross entropy loss with Adam Optimizer [10] is used for optimization. In the pre-training, the initial learning rates are set to $1e-5$ for the backbone and set to $1e-4$ for the rest. In the main training, they are set to $1e-7$ and $1e-6$, respectively. With 2 NVIDIA Tesla V100 GPUs, our pre-training stage takes 5 days for 30 epochs. The main training takes 0.5 day for 150 epochs on DAVIS and takes 2 days for 40 epochs on YTVOS.

**Object Proposal Generation.** Following [36], we fine-tune the COCO-pretrained object proposal generator on YTVOS. The batch size is set to 8 and the learning rate is set to $1e-6$. Due to the small number of training videos on DAVIS, we omit its fine-tuning stage to avoid overfitting. Before feeding proposals into our VOS framework, non-maximum suppression (NMS) [19] with a ratio of 0.4 is utilized to remove proposals with high overlap.

**Inference.** On both datasets, encoding all past frames into memory is allowed in our method, thanks to the efficient dynamic memory network. However, for a fair comparison with the state-of-the-art PVOS models, we follow the same memory frame sampling strategy as in [20, 24]. Specifically, the first and the previous frames are always used and other past frames are sampled with a stride of 5.

### 4.2. Comparison with State-of-the-Art Methods

**DAVIS 2017.** Results of our method and the SOTA methods are presented in Table 1. On both the validation set and the test-dev set, our method outperforms all these methods with a significant margin. Notably, the inference time of our method is much less than the SOTA methods like GraphMem [16] and CFBI [35]. This is mainly because in our method, base feature maps are computed only once for each frame and features of all objects in the frame are extracted through ROI pooling on this shared base feature maps. Visualized cases on DAVIS 2017 can be found in the supplementary material.

**YouTube-VOS 2018 (YTVOS).** The validation set of YTVOS contains 474 videos in total, with 65 seen object categories and 26 unseen object categories. On this dataset, our model is pre-trained but is not fine-tuned online. Results of our method and the SOTA methods are presented in Table 2. Our method again outperforms all SOTA methods with a significant margin. The superior performance on this large-scale dataset verifies that our method generalizes better than current methods.

### 4.3. Ablation Studies

In this section, we analyze our method under different settings on DAVIS 2017 validation set. These settings can be categorized into four kinds: architecture, memory sampling, training and object proposal generator tuning. The ablation results are shown in Table 3. Notably, the *Full* model (*F0*) is the best variant of our method, which uses the settings of these four kinds as follows: (a) both DMN and AOA are used, (b) the pre-training and the main training are applied successively, (c) the first and previous frames are always used as the memory frames and other past frames are sampled with a stride of 5, and (d) the object proposal

| Models | Backbone | *OL* | *Pre* | *YV* | $\mathcal{J}_{val}$ | $\mathcal{F}_{val}$ | $\mathcal{G}_{val}$ | $\mathcal{J}_{test}$ | $\mathcal{F}_{test}$ | $\mathcal{G}_{test}$ | t/s |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FRTM [23] (CVPR20) | R101 | ✓ | | ✓ | - | - | 76.7 | - | - | - | 0.09 |
| DMM-Net [36] (ICCV19) | R101 | | | | 68.1 | 73.3 | 70.7 | - | - | - | 0.12 |
| FEELVOS [28] (CVPR19) | DL3+ | | | ✓ | 69.1 | 74.0 | 71.5 | 55.2 | 60.5 | 57.8 | 0.54 |
| SAT [2] (CVPR20) | R50 | | | | 68.6 | 76.0 | 72.3 | - | - | - | **0.03** |
| TVOS [39] (CVPR20) | R50 | | | | 69.9 | 74.7 | 72.3 | 58.8 | 67.4 | 63.1 | **0.03** |
| TAN-DTTM [8] (CVPR20) | R50 | | | | 72.3 | 79.4 | 75.9 | 61.3 | 70.3 | 65.4 | 0.14 |
| CFBI [35] (ECCV20) | R101 | | | ✓ | 79.1 | 84.6 | 81.9 | 71.1 | 78.5 | 74.8 | 0.37 † |
| GC [12] (ECCV20) | R50 | | ✓ | | 69.3 | 73.5 | 71.4 | - | - | - | 0.08 † |
| AFB-URR [13] (NeuIPS20) | R50 | | ✓ | | 73.0 | 76.1 | 74.6 | - | - | - | - |
| Siam R-CNN [29] (CVPR20) | R101 | | ✓ | ✓ | 69.3 | 80.2 | 74.8 | 57.3 | 66.9 | 62.1 | 1.0 |
| STM [20] (ICCV19) | R50 | | ✓ | | 69.2 | 74.0 | 71.6 | | - | 62.1 | 0.32 † |
| STM [20] (ICCV19) | R50 | | ✓ | ✓ | 79.2 | 84.3 | 81.8 | 69.3 | 75.2 | 72.2 | 0.32 † |
| GraphMem [16] (ECCV20) | R50 | | ✓ | ✓ | 80.2 | 85.2 | 82.8 | - | - | - | 0.40 † |
| KMN [24] (ECCV20) | R50 | | ✓ | ✓ | 80.0 | 85.6 | 82.8 | 74.1 | 80.3 | 77.2 | 0.24 † |
| Ours | R50 | | ✓ | | 78.6 | 84.0 | 81.3 | - | - | - | 0.15 |
| Ours | R50 | | ✓ | ✓ | **81.0** | **87.0** | **84.0** | - | - | - | 0.15 |
| Ours | R101 | | ✓ | ✓ | **81.4** | **87.5** | **84.5** | **74.8** | **81.7** | **78.3** | 0.16 |

