Online Video Object Segmentation via Convolutional Trident Network

Won-Dong Jang
Korea University
wdjang@mcl.korea.ac.kr

Chang-Su Kim
Korea University
changsukim@korea.ac.kr

Abstract

A semi-supervised online video object segmentation algorithm, which accepts user annotations about a target object at the first frame, is proposed in this work. We propagate the segmentation labels at the previous frame to the current frame using optical flow vectors. However, the propagation is error-prone. Therefore, we develop the convolutional trident network (CTN), which has three decoding branches: separative, definite foreground, and definite background decoders. Then, we perform Markov random field optimization based on outputs of the three decoders. We sequentially carry out these processes from the second to the last frames to extract a segment track of the target object. Experimental results demonstrate that the proposed algorithm significantly outperforms the state-of-the-art conventional algorithms on the DAVIS benchmark dataset.

1. Introduction

Video object segmentation aims at clustering pixels in videos into objects or background. In general, video object segmentation algorithms can be grouped into three categories: unsupervised, semi-supervised, and supervised ones. Unsupervised algorithms [11, 27, 28, 33, 35, 53] do not require any annotations about objects. Instead of annotations, they discover primary objects in videos using objectness, saliency, and motion cues. While some algorithms [11, 35, 53] yield a single segment track, the others [27, 28, 33] produce multiple segment tracks. Semi-supervised algorithms [15, 37, 40, 54] track and segment a target object (or foreground), which is annotated by a user in the first frame. Supervised algorithms [1, 12, 13] take user annotations interactively during the segmentation process. Although they yield a fine segment track, the annotation tasks can be burdensome for users.

With explosive researches in deep learning, there are remarkable advances in many vision problems, *e.g.*, object detection [18], contour detection [57], and semantic segmentation [7]. The encoder-decoder architecture [20] is widely used in deep learning systems [7, 29, 32, 38, 57]. The encoder extracts features from an input image. The convolutional neural networks [18, 25, 45, 47], trained for image classification, are fine-tuned and used as the encoder in many cases. Since image classification attempts to identify the class of an object in an image [43], the trained networks extract high-level features effectively. The decoder design varies more according to applications. For example, pixel-level classification algorithms (*e.g.* contour detection [57] and semantic segmentation [32]) adopt convolution layers with unpooling layers in the decoder. On the contrary, fully connected layers are used for image-level classification problems (*e.g.* image classification [45] and object detection [41]). In this work, we design decoders for semi-supervised video object segmentation. To the best of our knowledge, this is the first deep learning-based approach to semi-supervised video object segmentation.

We propose a semi-supervised online segmentation algorithm, which can separate a target object from the background sequentially from the first to last frames with minimum user efforts at the first frame only. To track and segment the target object, we develop the convolutional trident network (CTN), which has the encoder-decoder architecture. The CTN outputs three probability maps from three decoding branches: separative, definite foreground, and definite background decoders. First, we propagate a segmentation label map from the previous frame to the current frame. We then predict the three probability maps via the CTN. The three maps are tailored for a two-class Markov random field (MRF) optimization problem. At the beginning of the MRF optimization, we assign initial pixel labels by thresholding the separative probability map. Also, we use the definite foreground and background maps to discover the pixels that should be definitely labeled as the foreground and the background, respectively. By fixing the labels on these definite pixels, the MRF optimizer extracts the target object more precisely. We perform these processes from the second to the last frames. Experimental results show that the proposed algorithm outperforms the state-of-the-art conventional algorithms [12, 31, 37, 40] on the DAVIS dataset [36]. Three major contributions of this work are:

- ▷ Development of the effective CTN, which yields three tailored probability maps for the MRF optimization.
- ▷ Implementation of a fast online algorithm for segmenting long videos in practical applications.
- ▷ Remarkable performance on the DAVIS benchmark dataset, which consists of challenging videos.

