IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2021, accepted August 10, 2021, date of publication August 20, 2021, date of current version August 27, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3106353*

# Adaptive Template and Transition Map for Real-Time Video Object Segmentation

**HYOJIN PARK[1], JAYEON YOO[1], GANESH VENKATESH[2], AND NOJUN KWAK[1], (Senior Member, IEEE)**

[1]Graduate School of Convergence Science and Technology, Seoul National University, Seoul 08826, South Korea
[2]Facebook Inc., Menlo Park, CA 94025, USA

Corresponding author: Nojun Kwak (nojunk@snu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant (2021R1A2C3006659) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (No.2021-0-01343) both funded by the Korea government (MSIT).

**ABSTRACT** Semi-supervised video object segmentation (semi-VOS) is required for many visual applications. This task is tracking class-agnostic objects from a given segmentation mask. Various approaches have been developed and achieved high accuracy in this field, but these previous models are hard to be utilized in real-world applications due to slow inference time and tremendous complexity. To significantly speed up inference while reducing performance gaps from those previous models, we introduce a fast segmentation model based on a template matching method and auxiliary loss with a transition map. Our template matching method consists of short-term and long-term matching. The short-term matching enhances target object localization by focusing on neighboring frames, while long-term matching improves fine details and handles object shape-changing by considering long-range frames. However, since both matching processes generate each template based on the previously estimated masks, this incurs error propagation for tracking objects in the next frames. To mitigate this problem, we add auxiliary loss with a newly proposed transition map for encouraging correction power to create accurate masks of the target object. Our model obtains 81.1% $J\&F$ score at the speed of 78.3 FPS on the DAVIS16 benchmark and achieves 1.4× faster speed and 11.3% higher accuracy than SiamMask, one of the fast semi-VOS models.

**INDEX TERMS** Semi-supervised video object segmentation, video object segmentation, video object tracking, deep learning.

## I. INTRODUCTION

Video object segmentation (VOS) is an essential technique to precisely identify the shape of target objects under various conditions in every video frame. This technique is necessary for many applications such as autonomous driving, video editing, and surveillance systems. In this paper[1], we focus on the *semi-supervised video object segmentation* (semi-VOS) task, which is to track a target in a pixel-wise resolution from a given annotated mask for the first frame.

Semi-VOS is a challenging task because we need to embed information about a target regardless of its class and have to consider the shape-changing of the target over time. Many methods are proposed to resolve these problems, but it is hard to be used in real-world environments due to tremendous computation. For example, online-learning fine-tunes model parameters using the given frame image and the corresponding ground truth mask [2]–[8]. This strategy makes the model's parameters more specialized in each target object, but it requires additional time and memory for fine-tuning at the inference stage. Other approaches,

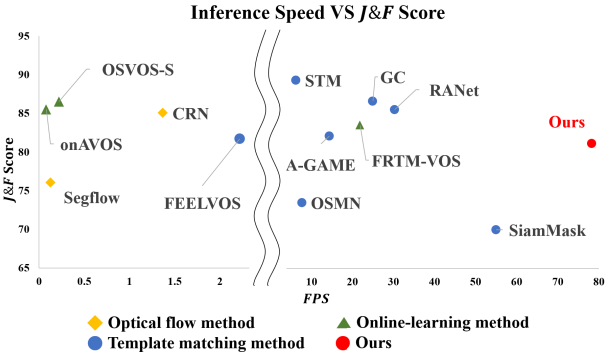



**FIGURE 1.** The speed (FPS) vs accuracy ($J\&F$ score) on the DAVIS2016 validation set. Our proposed model achieves high accuracy with small complexity.

The associate editor coordinating the review of this manuscript and approving it for publication was Victor Sanchez.

[1]This work is an extended version of TTVOS [1].

matching methods [9]–[11], [13], [14], construct target information as a template and match it to the current frames for finding the target. This method does not need extra-training in inference stage, but it is not fast enough for real-world applications. For example, the required memories of target information increase over time in some models for handling shape variation of the target, or a lot of computation is needed for the process of matching or updating templates at every frame.

Another challenge for semi-VOS is temporal consistency of masks across frames. There are two perspectives regarding temporal consistency. Firstly, a sequence of masks changes smoothly across frames. In other words, the target object's movements in consecutive frames are plausibly connected. Secondly, the target mask of the entire frame should exclude the non-target region, even if there is shape-changing in the target object across frames. This is a more challenging issue in a template matching method since the template is maintained based on previous results for the next frame's prediction. It can incur some risk: prediction failures started in a certain frame may accumulate over time and eventually lead the entire estimation in the wrong direction, resulting in the final mask containing a completely different region irrelevant from the target object.

Optical flow is one of the popular methods to resolve the mentioned problems for diverse video applications by correctly estimating pixel-wise movement vectors or trajectories. In semi-VOS tasks, it propagates a given mask or features across frames to re-align the information for current frames [18]–[21]. However, even though the optical flow is only a part of the entire process, it needs huge resource to provide too much information for the segmentation tasks. For example, RAFT [22] is one of fast models in this field but it needs 550 ms per frame on 1080p ($1088 \times 1920$) video from DAVIS, and this slows down both training and inference. We believe that it is sufficient to know the binary information of whether a pixel is changed to foreground or background. Therefore, we propose a more efficient method which trains the model how to correct wrong predictions without needs of optical flow. We use newly-defined ground truth which guarantees temporal consistency. The ground truth guides the model to learn in a direction that reduces error propagation and improves coherence regarding prediction masks.

Our approach complements the existing video object segmentation approaches by proposing an adaptive template matching method and a novel transition map to improve temporal consistency. Our contributions can be summarized as follows:

- We propose a new lightweight VOS model based on two template matching methods: short-term matching for localization and long-term matching for fine mask generation to achieve fast inference time and to reduce the accuracy gap from previous heavy models.
- Our template is adaptively updated to manage shape variation of target objects without heavy computation and occupying additional memory.
- We propose a concept of transition map for auxiliary loss to learn how to correct mis-estimated pixels from previous frames for current frames to prevent error propagation.

## II. RELATED WORK

Here, we discuss various approaches for semi-supervised video object segmentation (semi-VOS) as shown in Table 1.

### A. VOS BASED ON ONLINE-LEARNING

The online-learning method is training the model with new data in each inference iteration [2]–[4]. In the semi-VOS task, model parameters are fine-tuned in the inference stage with a given input image and a corresponding mask. Therefore, the model is specialized for the given condition [5]–[7]. However, fine-tuning causes additional computation in inference time. Robinson *et al.* [8] resolve this issue by dividing the model into two sub-networks. One is a lightweight network that is fine-tuned in the inference stage for making a coarse score map. The other is a heavy segmentation network without the need for fine-tuning. This network enables fast optimization and relieves the burden of online-learning.

### B. VOS BASED ON TEMPLATE MATCHING

Template matching is one of the traditional methods in the tracking task. It generates a template and calculates similarity with input as a matching operation. A-GAME [9] designs a target distribution by a mixture of Gaussian in embedding space, and it predicts posterior class probabilities for matching. RANet [10] applies a ranking system to the matching process between multiple templates and input for extracting reliable results. FEELVOS [11] calculates distance map by local and global matching for better robustness. SiamMask [12] uses a depth-wise operation for fast matching. This model show better speed than other model, but the accuracy is lower than others. Recently, memory-query based models improve accuracy a lot. STM [13] constructs multiple memories for embedding target information, and finding target by matching between memories and input frame as *key-value-query* concept. However, it requires lots of resources

**TABLE 1. Summary of various approaches for semi-supervised video object segmentation.**

| Approach | Advantage | Disadvantage | Example |
|---|---|---|---|
| Online Learning | High accuracy from specialized model | Slow speed due to fine tuning in inference stage | [2]–[8] |
| Template matching | Fast speed from simple matching method | Low accuracy and slow speed when using many templates | [9]–[14] |
| Optical flow | Refining results by propagated information | Slow speed due to computation for optical flow | [15]–[17] |