Table 1. Quantitative results On DAVIS 2017. $R50$, $R101$ and $DL3+$ denotes ResNet-50, ResNet-101 and DeepLabv3+ [1]. *OL* indicates fine-tuning on first frames. *Pre* indicates pre-training on image datasets. *YV* indicates an extra use of YTVOS for training. $val$ and $test$ denote the validation set and the test-dev set, respectively. $t/s$ denotes inference time per frame in seconds on DAVIS 2017 validation set, and † denotes the time extrapolated from single-object inference time on DAVIS 2016 validation set.

.

| Models | $\mathcal{J}_S$ | $\mathcal{J}_U$ | $\mathcal{F}_S$ | $\mathcal{F}_U$ | $\mathcal{G}$ |
|---|---|---|---|---|---|
| DMM-Net [8] | 60.3 | 50.6 | 63.5 | 57.4 | 58.0 |
| SAT [2] | 67.1 | 55.3 | 70.2 | 61.7 | 63.6 |
| PReMVOS [17] | 71.4 | 56.5 | 75.9 | 56.5 | 66.9 |
| TVOS [39] | 67.1 | 63.0 | 69.4 | 71.6 | 67.8 |
| FRTM [23] | 72.3 | 65.9 | 76.2 | 74.1 | 72.1 |
| SiamRCNN [29] | 73.5 | 66.2 | - | - | 73.2 |
| GC [12] | 72.6 | 68.9 | 75.6 | 75.7 | 73.2 |
| STM [20] | 79.7 | 72.8 | 84.2 | 80.9 | 79.4 |
| AFB-URR [13] | 78.8 | 83.1 | 74.1 | 82.6 | 79.6 |
| GraphMem [16] | 80.7 | 74.0 | 85.1 | 80.9 | 80.2 |
| CFBI [35] | 81.1 | 75.3 | 85.8 | 83.4 | 81.4 |
| KMN [24] | 81.4 | 75.3 | 85.6 | 83.3 | 81.4 |
| Ours (R50) | **82.5** | **76.2** | **86.9** | **84.2** | **82.5** |
| Ours (R101) | **82.6** | **76.7** | **87.0** | **84.8** | **82.7** |

Table 2. Quantitative results on YTVOS validation set. $S$ and $U$ denote the seen and unseen object categories in the training set.

generator is additionally tuned on YTVOS. When one kind of ablation settings is studied, the other settings are kept the same as the *Full* model.

**Architecture.** In *(A)* of Table 3, we investigate the contributions of our proposed DMN and AOA modules. The base model takes annotations in the first frame as a fixed template. By replacing the fixed template with the template generated from DMN, a 2.7% improvement on $\mathcal{G}$ is achieved (*A1 vs. A2*). This indicates that exploiting the object memories helps handle large appearance changes while being robust to error accumulation. Meanwhile, the AOA module brings in a 1.1% performance gain (*A1 vs. A3*). This indicates that the misalignment between the object proposals and templates degrades the performance of VOS significantly. The misalignment problem in VOS is not touched by previous works and we take a first try to shed some light on it. Moreover, combining DMN and AOA leads to an additional improvement (*F0 vs. A2+A3*), indicating that object memories and adaptive alignment boost each other. Specifically, on one hand, more appearance cues of the target object encoded in memories brings in a more accurate alignment. On the other, more robust similarity measurement between templates and aligned proposals provides better information for the learning of preceding modules including DMN. When the memory encoder in DMN uses the frame features instead of the object features, a 1.9% performance drop is observed (*A4 vs. F0*). This reveals that the object semantics are retained better in object-based memories compared with frame-based memories. Removing the matcher (*i.e.* not use AOA and proposals at all) and decoding masks from the DMN results in a significant performance drop of 11.3% (*A5 vs. F0*). This is because the model does not take the advantages of either PVOS (*e.g.* foreground-background integration) or OVOS (*e.g.* proposals as coarse estimates). Finally, our coarse masks achieve a fairly good performance of 69.2%, and introduce *RefineNet* upon the coarse masks further improves this result by 12.1% (*A6 vs. F0*).