2. Related Work

2.1. Unsupervised Video Object Segmentation

Unsupervised video object segmentation is a task to extract segment tracks of primary objects in a video. A primary object appears frequently across frames in a video. Inspired by this, early approaches [4, 14, 34, 44] perform motion segmentation to yield sparse point trajectories, and transform the sparse trajectories into dense segment tracks using converting methods such as [14, 33]. Instead of the point tracking, [27, 28, 30, 59] connect object proposals [9] across frames in a video.

To discover visually important objects in a video, [11, 35, 53] employ saliency detection techniques. Papazoglou and Ferrari [35] compute a motion saliency map, called the inside-outside map, using optical flow boundaries. Faktor and Irani [11] exploit both motion saliency and visual saliency to separate a segment track from the background. Wang *et al.* [53] first compute a spatiotemporal saliency map using geodesic distances, and then perform energy minimization based on a global appearance model and a per-frame location model.

Recently, Jang *et al.* [23] delineate a primary object in a video based on the alternate convex optimization of the foreground and background distributions. Xiao and Lee [55] first generate box tracks containing objects, and then apply a pixel-wise segmentation scheme to these boxes. Bideau and Learned-Miller [2] discover differently moving objects by considering angles and magnitudes of optical flow vectors.

2.2. Semi-Supervised Video Object Segmentation

In semi-supervised video object segmentation, a target object is identified by a user at the first frame, and then tracked automatically in subsequent frames. Note that semi-supervised algorithms can be categorized into offline or online ones. Offline algorithms separate the target object from the background in all frames simultaneously. On the contrary, online (or streaming) algorithms perform the task of tracking and segmentation sequentially from the second to the last frames. While offline algorithms require a huge memory space for a long video, online ones use a fixed space regardless of the video duration.

Offline Algorithm: Tsai *et al.* [49] construct a volumetric graph whose nodes are pixels in all frames, and perform MRF optimization using a target appearance model and a spatiotemporal coherency model. Jain and Grauman [22] first generate temporal superpixels in a video, and then define an energy function to be minimized. In the minimization, two pixels are encouraged to have the same label (foreground or background) if they belong to the same temporal superpixel. Perazzi *et al.* [37] train a support vector machine classifier by employing bounding boxes of a target object as training samples, and then select object proposals that are highly similar to the target object.

Online Algorithm: Chockalingam *et al.* [6] partition a target object into fragments and represent it with a Gaussian mixture model (GMM), which is used to trace the target object in subsequent frames. Chang *et al.* [5] divide a video into temporal superpixels, and achieve video object segmentation by selecting the superpixels that overlap with the target object at the first frame. Ramakanth and Babu [40] propagate labels from the previous frame to the current frame using video seams. Varas and Marques [51] perform the co-clustering between partitions in the previous frame and the current frame to address object deformation. Wen *et al.* [54] over-segment each frame into superpixels to construct multi-part models of a target object. They trace the target object based on the inter-frame matching, and update the multi-part models iteratively. Märki *et al.* [31] construct a grid model using color and location features of a target object and the background. The grid model at the previous frame is applied to the current frame to delineate the target object. Tsai *et al.* [50] perform semi-supervised video object segmentation and optical flow refinement jointly.

Instead of an object mask, online tracking-by-segmentation algorithms [8, 15, 52] accept an object box as user input. In [8, 15], the segmentation of a target object in a box is achieved by a Hough voting. Wang *et al.* [52] generate a superpixel-based appearance model to compute confidence maps, which are then converted into segmentation results based on adaptive thresholding.

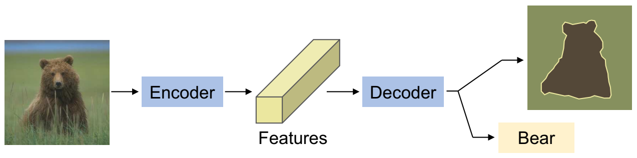

Figure 1. The encoder-decoder architecture [20] for solving vision problems. In this particular example, semantic segmentation is performed by pixel-level classification, and the object class ‘Bear’ is predicted by image-level classification.