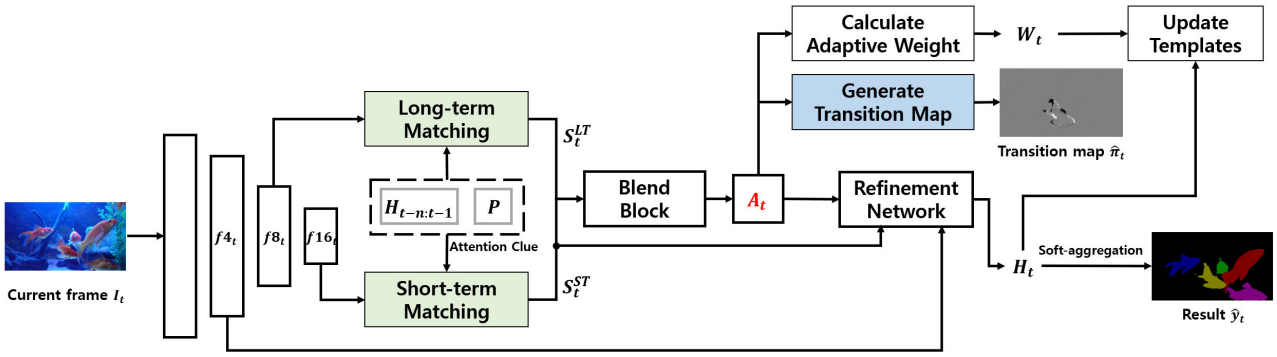



**FIGURE 2.** The overall architecture for real-time video object segmentation (VOS). A backbone feature is shared in all the processes of VOS for efficiency. The two types of template matching (short-term and long-term) generate rough attention map for tracking object, and then the refinement network makes accurate final mask. The heatmaps $Ht-n:t-1$ and position information $P$ are applied as attention clue. The long-term matching takes both information, but the short-term matching use only heatmaps. The transition map $\hat{\pi}_t$ is computed only in the training phase for enhancing temporal consistency. Finally, the template is updated with a current prediction and a adaptive weight.

because the amount of memory is increased over time and the size of memory is the square of the resolution of an input feature map. To lower this huge complexity, GC [14] does not stack memory at each time frame, but accumulates them into one, which has smaller size than an unit memory of STM.

### C. OPTICAL FLOW FOR TEMPORAL CONSISTENCY

Consistency loss is widely used for improving performance in semi-supervised learning by artificial perturbation to input [15]–[17]. In many VOS tasks, the meaning of consistency is temporal consistency which represents smooth changing of segmentation masks while excluding non-target region about consecutive frames. To enhance this temporal consistency, optical flow is applied in such a way that it re-aligns with the differences between masks or features of adjacent frames by estimated flow vectors of moving objects.

CRN [20] generates coarse masks of moving objects based on optical flow and refines them with the following networks. In many works, optical flow is used to convey additional information to guide the segmentation network for refinement [6], [23]. Similarly, FAVOS [18] leverages optical flow to warp previous predictions and uses them with current predictions to generate accurate ground truth masks for online learning. Segflow [21] designs end-to-end networks with two sub-networks complementing each other for image segmentation and optical flow, respectively, to estimate target masks. However, optical flow is difficult to apply to real-time applications because it requires a huge amount of computation. Instead of using optical flow, we propose a method based on binary information of whether pixels change from foreground to background, and vice versa.

## III. THE PROPOSED METHOD

In this section, we present our semi-VOS model. Section III-A introduces the whole model architecture and how to manage multi-object VOS. Section III-B describes short-term and long-term template matching methods how to calculate the correlation between templates and frames for finding targets. Section III-C explains the details of generating and updating a template for long-term matching. Finally, Section III-D demonstrates our concept of a transition map to enhance temporal consistency by L2 loss with newly-defined ground truth for mitigating error propagation between neighboring frames.

### A. OVERALL ARCHITECTURE

We propose a new architecture for VOS as shown in Figure 2. Our model consists of feature extraction, template matching, generation of transition map, refinement network, and template update with adaptive weight. In the feature extraction stage, we extract feature maps of multiple sizes from the current frame for the other modules.

We perform template matching in two ways: short-term and long-term with attention clue. In short-term matching, we compute the similarity between the short-term template and the current feature map. Here, the short-term template is generated from the feature map of the previous frame $t-1$ and we use small feature maps to reduce the computation. However, this incurs two problems: 1) Utilizing only feature maps from the previous frames $t-1$ causes too much dependency on previous results in the output masks. 2) Matching results from small feature maps are not sufficient to obtain a fine target shape.

To resolve these problems, we also propose long-term matching by using long-range information. The long-term template contains the entire frame's information as a global template to avoid over-dependence on the previous frame's information. In this module, we use larger feature maps to get more detailed information sufficient for generating accurate masks. After then, two matching results are combined to produce an attention map $A_t$. The backbone generates features of various sizes $fN_t$, where $fN_t$ denotes a feature map of the current frame $I_t$ with a $1/N$-sized width and height compared to the input. The short-term matching uses $f16_t$, and the long-term template matching uses $f8_t$ for producing each similarity map. The details are in Section III-B.

After then, our model generates the final segmentation mask by refinement network. In the refinement network, we apply ConvTranspose to $A_t$ and other feature maps for upsampling. We conduct PixelShuffle [24] in the last layer to prevent the grid-effect on the final segmentation heatmap $\hat{H}_t$ and the detail is described in Section III-E. Also, a scalar value as weight $w_t$ is calculated for updating the long-term template, and $w_t$ adaptively decides how much current information is added to the template for the next frame. Finally, $f16_t$ and current heatmap $\hat{H}_t$ are used for replacing short-term template matching for next frames. In the case of long-term template, $f8_t$, $\hat{H}_t$ and $w_t$ are utilized to update for the next template matching as described in following Section III-C. At only training time, the model estimates a transition map $\hat{\pi}_t$ from $A_t$. The transition map estimates pixel-wise transition information between background and foreground from frame $t-1$ to frame $t$ to induce correction power and to encourage temporal consistency for the current results, and the detail is explained in Section III-D.

#### 1) MULTI OBJECT CASE

All the backbone features are also shared in a multi-object scenario. However, the other remained stages are conducted separately for each object. All the heatmaps of objects have two channels for the probability of background and foreground, and we use these heatmaps to produce the final segmentation mask $\hat{y}_t$ by the soft-aggregation method in the inference time fallowing [9], [25]. In single object case, we directly use $\hat{H}_t$ as the final segmentation mask $\hat{y}_t$.

#### 2) IMPROVEMENT FROM THE PREVIOUS WORK [1]

In our previous work, we used different attention clue for template matching. In this paper, we use multiple heatmaps and coordinate information for new attention clue. Also, we propose adaptive weight module for updating the long-term template. In our previous work, we used only a previous heatmap $H_{t-1}$ and fixed the weight decay parameter by $1/t$.

### B. TEMPLATE MATCHING

In tracking objects, object localization and fine segmentation mask generation are critical to improving accuracy. Localization means that the model finds where the target object is, even if there are other similar objects in the same frame. Fine segmentation mask is generated by handling a target's shape-changing from the first given image and corresponding mask. To tackle the above problems, we make two assumptions based on empirical observation. Firstly, the position of the object in the current frame $t$ is close to that in the previous frame $t-1$. Secondly, we can extract common factors encompassing long-range frames, even if the shape of the object changes across frames. Therefore, we devise short-term matching for localization from the first assumption and long-term matching for accurate masks from the second assumption.

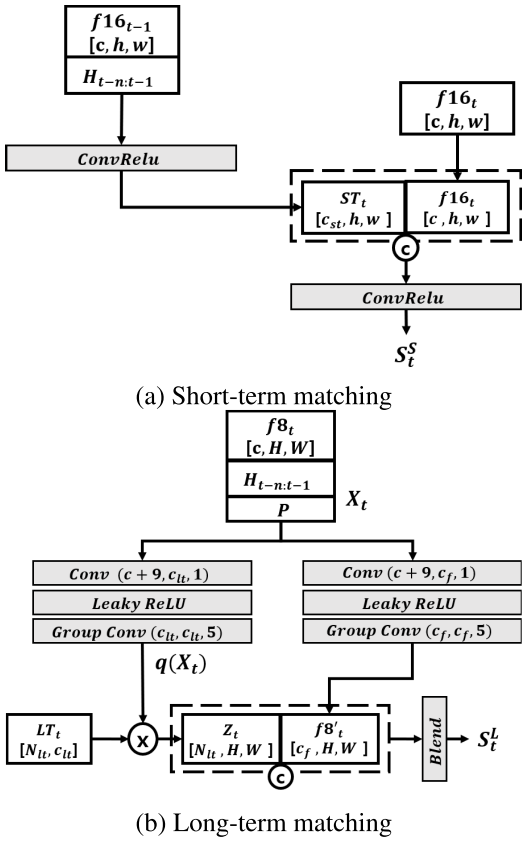



**FIGURE 3.** (a) Structure of generating short-term template $ST_t$ and matching process (b) Structure of process of long-term matching by using long-term template $LT_t$.