**Memory Sampling.** Results of different memory sampling strategies are shown in *(B)* of Table 3. The first and

| | Ablation Settings | $\mathcal{J}$ | $\mathcal{F}$ | $\mathcal{G}$ | t/s |
|---|---|---|---|---|---|
| *F0* | Full | 78.6 | 84.0 | 81.3 | 0.15 |
| | (A) Architecture | | | | |
| *A1* | Base | 75.2 | 80.3 | 77.8 | 0.13 |
| *A2* | Base w/ DMN | 77.9 | 83.1 | 80.5 | 0.14 |
| *A3* | Base w/ AOA | 76.4 | 81.4 | 78.9 | 0.14 |
| *A4* | Full w/ FrameMem | 76.8 | 82.0 | 79.4 | 0.15 |
| *A5* | Full w/o Matcher | 67.5 | 72.5 | 70.0 | 0.11 |
| *A6* | Full w/o RefineNet | 67.1 | 71.2 | 69.2 | 0.11 |
| | (B) Memory Samspling | | | | |
| *B1* | First frame only | 77.3 | 82.5 | 79.9 | 0.14 |
| *B2* | Previous frame only | 77.8 | 82.6 | 80.2 | 0.15 |
| *B3* | First+previous frames | 77.7 | 83.0 | 80.4 | 0.15 |
| *B4* | All past frames | 79.2 | 84.3 | 81.8 | 0.19 |
| | (C) Training | | | | |
| *C1* | Main training only | 70.3 | 74.7 | 72.5 | 0.15 |
| *C2* | Pre-training only | 65.8 | 70.0 | 67.9 | 0.15 |
| | (D) Object Proposal Generator Tuning | | | | |
| *D1* | Tuning w/ DAVIS | 76.3 | 81.7 | 79.0 | 0.15 |
| *D2* | Tuning w/o YTVOS | 77.5 | 82.7 | 80.1 | 0.15 |

Table 3. The ablation results on DAVIS 2017 validation set. Refer to Section 4.3 for details about the *Full* model. Notably, in ablation studies, we use ResNet-50 as the backbone and the YTVOS dataset is not used for training.

previous frames are believed to provide important information for object segmentation of the current frame. Comparing *B1* and *B2*, the previous frame looks more useful to handle failure cases. In addition to first and previous frame, we sample a new intermediate memory frame at every 5 frames in our *Full* model. Utilization of these intermediate frames leads to an improvement of 0.9% (*B3* vs. *F0*). Having the intermediate-frame memories further boosts performance in providing more information about the object appearances. In addition, an interesting observation is that the *B1* model is slightly better than the *A3* model. All configurations and inputs of the two models are the same, except for the use of DMN in the previous model. This result indicates that the template decoding with memory addressing mechanism itself is beneficial for object matching. Last but not least, our method is capable of exploiting all past frames thanks to the efficient dynamic memory network. By doing so, a slightly better performance (+0.5%) is achieved while the inference time is increased by 27% approximately.

**Training.** As shown in *(C)* of Table 3, the *C1* model drops by 8.8%, indicating that the amount of training video data is not enough to bring out the potential of our method. However, previous works perform much worse (*e.g.* 43.0% of [20] *vs.* 72.5% of $C1$) on DAVIS without pre-training. This verifies that our method is rather robust and generalizes well on datasets in various scales. The *C2* model achieves a result of 67.9% on $\mathcal{G}$ with a drop of 13.4%. This is because

| *BFE* | *OPG* | *OTG* | *AM* | *MRN* |
|---|---|---|---|---|
| 0.009 | 0.020 | 0.015 | 0.044 | 0.066 |

Table 4. The inference time per frame ($t/s$) of the 5 modules of our method on DAVIS-17 validation set. These modules are respectively base feature extractor (w/ ResNet-50), object proposal generator, object template generator, aligned matcher and mask refinement network.

the object categories are quite different between DAVIS and COCO. Lastly, maximum performance is obtained by using both training strategies.