2.3. Encoder-Decoder Architecture

Many computer vision problems are addressed based on the encoder-decoder architecture [20], which is illustrated in Figure 1. The encoder extracts features from an input image. Then, for example, the decoder can perform pixel-

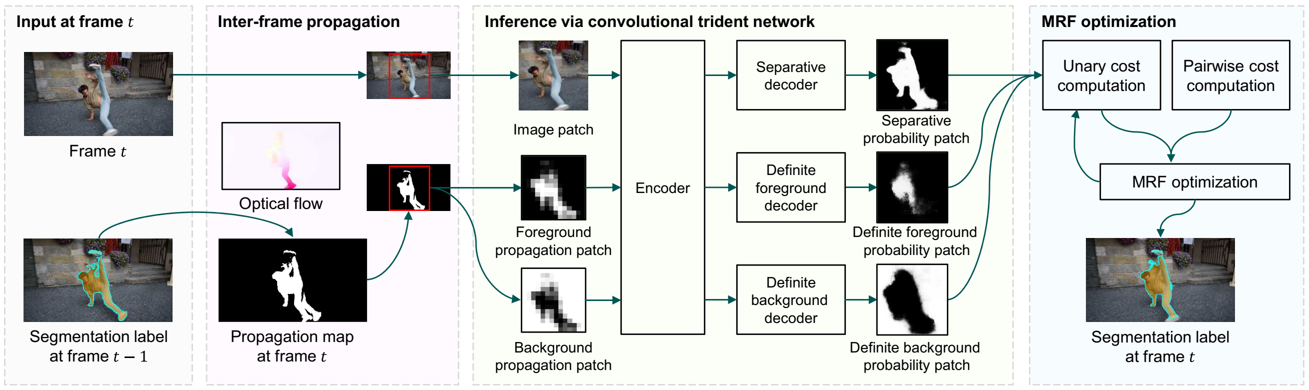
Input at frame t
Frame t
Segmentation label at frame $t-1$
Inter-frame propagation
Optical flow
Propagation map at frame t
Inference via convolutional trident network
Image patch
Foreground propagation patch
Background propagation patch
Encoder
Separative decoder
Definite foreground decoder
Definite background decoder
Separative probability patch
Definite foreground probability patch
Definite background probability patch
MRF optimization
Unary cost computation
Pairwise cost computation
MRF optimization
Segmentation label at frame t

Figure 2. Overview of the proposed algorithm. We perform this process from the second to the last frames sequentially.

level classification or image-level classification using the extracted features. Noh *et al.* [32] perform category-wise semantic segmentation by constructing a decoder, which is symmetrical to the VGG-16 net encoder [45]. Inspired by this, Yang *et al.* [57] train a decoder to detect object contours using a small number of layers. Liu and Han [29] propose a saliency detection algorithm, which predicts a multi-scale saliency map at each layer in a decoder. For each encoded patch, Pinheiro *et al.* [38] generate an object proposal by predicting a segmentation mask and an objectness score. Dai *et al.* [7] develop multi-task decoders, which share the same encoded features, for instance-aware semantic segmentation.

3. Proposed Algorithm

We propose a semi-supervised online video object segmentation algorithm, which yields a segment track of a target object, annotated by a user at the first frame. The annotation can be performed by employing interactive image segmentation techniques [16, 42, 56].

Figure 2 is an overview of the proposed algorithm. First, given the segmentation label map for the previous frame $I^{(t-1)}$, we propagate it to the current frame $I^{(t)}$ using optical flows. Then, the proposed CTN, which has the encoder-decoder architecture, produces three probability maps: separative, definite foreground, and definite background maps. These maps are tailored for the next MRF optimization step, which computes the segmentation label map for the current frame $I^{(t)}$ using unary and pairwise costs. We execute this process from the second frame $I^{(2)}$ to the last frame $I^{(T)}$ to yield a segment track.