#### 1) ATTENTION CLUE

Not all vectors in a feature map of frame contain the information of target objects since most vectors are generated from background pixels, and only a few vectors are involved in the target objects. To mitigate the inflow of non-object information, we give attention clue from previous information. We set $n$ as 3 for $\hat{H}_{t-n:t-1}$ (*i.e.* $\hat{H}_{t-3}$, $\hat{H}_{t-2}$, *and* $\hat{H}_{t-1}$) under empirical observation as previous information. Each heatmap consists of two channels containing the probability of background and foreground, respectively. We also add three types of coordinate layers $P$ to enhance position information on heatmaps for the long-term template. $P$ consists of coordinate values along the x-axis and y-axis, and radial distance from the center.

#### 2) SHORT-TERM MATCHING

First, $f16_{t-1}$ and attention clue are concatenated together, and the concatenated feature map is forwarded by several convolution layers to embed short-term template $ST_t$ as shown in Figure 3 (a). Finally, the latest updated short-term template $ST_t$ and the current feature map $f16_t$ are concatenated to compute similarity between both features as $S_t^{ST}$ by a single convolution layer and a single ReLU layer.

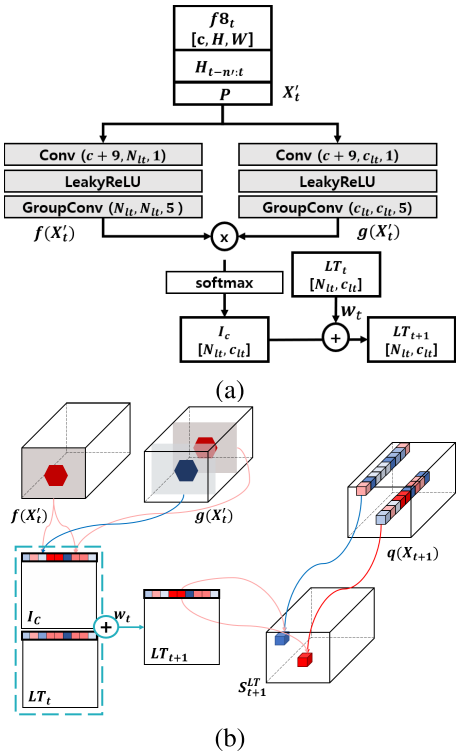



**FIGURE 4.** (a) The detailed structure of template update. An operation (a,b,c) denotes the input channel, output channel, and kernel size of convolution operation, respectively. (b) Process in updating long-term template and matching the template with query. Here, a red (blue) color means a high (low) similarity between two information. The size of $f(X'_t)$ and $g(X_t)$ is finally reshaped to $c_{lt} \times HW$, but we draw feature maps as $c_{lt} \times H \times W$ for the sake of convenient understanding.

### 3) LONG-TERM MATCHING

Long-term template matching consists of 1) generating a correlation map $Z_t$ by comparing the query $q(X_t)$ to the long-term template $LT_t$ 2) blending $Z_t$ and $f8'_t$, the modified feature map of the current frame, to produce long-term similarity $S^{LT}_t$ as shown in Figure 3 (b). To generate query $q(X_t)$, the backbone feature map $f8_t$ is concatenated with attention clue to suppress information far from target object location of the previous frames, and the concatenated feature is forwarded to convolution layers. $Z_t$ is produced by matrix multiplication between the template $LT_t \in \mathbb{R}^{N_{lt} \times c_{lt}}$ and the query feature map $q(X_t) \in \mathbb{R}^{c_{lt} \times HW}$ as follows:

$$Z_t = LT_t \times q(X_t). \tag{1}$$

Here $LT_t$ is a template matrix consisting of $N_{lt}$ vectors with $c_{lt}$ dimensions describing the common factor of the target object. In other words, each row of $LT_t$ is one template vector which is generated across long-range frames following Section III-C, and the number of columns $N_{lt}$ means that the model extracts $N_{lt}$ types of common factor. Each column vector of $q(X_t)$ is a feature vector in the particular position $(h, w)$. The row vector from $LT_t$ and the column vector of $q(X_t)$ are compared using the inner product for calculating correlation. Therefore, if the resultant value from Eq. (1) is high at the certain $(h, w)$ position, there is high possibility that the $(h, w)$ position be included in the target, and vice versa. After then, the long-term correlation map $Z_t$ and modified feature map $f8'_t$ are concatenated to make the final feature map by blending both results as shown in the bottom of Figure 3(b).

To reduce computational cost while retaining a large receptive field, we use group convolution (group size of 4) with a large kernel size of $5 \times 5$ of kernel for generating $q(\cdot)$. Although depth-wise convolutions has less FLOPs than group convolutions, we do not use them because the larger number of groups in convolutions adversely affects the model execution time [26]. We select LeakyReLU as the non-linearity to avoid the dying ReLU problem. We use a point-wise convolution first then apply the group convolution due to the group size for convolution.

### C. UPDATE LONG-TERM TEMPLATE

We conjecture that pixels inside a target object have a distinct embedding vector distinguished from non-target object pixels. Our model is designed to embed this vector by self-attention while removing the irrelevant information of the target object. Each current embedding vector updates a previous long-term template by adaptive weight at each frame, and finally the template represents common factor describing target object. The updated template is used to find the target object in the next frame by generating similarity map as shown in Figure 4.

For constructing the current embedding vector, the backbone feature $f8_t$ and attention clue are concatenated to generate $X'_t$ while suppressing information far from the target object. Here, heatmaps $\hat{H}_{t-n+1:t}$ in attention clue consist of $\hat{H}_{t-2}$, $\hat{H}_{t-1}$, and $\hat{H}_t$. The concatenated feature map $X'_t$ is forwarded to two separate branches $f(\cdot)$ and $g(\cdot)$, making $f(X'_t) \in \mathbb{R}^{N_{lt} \times H \times W}$ and $g(X'_t) \in \mathbb{R}^{c_{lt} \times H \times W}$ like embedding key and value function as shown in Figure 4 (a). After then, the feature maps are reshaped to $N_{lt} \times HW$ and $c_{lt} \times HW$ and multiplied to generate an embedding matrix about current information $I_c$ as follows:

$$I_c = \sigma(f(X'_t) \times g(X'_t)^T) \in \mathbb{R}^{N_{lt} \times c_{lt}}. \tag{2}$$

Here, $\sigma$ is a softmax function applied row-wise. This operation is similar to global pooling and region-based operation [27] in terms of making one representative value $I^{i,j}_c$. In detail, if the hexagon in Figure 4 (b) indicates the enhanced region by attention clue, and the information outside of the hexagon is suppressed. Therefore, we can compare a $i$th channel of $f(X'_t)$ and a $j$th channel of $g(X'_t)$ regarding target information along the whole $HW$ plane. If the two channels are similar, the resultant value of $I^{i,j}_c$ will be high (red pixel in Figure 4(b)); otherwise, it will be low (blue pixel). In other words, $I^{i,j}_c$ is the $(i, j)$ element of $I_c$, and it represents the $j$th information of the $i$th factor describing target's property. Finally, we have $N_{lt}$ factor which consist of $c_{lt}$ dimension about current target object.