**Object Proposal Generator Tuning.** As shown in *(D)* of Table 3, we test the performance of our method under different object proposal generator tuning strategies. Specifically, tuning the generator on images of a small-scale dataset like DAVIS results in a degradation of performance (*D1 vs. F0*/*D2*). The degradation is mainly caused by the overfitting problem and we suggest not to tune the generator on the small-scale datasets. Actually, in most cases, an instance segmentation model pre-trained on COCO works well on generating high-recall object proposals for our method (*D2*). This means that our method can be easily transferred to a new dataset with an end-to-end training on the VOS task only. To further boost the performance, we tune the generator on the large-scale dataset YTVOS. This extra tuning brings a 1.2% improvement on $\mathcal{G}$ (*F0* vs. *D2*).

**Inference Time.** On DAVIS 2017 validation set, we run the inference of each module independently and the detailed processing time is shown in Table 4. Our method is fast for three reasons. Firstly, the object proposal generator module is efficient since it can be performed for multiple frames concurrently. Secondly, the base features in our work are computed once for a frame while in previous works [13, 16, 20, 24] they are repeatedly computed for every target object. Lastly, the DMN module and the AOA module are hugely accelerated by matrix multiplication.

## 5. Conclusion

In this paper, we present a novel object-matching framework for semi-supervised video object segmentation. Our model alleviates the mismatches between object templates and proposals by incorporating two key components. The first one is a dynamic memory network that learns to read relevant object information from multiple past frames, and the second is an adaptive object alignment module that aligns proposals towards templates in the feature space. The experimental results show that our method outperforms all state-of-the-art methods on VOS benchmarks with a very efficient inference time.

## References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 6

[2] Xi Chen, Zuoxin Li, Ye Yuan, Gang Yu, Jianxin Shen, and Donglian Qi. State-aware tracker for real-time video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9384–9393, 2020. 1, 3, 6, 7

[3] Yuhua Chen, Jordi Pont-Tuset, Alberto Montes, and Luc Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1189–1198, 2018. 2

[4] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and accurate online video object segmentation via tracking parts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7415–7424, 2018. 1, 2, 3

[5] 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. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[7] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G Schwing. Videomatch: Matching based video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 54–70, 2018. 1, 2

[8] Xuhua Huang, Jiarui Xu, Yu-Wing Tai, and Chi-Keung Tang. Fast video object segmentation with temporal aggregation network and dynamic template matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8879–8889, 2020. 3, 6, 7

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 6

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[11] Xiaoxiao Li and Chen Change Loy. Video object segmentation with joint re-identification and attention-aware mask propagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 90–105, 2018. 1, 2, 3

[12] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module. In *European Conference on Computer Vision*, pages 735–750. Springer, 2020. 2, 3, 6, 7

[13] Yongqing Liang, Xin Li, Navid Jafari, and Qin Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. *arXiv preprint arXiv:2010.07958*, 2020. 2, 3, 6, 7, 8

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[16] Xiankai Lu, Wenguan Wang, Danelljan Martin, Tianfei Zhou, Jianbing Shen, and Van Gool Luc. Video object segmentation with episodic graph memory networks. In *ECCV*, 2020. 2, 6, 7, 8

[17] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*, pages 565–580. Springer, 2018. 7

[18] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016. 2

[19] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006. 7

[20] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9226–9235, 2019. 1, 2, 4, 5, 6, 7, 8

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[22] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 5, 6

[23] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7406–7415, 2020. 6, 7

[24] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *European Conference on Computer Vision*, pages 629–645. Springer, 2020. 2, 4, 5, 6, 7, 8

[25] Jae Shin Yoon, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2167–2176, 2017. 2

[26] Mingjie Sun, Jimin Xiao, Eng Gee Lim, Bingfeng Zhang, and Yao Zhao. Fast template matching and update for video object tracking and segmentation. In *Proceedings of*

*the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10791–10799, 2020. 1, 3

[27] 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. 5

[28] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9481–9490, 2019. 1, 2, 6

[29] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam r-cnn: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6578–6588, 2020. 1, 2, 3, 6, 7

[30] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 6

[31] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7376–7385, 2018. 4, 5

[32] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3

[33] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018. 5, 6

[34] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6499–6507, 2018. 6

[35] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *Proceedings of the European Conference on Computer Vision*, 2020. 1, 3, 6, 7

[36] Xiaohui Zeng, Renjie Liao, Li Gu, Yuwen Xiong, Sanja Fidler, and Raquel Urtasun. Dmm-net: Differentiable mask-matching network for video object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3929–3938, 2019. 1, 2, 3, 6, 7

[37] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8061, 2019. 5

[38] Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5143–5153, 2020. 5

[39] Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin. A transductive approach for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6949–6958, 2020. 3, 6, 7

[40] Wei-Shi Zheng, Xiang Li, Tao Xiang, Shengcai Liao, Jianhuang Lai, and Shaogang Gong. Partial person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4678–4686, 2015. 5