3.1. Propagation of Segmentation Labels

We exploit the segmentation label map for $I^{(t-1)}$ to roughly locate the target object in the current frame $I^{(t)}$. To this end, we first compute backward optical flow vectors from $I^{(t)}$ to $I^{(t-1)}$. In implementation, we adopt one of the two optical flow techniques [19, 26]; whereas [19] provides more accurate optical flow information, [26] requires much lower computational complexity. For pixel $\mathbf{p} = [x, y]^T$, we propagate the segmentation label from $I^{(t-1)}$ to $I^{(t)}$ by

$$H^{(t)}(\mathbf{p}) = S^{(t-1)}(x + u_{\text{b}}^{(t)}(\mathbf{p}), y + v_{\text{b}}^{(t)}(\mathbf{p})) \qquad (1)$$

where $[u_{\text{b}}^{(t)}(\mathbf{p}), v_{\text{b}}^{(t)}(\mathbf{p})]^T$ is the backward optical flow vector of $\mathbf{p}$ in $I^{(t)}$ to $I^{(t-1)}$, and $H^{(t)}$ is the propagation map for $I^{(t)}$. $S^{(t-1)}$ is the segmentation label map for $I^{(t-1)}$, whose element is 1 if the corresponding pixel belongs to the foreground, and 0 otherwise. Figure 3 illustrates the inter-frame propagation of segmentation labels. We see that the target object is roughly estimated.

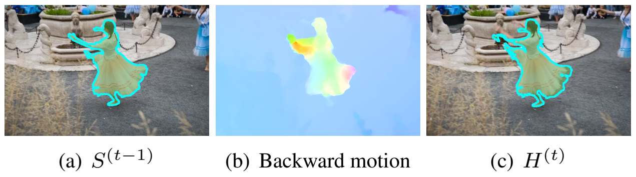

Figure 3. Inter-frame propagation of a segmentation label map. The segmentation label map $S^{(t-1)}$ for frame $t-1$ in (a) is propagated by employing the backward optical flow vectors in (b), to generate the propagation map $H^{(t)}$ in (c)

3.2. Inference via Convolutional Trident Network

The propagation map may be inaccurate due to object deformation, motion blur, occlusion, and optical flow errors. Hence, we infer segmentation information via the CTN, which has the encoder-decoder architecture, to consider high-level features of the target object. The inferred information is effectively used to solve a binary labeling problem as will be discussed in Section 3.3. Figure 4 shows the architecture of the proposed network.

Network Architecture: The encoder extracts features from a $224 \times 224 \times 3$ input image patch. We choose the VGG-16 net [45] as the encoder, which consists of 13 convolution layers, 3 fully connected layers, and 5 max-pooling layers.