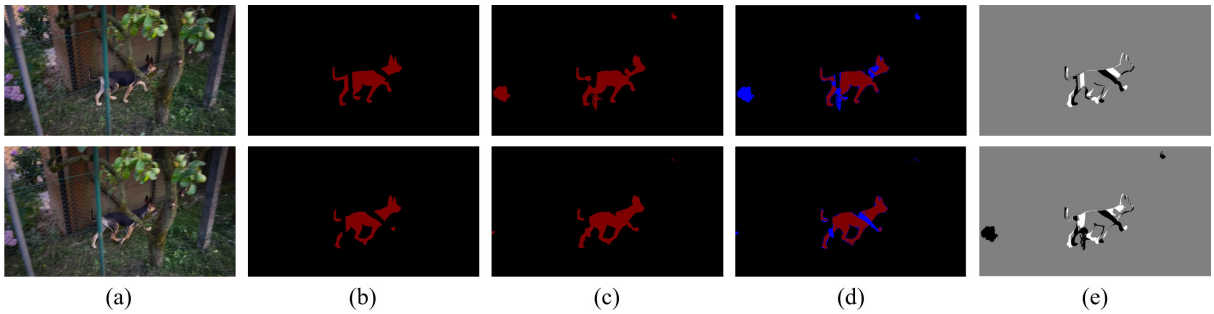



**FIGURE 5.** ((a)-(d)) Frame $t-1$ and $t$ from top to bottom. (a) Input image. (b) Ground truth. (c) Our result. (d) Estimated mask with color marking. Blue color means wrong segmentation result, and the blue region in frame $t$ is corrected from frame $t-1$. (e) Visualizing $\pi_{t,2}$. Top: $H_t - H_{t-1}$, Bottom: $H_t - \hat{H}_{t-1}$. $H_t - H_{t-1}$ cannot remove false positive region in the top of (c).

**TABLE 2.** The complexity and accuracy comparison between GC and ours when the input image size is 480 × 853. *Segmentation* and *Update* mean the requirement of FLOPS for executing segmentation without refinement stage, and updating a memory or a template. Our method reduces lots of computations for update process.

| | Segmentaiton | Update | #Param | $J\&F$ |
|---|---|---|---|---|
| GC | 37.9 G | 37.1 G | 38 M | 86.6 |
| Ours | 5.4 G | 0.06 G | 1.6 M | 81.1 |

The final long-term template $LT_{t+1}$ is updated by adaptive weight $w_t$, the embedding matrix $I_c$ and the previous template $LT_t$ as below:

$$LT_{t+1} = \frac{1}{1+w_t} LT_t + \frac{w_t}{1+w_t} I_c. \tag{3}$$

The $w_t$ determines how much the template is updated by current information, and it is calculated through the convolution block as shown in Figure 2. The convolution block for adaptive weight consists of average-pooling, convolution layer $f$ and sigmoid function $\sigma$ as follow:

$$w_t = \sigma(f_2(AvgPool(f_1(A_t))), \tag{4}$$

Here, we only apply ReLU activation function to $f_1$. When the template vector is similar to the input frame's query feature $q(X_{t+1})$, the resultant $S^{LT}_{t+1}$ value will be high (red pixel in Figure 4 (b)). Otherwise, it will be low (blue in Figure 4(b)) as mention in Section III-C.

The method of updating long-term template has some similarity to GC, but ours is computationally much cheaper by using attention clue. Specifically, GC re-extracts backbone features again from a new input which consists of current frame $f_t$ and estimated mask $\hat{M}_t$ for updating target information. However, re-extracting feature is huge burden for executing in real-time environment. In detail, when GC updates their memory, they need similar computation to conducting whole segmentation process as shown in Table 2. Our template method just combines the feature map and attention clue to mitigate inflow of irrelevant information, and thus our model can reuse the already produced backbone feature for updating process with simple calculation.

#### 1) INITIALIZATION FOR $LT_0$

In initialization stage, we generates $LT_0$ with a frame and a corresponding ground truth mask. We use same method for $P$, and fill zero for $\hat{H}_{t-2}$, and $\hat{H}_{t-1}$. For $\hat{H}_0$, we utilize ground truth mask. As following Eq 2, we construct $I_c$ and take it as $LT_0$.

### D. TRANSITION MAP FOR TEMPORAL CONSISTENCY

Our templates are constructed based on estimated results along the entire executed frames. Therefore, when the template is updated with incorrect results, it will gradually lead to incorrect tracking. In this case, if the model obtains the proper transition information on how to modify the wrong estimation of the previous frame to the true estimation for the current frame, the model can alleviate this error propagation problem. For this reason, we calculate a transition map $\hat{\pi}_t$, which contains pixel-wise conversion between background and foreground from $A_t$ as shown in Figure 2. The model simply produces the transition map with a single convolution layer from $A_t$. We add L2 loss as $L_{tm}$ by using the transition map $\hat{\pi}_t$ to encourage correction power and achieve consistency between neighboring frames:

$$\pi_t = H_t - \hat{H}_{t-1}, \quad L_{tm} = (\hat{\pi}_t - \pi_t)^2. \tag{5}$$

As a new learning target, we make a novel transition map from ground truth heatmap $H_t$ and previous estimated mask heatmap $\hat{H}_{t-1}$ as in Eq. (5). Note that the first and the second channel of $H_t$ are the probability of background and foreground from the ground truth mask of frame $t$, respectively. By Eq. (5), the range of all values of $\pi_t$ becomes $(-1, 1)$, and $\pi_t$ also consists of two channel feature map indicating transition tendency from $t-1$ to $t$. In detail, the first channel contains the transition tendency of the background, while the second is for the foreground. For example, $\pi^{h,w}_{t,2}$ means a value of $\pi_t$ in the $(h, w)$ position of second channel. If it is close to 1, it helps the estimated class at position $(h, w)$ to change into foreground from frame $t-1$ to $t$. On the other hand, if it is close to $-1$, it prevents the estimated class from turning to the foreground. Finally, when the value is close to 0, it keeps the originally estimated class of frame $t-1$ for a frame $t$ result.

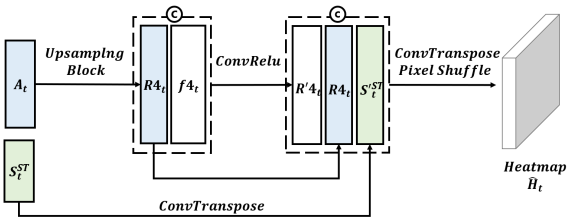



**FIGURE 6.** Structure of refinement network. © indicates concatenation of feature maps.

The reason why we use $\hat{H}_{t-1}$ instead of $H_{t-1}$ is illustrated in Figure 5. Figure 5 (b) shows ground truth masks, and (c) is the estimated masks at frame $t-1$ (top) and $t$ (bottom). First row of Figure 5 (e) is a visualization of $(H_t - H_{t-1})$ that guides the estimation to maintain the false positive region from the frame $t-1$ to $t$. Second row of Figure 5 (e) is a visualization of $(H_t - \hat{H}_{t-1})$ that guides the estimation to remove false positive region of the frame $t-1$. Figure 5 (d) is marked by blue color for denoting false estimation results comparing between (b) and (c). As shown in Figure 5 (d), the transition map $\pi_t$ helps reducing the false positive region from frame $t-1$ to $t$. With $L_{tm}$, the overall loss becomes:

$$Loss = CE(\hat{y}_t, y_t) + L_{tm}, \tag{6}$$

$CE$ denotes the cross entropy between the pixel-wise ground truth $y_t$ at frame $t$ and its predicted value $\hat{y}$.

### E. REFINEMENT NETWORK

The refinement network produces a final heatmap of the target object by repeatedly using ConvTranspose and PixelShuffle as shown in Figure 6. Here, ConvTranspose and PixelShuffle are two of the widely used upsampling methods. ConvTranspose pads or inserts zeros into the input feature map to carry out convolutions for upsampling. PixelShuffle rearranges the shape of the input feature map from $(C, H, W)$ to $(C/r^2, H \times r, W \times r)$, where $C, H, W$ and $r$ are the number of channels, the height and the width of the input feature map, and an upscaling factor, respectively. Firstly, $A_t$ is forwarded by upsampling block which consists of single convolution layer, ReLU and ConvTranspose to produce $R4_t$. Secondly, $R4_t$ is concatenated with $f4_t$ for adding more detailed local information to generate $R'4_t$. We also add $S'^{ST}_t$, which is upsampled by ConvTranspose from $S^{ST}_t$, for enhancing localization information. Finally, $R4_t$, $R'4_t$, and $S'^{ST}_t$ are concatenated together and upsampled by ConvTranspose for changing the number of channels and by PixelShuffle for avoiding the grid-effect problem. Then, a heatmap of target object is produced by calculating probability of background and foreground. In single object tracking scenario, we use the heatmap for final segmentation map, but in multiple object tracking case, we apply soft-aggregation to merge every heatmap for making the final segmentation map as follow [9], [13], [28].