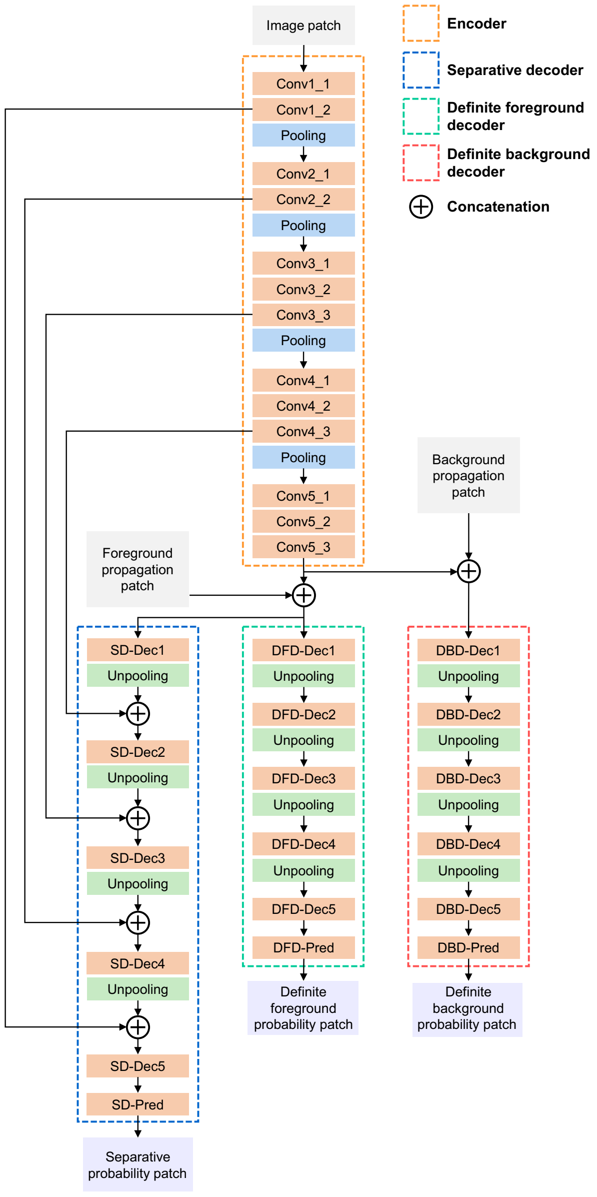

Figure 4. Architecture of the proposed convolutional trident network (CTN) for semi-supervised online video object segmentation.

We only use the layers up to the 13 convolution layers, as in [7,29,41]. While early layers encodes low-level features, later ones characterize high-level attributes.

We draw segmentation inferences from the encoded features using three decoders: separative decoder (SD), definite foreground decoder (DFD), and definite background decoder (DBD). These three decoders provide functional information, respectively, for the target object segmentation, which is the binary labeling problem of the foreground and the background. The decoders have unpooling layers [58] and convolution layers. The unpooling layers guide the encoded features to be decoded in the original image patch size. In this work, all unpooling layers enlarge their input patches by a factor of 2 horizontally and vertically.

The goal of the SD is to separate a target object from the background. First, we concatenate the encoded $14 \times 14 \times 512$ feature, 'Conv5_3,' and the 14×14 foreground propagation patch, which is resized from the propagation map $H^{(t)}$. Due to the resizing into a rather smaller size, the foreground propagation patch loses details. This is, however, acceptable since the inter-frame propagation is prone to errors and can be misleading. We make the decoder robust to those errors, by employing the smaller but more reliable patch. Note that the resizing is a kind of low-pass filtering, which reduces high-frequency noisy components. Then, we feed the concatenated data into a convolution layer ('SD-Dec1') and an unpooling layer. Pinheiro *et al.* [39] showed that the intermediate outputs of the encoder, 'Conv1_2,' 'Conv2_2,' 'Conv3_3,' and 'Conv4_3,' can improve segmentation qualities in the decoder. Inspired by this, we concatenate the output of the unpooling layer and 'Conv4_3,' and pass it through a convolution layer ('SD-Dec2') and an unpooling layer again. After repeating this sequential process of concatenation, convolution, unpooling two more times, we perform concatenation and convolution ('SD-Dec5') once more and then use a prediction layer ('SD-Pred') to yield the separative probability patch, whose elements have high probabilities on foreground regions.

In segmentation, fixing labels in definite pixels improves labeling accuracies. Definite pixels indicate locations that should be labeled as the foreground or the background indubitably. However, it is hard to decide where to fix the labels. Hence, we develop the DFD and DBD to discover definite pixels. The DFD identifies definite foreground pixels. We set the concatenated data, which is used in the SD, as the input again. Then, we feed the concatenated input data into convolution layers ('DFD-Dec1$\sim$5') and unpooling layers alternately. In the definite decoders, it is important to determine only indubitable pixels, instead of high quality object boundaries. Therefore, the definite decoders do not use the intermediate outputs of the encoder. The last convolution layer, 'DFD-Pred,' produces the definite foreground probability patch that represents the probability of each pixel to be a definite foreground one. On the contrary, the DBD finds definite background pixels. We first invert the foreground propagation patch to compute the 14×14 background propagation patch. Then, we concatenate the encoded feature 'Conv5_3' and the background propagation patch to form the input to the DBD, which has the same architecture as the DFD. The DBD yields the definite background probability patch.

In each decoder, the final prediction layer consists of a convolution layer and a sigmoid layer. The sigmoid layers make the decoders to yield normalized outputs within [0, 1].

Table 1. Specification of the network decoders. We use the same kernel settings and the normalization strategies for all three decoders, *i.e.* $i \in \{\text{SD}, \text{DFD}, \text{DBD}\}$.

	i-Dec1	i-Dec2	i-Dec3	i-Dec4	i-Dec5	i-Pred
Kernel size	5×5	5×5	5×5	5×5	5×5	3×3
# of kernels	512	256	128	64	32	1
BN	✓	✓	✓	✓	✓	
ReLU	✓	✓	✓	✓	✓	

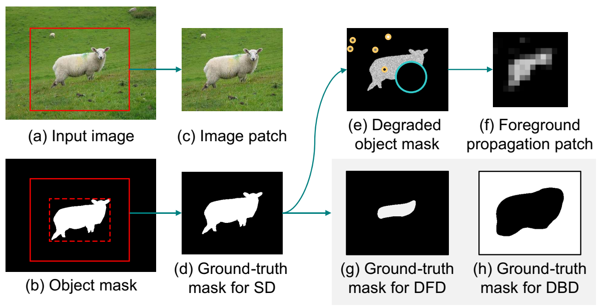
(a) Input image
(c) Image patch
(e) Degraded object mask
(f) Foreground propagation patch
(b) Object mask
(d) Ground-truth mask for SD
(g) Ground-truth mask for DFD
(h) Ground-truth mask for DBD

Figure 5. Preprocessing of training data.

The batch normalization (BN) [21] is applied to all convolution layers in the decoders except for the prediction layers. Also, the rectified linear unit (ReLU) activation function is employed after the batch normalization. We use 5×5 kernels in all convolution layers, except for the prediction layers whose kernel size is 3×3. Table 1 details the decoder parameters. Note that we use the identical kernel sizes and number of kernels for the three decoders.

Training Phase: Annotating objects in all frames in videos is an arduous task. While there are several datasets for video object segmentation [4, 12, 23, 28], each of them consists of a small number of videos from 12 to 59. They are hence insufficient for training the network in Figure 4. Therefore, we instead use the PASCAL VOC 2012 dataset [10], which is released for object classification, detection, and segmentation. Hariharan *et al.* [17] annotate 26,844 object masks on 11,355 images in the PASCAL dataset. Among the object masks, we choose 25,093 object masks to compose a training dataset, by discarding small object masks.

Figure 5 illustrates how to preprocess the training dataset. By cropping a training image and its object mask, we extract an image patch and its ground-truth mask for the SD, respectively. We perform the cropping with margins, proportional to the object size, as shown in Figure 5(b). Note that the foreground propagation patch and the ground-truth masks for the DFD and DBD are not available in the PASCAL dataset. Hence, we generate them through simple image processing. First, we degrade the object mask to yield the foreground propagation patch. We imitate propagation errors, by filling in the masked region with random pixel intensities within [0.5, 1] and then performing partial suppression and noise addition with circular masks. In Figure 5(e), the suppression and noise addition are depicted by blue and yellow circles, respectively. We adjust the radii of the circular masks according to the object mask size, and determine their locations randomly. Then, we resize the degraded object mask to generate the 14×14 foreground propagation patch in Figure 5(f). Next, we synthesize the ground-truth masks for the DFD and DBD. We apply Gaussian smoothing to the object mask and then perform thresholding to extract inner regions of the object mask. The extracted regions are defined as the ground-truth mask for the DFD, as shown in Figure 5(g). The ground-truth mask for the DBD is produced in a similar manner, by employing the inverse of the object mask, as shown in Figure 5(h).