## IV. EXPERIMENT

Here, we show various evaluations by using DAVIS benchmarks [38], [39]. DAVIS16 is a single object task consisting of 30 training videos and 20 validation videos, while DAVIS17 is a multiple object task with 60 training videos and 30 validation videos. We evaluate our model by using official benchmark code.[2] The DAVIS benchmark reports model accuracy by average of mean Jaccard index $J$ and mean boundary score $F$. $J$ index measures overall accuracy by comparing estimated mask and ground truth mask. $F$ score focuses on more contour accuracy by delimiting the spatial extent of the mask.

*Implementation Detail:* We use HRNetV2-W18-Small-v1 [40] for a lightweight backbone network and initialize it from the pre-trained parameters of the official code.[3] We freeze every backbone layer except the last block. The size of the smallest feature map is $1/32$ of the input image. We upsample the feature map and concatenate it with the second smallest feature map whose size is $1/16$ of the input image. We use ADAM optimizer for training our model. Firstly, we pre-train our model with synthetic video clips from image dataset, after then we train it with video dataset with single GPU following [9], [11]–[14].

*Pre-Train With Images:* We follow [10], [13], [14] pre-training method, which applies a random affine transformation to a static image for generating synthetic video clips. We use the saliency detection dataset MSRA10K [41], ECSSD [42], and HKU-IS [43] for various static images. Synthetic video clips consisting of three frames with a size of $240 \times 432$ are generated. We train 100 epochs with an initial learning rate of $2e^{-4}$ and a batch size of 48.

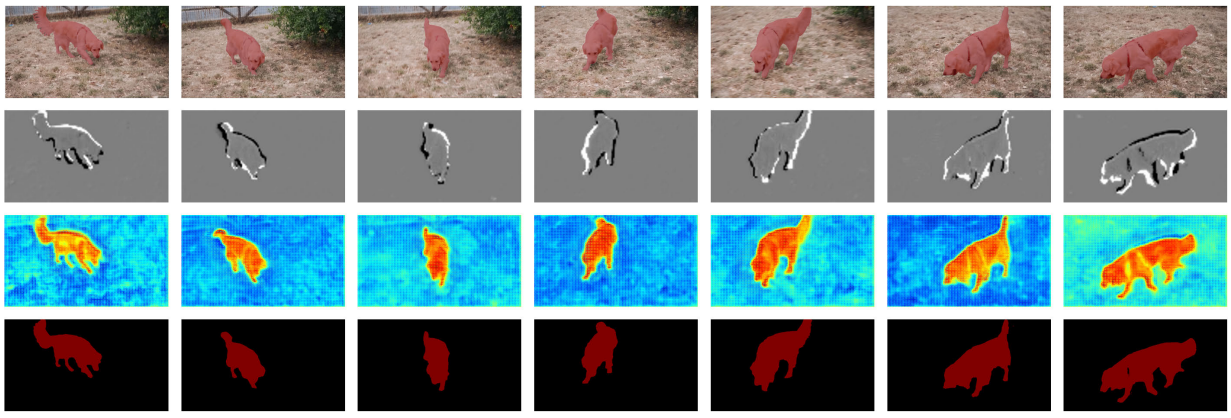
*Main-Train With Videos:* We use a two-stage training method; for the first 100 epochs, we only use Youtube-VOS [29], [30] with $240 \times 432$ image. We then train on the DAVIS17 [38], [39] dataset with $480 \times 864$ image for an additional 100 epochs. In both trainings, we use 8 consecutive frames for each batch, and the initial learning rate does not change. We set a batch size of 24 and an initial learning rate to $3e^{-4}$ for Youtube-VOS, and we use a batch size of 8 and an initial learning rate to $1e^{-4}$ for DAVIS.

### A. DAVIS BENCHMARK RESULT

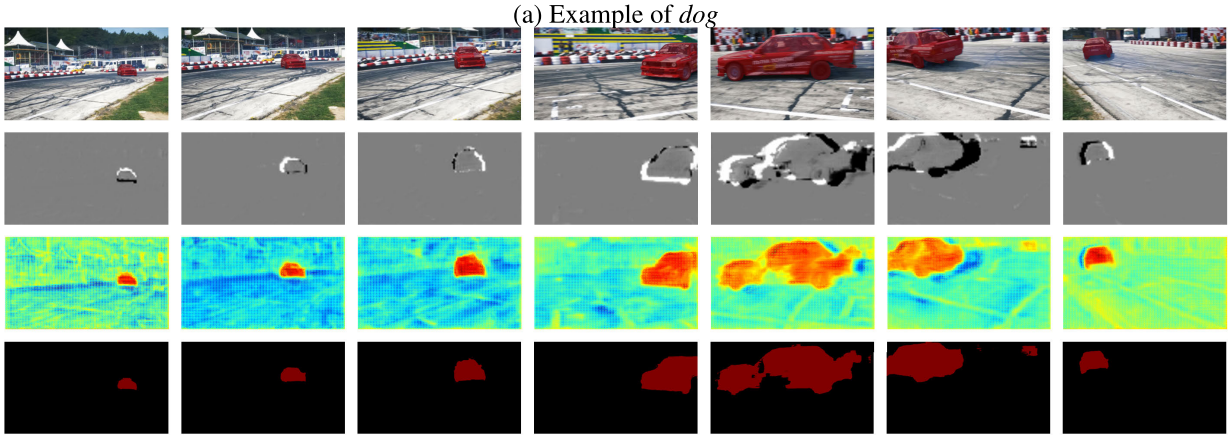
#### 1) COMPARISON TO STATE-OF-THE-ART

We compare our method with other recent models as shown in Table 3. We report backbone models and training datasets of each model for clarification because each model has a different setting. Our model shows better or competitive accuracy among fast models with more than 30 FPS speed for real-time applications. Specifically, SiamMask, which uses depth-wise style template matching method, is the fastest model among the previously proposed methods. Our model has better accuracy and speed than SiamMask

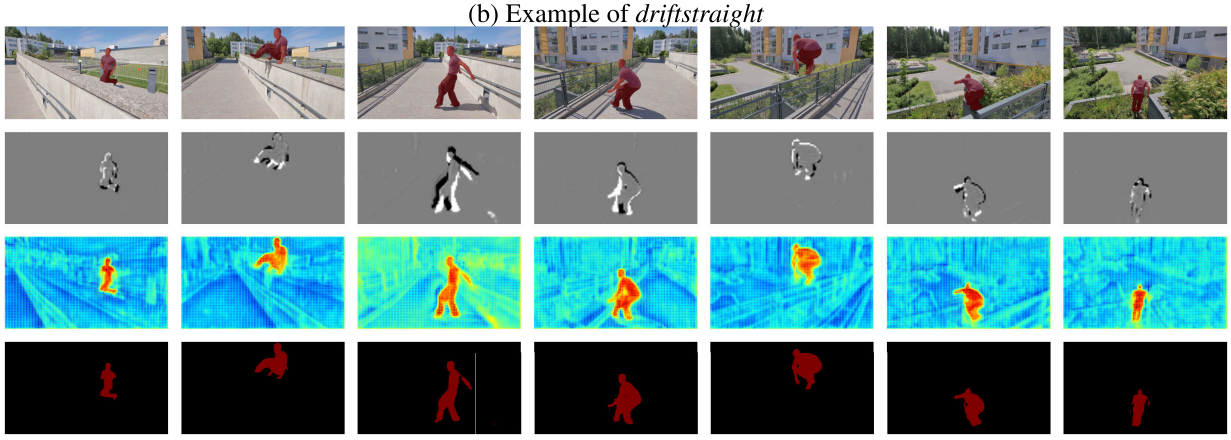
[2]https://github.com/davisvideochallenge/davis2017-evaluation
[3]https://github.com/HRNet/HRNet-Semantic-Segmentation



(a) Example of *dog*



(b) Example of *driftstraight*



(c) Example of *parkour*

**FIGURE 7.** Example of three videos. Each first row is the input overlapped with the ground truth mask. Each second row is the estimated transition map $\pi_t$. Here, white region means there are high possibility that the pixel has to change foreground and vice versa. Each third row is the activation map of blended results with two template matching $A_t$ Each fourth row is estimated mask of the video frame.

on both DAVIS16 and DAVIS17 benchmarks. Also, our model achieves better performance than FAVOS, OSMN, and RGMP. Also, in the DAVIS16 single video tracking case, ours show competitive accuracy with FRTM-VOS, A-GAME, and FEELVOS. The performance gap is 2.4%, 1.0%, and 0.6%, but our model's speed becomes 3.6×, 5.5×, and 10.2× faster, respectively. Therefore, our method achieves favorable performance among fast VOS models and reduces the performance gap from the online-learning and memory network-based models. Figure 7 shows various example for qualitative evaluation with activation map of blended feature map from two template matching results $A_t$, transition map $\hat{\pi}_t$ and estimated mask $\hat{y}_t$. The transition map results match the predicted mask, which shows that the transition map guides the model's training to improve temporal consistency.