We use the Caffe library [24] to train the proposed network. We compose a minibatch with eight training data. We fix the weight parameters of the encoder as in [7, 57]. We initialize the convolution layers in the decoders with random values. We adopt the cross-entropy losses between ground-truth masks and predicted probability patches. We train the proposed network via the stochastic gradient descent. We set the learning rate to 0.001 for the first 55 epochs and 0.0001 for the next 35 epochs.

Inference Phase: The proposed CTN takes an image patch, a foreground propagation patch, and a background propagation patch as input. First, we extract the image patch and the foreground propagation patch by cropping the current frame $I^{(t)}$ and the propagation map $H^{(t)}$, respectively, around the object pixels in $H^{(t)}$ with margins of 50 pixels. Then, we resize the image patch and the foreground propagation patch to $224 \times 224 \times 3$ and 14×14. We obtain the background propagation patch by inverting the foreground one.

The proposed CTN outputs three probability patches of size 224×224. We restore these patches to the sizes and locations before the cropping, in order to yield the three probability maps: separative probability map R_{S}, definite foreground probability map R_{F}, and definite background probability map R_{B}. Note that these probability maps have the same size as the input frame. We classify each pixel $\mathbf{p}$ in the separative probability map as the foreground, if $R_{\text{S}}(\mathbf{p}) > \theta_{\text{sep}}$. Let $\mathcal{L}$ be the coordinate set for such foreground pixels. In the two probability maps R_{F} and R_{B}, we determine the pixels, whose probabilities are higher than another threshold θ_{def}, as the definite ones. Let $\mathcal{F}$ and $\mathcal{B}$ denote the set of the definite foreground and background pixels, respectively.

3.3. Markov Random Field Optimization

The pooling layers in Figure 4 reduce the number of parameters and the amount of computations. However, they also degrade details of a predicted target object. In other words, the coordinate set $\mathcal{L}$ of foreground pixels may not provide sufficiently detailed segmentation information. Thus, we further improve the segmentation quality by solving a two-class (foreground or background) MRF optimization problem. For notational simplicity, let us omit the superscripts for frame indices. First, we define a graph $G = (N, E)$, whose nodes are pixels in the current frame. N and E denote sets of nodes and edges, respectively. We connect each pixel to its four neighbors by edges. By combining unary and pairwise costs, the MRF energy function $\mathcal{E}(S)$ of the segmentation label map S is defined as

$$\mathcal{E}(S) = \sum_{\mathbf{p}\in N} \mathcal{D}(\mathbf{p}, S) + \gamma \times \sum_{(\mathbf{p},\mathbf{q})\in E} \mathcal{Z}(\mathbf{p}, \mathbf{q}, S) \tag{2}$$

where γ controls the balance between the unary cost $\mathcal{D}$ and the pairwise cost $\mathcal{Z}$.

To compute the unary cost, we build the RGB color GMMs of the foreground and the background, respectively. In this work, we set the same number of Gaussian components, $K = 10$, for both GMMs. We use the pixels in $\mathcal{L}$ to construct the foreground GMMs, based on the expectation-maximization algorithm, and those in $\mathcal{L}^c$ for the background GMMs. Let us define a Gaussian cost as

$$\psi(\mathbf{p}, s) = \min_k \{-\log f(\mathbf{p}\,;\, \mathcal{M}_{s,k})\} \tag{3}$$

where $f(\cdot\,;\, \mathcal{M}_{s,k})$ denotes the probability distribution function of $\mathcal{M}_{s,k}$, which is the kth Gaussian component of the foreground ($s = 1$) GMMs or the background ($s = 0$) GMMs. A high Gaussian cost is returned when the Gaussian distribution function has a low probability. Then, we define the unary cost as

$$\mathcal{D}(\mathbf{p}, S) = \begin{cases} \max_{\mathbf{g}} \psi(\mathbf{g}, 0) & \text{if } \mathbf{p} \in \mathcal{F} \text{ and } S(\mathbf{p}) = 0, \\ \max_{\mathbf{g}} \psi(\mathbf{g}, 1) & \text{if } \mathbf{p} \in \mathcal{B} \text{ and } S(\mathbf{p}) = 1, \\ \psi(\mathbf{p}, S(\mathbf{p})) & \text{otherwise.} \end{cases} \tag{4}$$