**TABLE 3.** Quantitative comparison on DAVIS benchmark validation set. DAVIS17 is a multi-object scenario and DAVIS16 is a single-object scenario. YTB is using Youtube-VOS [29], [30] for training. Seg is segmentation dataset for pre-training by pascal [31] or COCO [32]. Synth is using saliency dataset for making synthetic video clip by affine transformation. Segflow uses customized feature extractor and additional dataset with KITTI [33], SIntel [34] and scene flow [35] dataset for a optical flow branch.

| | | Model Method | | | Train Dataset | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | Backbone | Online Learning | Template | Optical Flow | YTB | Seg | Synth | DV17 | DV16 | FPS |
| OnAVOS [36] | VGG16 | o | - | - | - | o | - | 67.9 | 85.5 | 0.08 |
| OSVOS-S [5] | VGG16 | o | - | - | - | o | - | 68.0 | 86.5 | 0.22 |
| FRTM-VOS [8] | ResNet101 | o | - | - | o | - | - | 76.7 | 83.5 | 21.9 |
| RGMP [28] | ResNet101 | - | o | - | - | o | o | 66.7 | 81.8 | 7.69 |
| CRN [20] | ResNet101 | - | - | o | - | o | - | - | 85.1 | 1.37 |
| SegFlow [21]* | ResNet101 | o | - | o | - | - | - | - | 83.1 | 0.12 |
| STM [13] | ResNet50 | - | o | - | o | - | o | 81.8 | 89.3 | 6.25 |
| GC [14] | ResNet50 | - | o | - | o | - | o | 71.4 | 86.6 | 25.0 |
| OSMN [37] | VGG16 | - | o | - | - | o | - | 54.8 | 73.5 | 7.69 |
| RANet [10] | ResNet101 | - | o | - | - | - | o | 65.7 | 85.5 | 30.3 |
| A-GAME [9] | ResNet101 | - | o | - | o | - | o | 70.0 | 82.1 | 14.3 |
| FEELVOS [11] | Xception 65 | - | o | - | o | o | - | 71.5 | 81.7 | 2.22 |
| SiamMask [12] | ResNet50 | - | o | - | o | o | - | 56.4 | 69.8 | 55.0 |
| **TTVOS (Ours)** | **HRNet** | **-** | **o** | **-** | **o** | **-** | **o** | **62.5** | **81.1** | **78.3** |

**TABLE 4.** Ablation study on DAVIS16 and DAVIS17. Short-term M, Long-term M means using short-term matching and long-term matching. Update long-term represents updating long-term template at every frame, and adaptive weight means whether using adaptive weight. Attention clue is using multiple of heatmaps and coordinate layers as clue, and × indicates not using any attention clue but re-extracts feature again with previous mask. Transition map means using auxiliary loss.

| Exp | Short-term M | Long-term M | Adaptive Weight | Update Long-term | Attention Clue | Transition Map | DV17 | DV16 |
|---|---|---|---|---|---|---|---|---|
| 1 | o | - | - | - | - | - | 57.0 | 75.9 |
| 2 | - | o | - | o | - | - | 54.5 | 78.8 |
| 3 | o | o | - | o | - | - | 57.5 | 77.1 |
| 4 | o | o | - | o | - | o | 58.7 | 79.5 |
| 5 | o | o | o | - | o | o | 61.9 | 78.4 |
| 6 | o | o | - | o | o | o | 61.8 | 80.0 |
| 7 | o | o | o | o | × | o | 57.3 | 76.2 |
| 8 | o | o | o | o | o | o | 62.5 | 81.1 |

### B. ABLATION STUDY

To prove our proposed methods, we perform an ablative analysis on DAVIS16 and DAVIS17 benchmark as shown in Table 4 and visualize the efficacy of several methods as shown in Figure 8. *Short-term M, and Long-term M* mean using short-term matching and long-term matching. When we do not use the short-term or long-term matching, we replace the original matching method by concatenating the previous mask heatmap and the current feature map. After then the concatenated feature map is forwarded by several convolution layers. *Update Long-term* represents updating long-term template at every frame, and if not used, the model never updates the long-term template. *Adaptive Weight* means whether using adaptive weight or using decaying weight at a fixed rate as like GC for updating long-term template. *Attention Clue* means our proposed clue by using multiple heatmaps and coordinate layers, and if not used, the model uses only previous heatmap $\hat{H}_{t-1}$ as clue. Also, × indicates that model does not use any attention clue but re-extracts feature again with previous mask for updating long-term template. In detail, the model extracts feature from predicted mask with a single convolution layer, batch-normalization and ReLU for making a feature map of same size as a backbone feature map. After then, the two feature maps are added together for enhancing only nearby regions from previous target locations like STM. Finally, the merged feature map is forwarded into remained layers to end feature extraction stage. *Transition Map* denotes adding auxiliary loss for improving temporal consistency. Figure 8 shows several examples of each module's efficacy with three videos (1st-2nd rows, 3th-4th rows, 5th-6th rows). Each video's first row is the early frame of the video and the second row is the later frame of the video.

We find that each matching method has a different responsibility for what we desire. The short-term matching helps maintain objects ID from localization clue, and long-term matching improves mask quality by enhancing the fine details. For example, Exp1 keeps object ID but fails to make an accurate mask for rider and motorbike, as shown in Figure 8 (c) 5th-6th rows. On the contrary, Exp2 makes an accurate shape but loses green-object ID (motorbike) as shown in Figure 8(d) 6th row. Exp2 shows performance degradation on multi-object tracking task (DAVIS17) due to failure in maintaining object ID, even it generates more accurate masks than Exp1. Even Exp2 fails to keep green-object ID in the early frame as shown in 8(d) the first row (human) and the third row (box). Therefore, Exp1 achieves better performance in DAVIS17, and Exp2 shows high accuracy in DAVIS16 as shown in Table 4. Exp3 gets every advantage

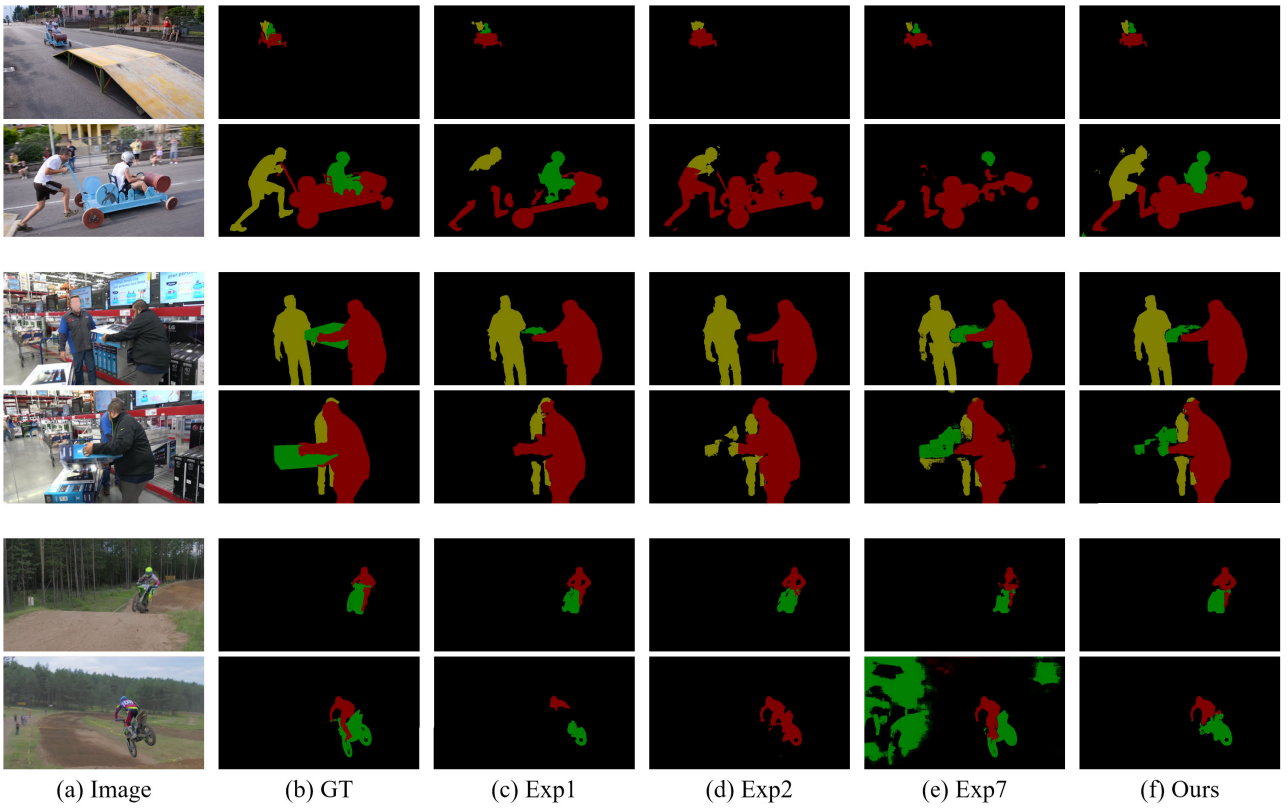



**FIGURE 8.** Three examples of video regarding various ablation study with early frames and later frames in video. 1st and 2nd rows *soapbox* example of ablation study for frame 3 and 78 from top to bottom. 1st and 2nd rows *loading* example of ablation study for frame 4 and 41 from top to bottom. 1st and 2nd rows *motocross-jump* example of ablation study for frame 1 and 35 from top to bottom. (a) Input frame. (b) Ground truth. (c) Using only short-term matching (Exp1). (d) Using only long-term matching (Exp2). (e) Without attention clue for updating long-term template (Exp7). (f) Our proposed method (Exp8).

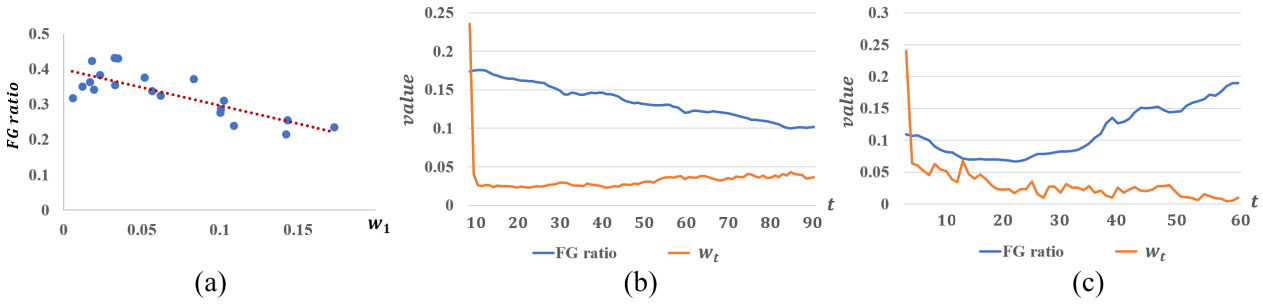



**FIGURE 9.** (a) Scatter plot about foreground (FG) ratio versus scale of weight. (b) and (c) are the plots about FG ratio and weight at each frame. The size of object is decreased in (b), but the size of object is increased in (c) over time.

from both template matching methods, and Exp4[4] uses auxiliary loss with transition map to improve both accuracies.

Exp5-Exp7 explain the importance of updating method for the long-term template. Exp5 is using the long-term template, but the template is never updated, and Exp6 conducts an update stage at every frame without adaptive weight. In detail, the update weight is decayed with fixed rate over time following the proposed method in GC. Firstly, even though Exp5 does not update the long-term template, the performance of DAVIS17 is increased a lot by 3.2% than Exp4 with the proposed attention clue. However, the accuracy of DAVIS16 is lower than Exp4, and it means the model fails to generate an accurate mask. In the case of Exp6, using naive update weight shows improving DAVIS16 accuracy with a similar performance of DAVIS17 from Exp5. In the case of Exp6, using naive updating weight improves DAVIS16 accuracy, while it shows similar performance in DAVIS17 from Exp5. Therefore, we can deduce two characteristics.

[4]This is result of previous work TTVOS [1].

**TABLE 5. DAVIS17 and DAVIS16 results by additionally applying auxiliary loss with transition map.**

| | Backbone | DV17 | DV16 |
|---|---|---|---|
| FRTM-VOS [8] | ResNet101 | 76.7 | 83.5 |
| | ResNet18 | 70.2 | 78.5 |
| with auxiliary loss | ResNet101 | 76.6 | 85.2 |
| | ResNet18 | 71.8 | 82.0 |

**TABLE 6. Additional experiments with using different backbone. MobileV3L denotes MobileNetV3-Large. DV17 and DV16 are DAVIS17 and DAVIS16, respectively. #FLOPs and #Param indicate the number of FLOPs and parameters.**

| Backbone | #FLOPs | #Param | DV17 | DV16 | FPS |
|---|---|---|---|---|---|
| HRNet | 10.6 | 1.61 | 62.5 | 81.1 | 78.3 |
| ResNet18 | 55.2 | 12.5 | 67.5 | 81.4 | 82.6 |
| ResNet50 | 83.9 | 14.8 | 69.5 | 83.1 | 37.7 |
| MobileV3L | 7.56 | 3.66 | 62.9 | 79.8 | 74.6 |

First, using multiple heatmaps and position information as attention clue is helpful to enhance the localization power of model. Second, updating the long-term template can handle an object's shape-changing to generate a more accurate mask. Accuracy degradation in Exp7 proves that concatenating attention clue is an adequate method for maintaining a proper template without heavy computation of re-extracting features. We follow the STM method, which extracts features again, but the result is lower than ours. Also, re-extracting feature increases FLOPs a lot, and this incurs tricky problem to apply real-environments. Finally, ours (Exp8) does not lose object ID and generate delicate masks with high performance on both benchmarks.

### C. ANALYSIS OF ADAPTIVE WEIGHTS

We conduct further analysis about adaptive weight. We find that the scale of weight is inversely proportional to the ratio of foreground (FG) region in the input frame. Figure 9 (a) is a scatter plot about the FG ratio versus the scale of the first frame's weight, which shows an negative correlation between two factors. Figure 9 (b) and (c) are scale of FG ratio and weight for each frame. The weight for the first frame is significantly higher than the weights of other frames. The trend of weight is different depending on the change in the FG ratio. The ratio of FG is gradually getting smaller over time as shown on Figure 9 (b), but the ratio in Figure 9 (c) is progressively larger over time. Therefore, the scale of weight in Figure 9 (b) is steadily increased, but the scale of weight in Figure 9 (c) behaves inversely. We conjecture that the amount of suppression about feature vectors is far from the FG region by attention clue. We conjecture that the amount of suppression, which is for feature vectors not activated by attention clue, is related to the scale of weight. In detail, when the object is small, the quantity of information is smaller than the large object case. To make balanced quantity across frames, the model derives a large scale of weight for amplifying current information.

### D. IMPROVING ACCURACY BY TRANSITION MAP

We conduct further experiments for proving the efficacy of our auxiliary loss by transition map with FRTM-VOS, which is one of the fast online-learning methods, using ResNet101 and ResNet18 for the backbone network. We implement our proposed loss function based on FRTM-VOS official code[5] and follow their training strategy. Our proposed loss is more useful in the lightweight backbone network (ResNet18) as shown in Table 5. When we apply our loss to the ResNet101 model, the accuracy on DAVIS17 decreased slightly by 0.1%, but it increased 1.7% on DAVIS16. In the ResNet18 model, we improve the accuracy a lot on both DAVIS17 and DAVIS16. We conjecture that using our loss not only improves mask quality but also resolves a problem of overfitting due to fine-tuning by a given condition.

### E. DIFFERENT BACKBONE EXPERIMENTS

We conduct further experiments with multiple backbone networks (ResNet18, ResNet50, MobileNetV3-Large) to compare accuracy and complexity as shown in Table 6. We initialize the backbone network from the pre-trained model and freeze every block except the last block of each backbone network. For ResNet50, we do not use the original four-block structure but use only a three-block structure like STM and GC. Ours with ResNet50 shows lower accuracy by 12.3 (DAVIS17) and 6.2 (DAVIS16), but ours has 6× faster speed than STM. Similarly, this model has lower accuracy by 1.9 (DAVIS17) and 3.5 (DAVIS16) but achieves 1.5× faster than GC. The ResNet18 model has better performance than HRNet, and even it takes a large number of FLOPs. We conjecture that the HRNet has a multiple-branch structure, and the structure is usually harmful to real model speed and not friendly to GPU parallel computing. Likewise, the MobileNetV3-Large model has fewer FLOPs than HRNet, but the model is slow. Most layers in MobileNet consist of depth-wise separable convolution. This has fewer FLOPs than standard convolution, but it has slow executed time due to not friendly structure for GPU computation.

## V. CONCLUSION

Many semi-VOS methods have improved accuracy, but they are hard to utilize in real-world applications due to tremendous complexity. To resolve this problem, we propose a novel lightweight semi-VOS model consisting of short-term and long-term matching modules. The short-term matching enhances localization, while long-term matching improves mask quality by handling an object's shape-changing with adaptive weight. However, using previously estimated results incurs an error-propagation problem. To mitigate this problem, we also devise auxiliary loss, which guides the model to learn correction of false estimated regions with

[5]https://github.com/andr345/frtm-vos

transition maps. Finally, Our model achieves fast inference time while reducing the performance gap from heavy models.

One of the limitations of the proposed method is that a target may be lost when the target is occluded from other objects for a long time. When the target object disappears due to occlusion, many regions in the previous mask become zeros. It implies that we cannot give attention clues for the next frame, and the model has to find the target in a broader range. We observe that it may cause performance degradation. One of our research directions might be designing a more robust method to handle this problem in environments with severe occlusions.

## REFERENCES

[1] H. Park, G. Venkatesh, and N. Kwak, "TTVOS: Lightweight video object segmentation with adaptive template attention module and temporal consistency loss," in *Proc. 1st Int. Res. Symp. Tiny Mach. Learn. (TinyML)*, Mar. 2021.

[2] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online deep learning: Learning deep neural networks on the fly," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2660–2666.

[3] G. Zhou, K. Sohn, and H. Lee, "Online incremental feature learning with denoising autoencoders," in *Proc. Artif. Intell. Statist.*, 2012, pp. 1453–1461.

[4] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[5] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "Video object segmentation without temporal information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1515–1530, Jun. 2019.

[6] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2663–2672.

[7] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixe, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 221–230.

[8] A. Robinson, F. J. Lawin, M. Danelljan, F. S. Khan, and M. Felsberg, "Learning fast and robust target models for video object segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7406–7415.

[9] J. Johnander, M. Danelljan, E. Brissman, F. S. Khan, and M. Felsberg, "A generative appearance model for end-to-end video object segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8953–8962.

[10] Z. Wang, J. Xu, L. Liu, F. Zhu, and L. Shao, "RANet: Ranking attention network for fast video object segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3978–3987.

[11] P. Voigtlaender, Y. Chai, F. Schroff, H. Adam, B. Leibe, and L.-C. Chen, "FEELVOS: Fast end-to-end embedding learning for video object segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9481–9490.

[12] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1328–1338.

[13] S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim, "Video object segmentation using space-time memory networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 9226–9235.

[14] Y. Li, Z. Shen, and Y. Shan, "Fast video object segmentation using the global context module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 735–750.

[15] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10759–10768.

[16] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.

[18] F. Lin, Y. Chou, and T. Martinez, "Flow adaptive video object segmentation," *Image Vis. Comput.*, vol. 94, Feb. 2020, Art. no. 103864.

[19] W. Wang, J. Shen, F. Porikli, and R. Yang, "Semi-supervised video object segmentation with super-trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 985–998, Apr. 2019.

[20] P. Hu, G. Wang, X. Kong, J. Kuen, and Y.-P. Tan, "Motion-guided cascaded refinement network for video object segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1400–1409.

[21] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "SegFlow: Joint learning for video object segmentation and optical flow," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 686–695.

[22] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2020, pp. 402–419.

[23] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid data dreaming for object tracking," in *Proc. CVPR Workshop*. New York, NY, USA: IEEE, 2017.

[24] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.

[25] S. Cho, M. Cho, T.-Y. Chung, H. Lee, and S. Lee, "Crvos: Clue refining network for video object segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 2301–2305.

[26] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 116–131.

[27] H. Caesar, J. Uijlings, and V. Ferrari, "Region-based semantic segmentation with end-to-end training," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2016, pp. 381–397.

[28] S. W. Oh, J.-Y. Lee, K. Sunkavalli, and S. J. Kim, "Fast video object segmentation by reference-guided mask propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7376–7385.

[29] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, "YouTube-VOS: Sequence-to-sequence video object segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 585–601.

[30] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "YouTube-VOS: A large-scale video object segmentation benchmark," 2018, *arXiv:1809.03327*. [Online]. Available: http://arxiv.org/abs/1809.03327

[31] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2014.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2014, pp. 740–755.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[34] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. 12th Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2012, pp. 611–625.

[35] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4040–4048.

[36] P. Voigtlaender and B. Leibe, "Online adaptation of convolutional neural networks for video object segmentation," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–13.

[37] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos, "Efficient video object segmentation via network modulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6499–6507.

[38] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 724–732.

[39] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 Davis challenge on video object segmentation," 2017, *arXiv:1704.00675*. [Online]. Available: http://arxiv.org/abs/1704.00675

[40] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Apr. 1, 2020, doi: 10.1109/TPAMI.2020.2983686.
[41] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 569–582, Mar. 2015.
[42] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1155–1162.
[43] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5455–5463.

**HYOJIN PARK** received the B.S. degree in industrial engineering from Ajou University, Suwon, South Korea, in 2013, and the M.S. degree in the robotics program from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2016. She is currently pursuing the Ph.D. degree in intelligent systems with the Graduate School of Convergence Science and Technology, Seoul National University, Seoul, South Korea. Her research interests include image segmentation, semi-supervised video object segmentation, and generative model and efficient model design for real-world application.

**JAYEON YOO** received the B.S. degree in industrial management system from POSTECH, Pohang, South Korea, in 2016. She is currently pursuing the Ph.D. degree in the Department of Intelligence and information, Seoul National University, Seoul, South Korea. Her research interests include semi-supervised video object segmentation and domain adaptation for object detection and segmentation.

**GANESH VENKATESH** received the Ph.D. degree in computer science from the University of California San Diego. He has published early work demonstrating dark silicon and using specialization to address the power wall with the University of California San Diego. At Nvidia, he contributed/lead the effort on multiple headliner features, including 4-bit inference in turing and sparse tensor support in ampere. He worked at Intel labs on building accelerators for machine learning as well as developing techniques to enable 2-bit weight CNNs. He currently works at Facebook Reality Labs, where he leads the effort on deploying visual and assistant experiences on augmented reality glasses.

**NOJUN KWAK** (Senior Member, IEEE) was born in Seoul, South Korea, in 1974. He received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, in 1997, 1999, and 2003, respectively. From 2003 to 2006, he was with Samsung Electronics, Seoul. In 2006, he joined Seoul National University as a BK21 Assistant Professor. From 2007 to 2013, he was a Faculty Member with the Department of Electrical and Computer Engineering, Ajou University, Suwon, South Korea. Since 2013, he has been with the Graduate School of Convergence Science and Technology, Seoul National University, where he is currently a Professor. His current research interests include feature learning by deep neural networks and their applications in various areas of pattern recognition, computer vision, and image processing.