Note that $\mathcal{D}(\mathbf{p}, S)$ yields a very high cost, if $\mathbf{p}$ is a foreground definite pixel in $\mathcal{F}$ but is labeled as the background class $S(\mathbf{p}) = 0$. Consequently, the minimization of the unary cost in the MRF energy function in (2) discourages the foreground definite pixels in $\mathcal{F}$ from being labeled as the background. Similarly, it discourages the background definite pixels in $\mathcal{B}$ from being labeled as the foreground.

To encourage neighboring pixels to have the same label, we compute the pairwise cost by

$$\mathcal{Z}(\mathbf{p}, \mathbf{q}, S) = \begin{cases} \exp(-d(\mathbf{p}, \mathbf{q})) & \text{if } S(\mathbf{p}) \neq S(\mathbf{q}), \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where d is the distance between the color and motion features of pixels $\mathbf{p}$ and $\mathbf{q}$. We extract the RGB color features, and use the backward optical flow vectors as the motion features. A high pairwise cost is incurred, if neighboring pixels with similar features are assigned different labels.

For more reliable estimation of foreground and background colors at frame t, we employ the GMMs at the first and $(t-1)$th frames, as well as the GMMs at the current frame t. We use the sum of the three corresponding unary costs in the MRF optimization. We adopt the graph-cut algorithm [3] to minimize the MRF energy function and obtain an optimal segmentation label map S^*. Then, we refine the GMMs at frame t, based on the label map S^*. We iterate these two processes until the convergence.

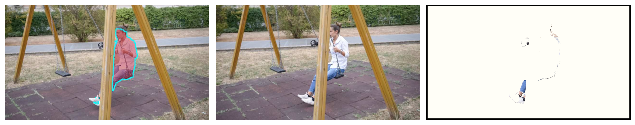
(a) $I^{(t-1)}$ (b) $I^{(t)}$ (c) Reappearing parts

Figure 6. The proposed reappearing object detector discovers reappearing parts of the legs, which are occluded in the previous frame $I^{(t-1)}$. In (a), the foreground boundaries are in a cyan color.

3.4. Reappearing Object Detection

A target object may disappear and be occluded by other objects. If the occluded parts reappear in the current frame, the inter-frame propagation in Section 3.1 may be ineffective in the corresponding regions. Thus, we attempt to identify the reappearing parts. If there is no occlusion and the optical flow estimation is accurate, the backward flow vector should be identical to the inverse of the corresponding forward flow vector. Based on this backward-forward consistency [46], we detect reappearing pixels. Specifically, we first perform the backward matching from pixel $\mathbf{p} = [x, y]^T$ in frame t to pixel $\tilde{\mathbf{p}}$ in frame $t-1$ using the backward optical flow vector at $\mathbf{p}$. Next, we perform the forward matching of $\tilde{\mathbf{p}}$ to $\hat{\mathbf{p}} = [\hat{x}, \hat{y}]^T$ by adopting the forward optical flow vector from frame $t-1$ to frame t. Ideally, the restored pixel $\hat{\mathbf{p}}$ should be equal to the original pixel $\mathbf{p} = [x, y]^T$. Thus, we compute the inconsistency of $\mathbf{p}$ by

$$\phi(\mathbf{p}) = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}/(\mu_{\text{height}} + \mu_{\text{width}}) \tag{6}$$

where μ_{height} and μ_{width} are the height and width of the input video sequence, respectively. If the inconsistency $\phi(\mathbf{p})$ is higher than $1/400$, we declare that $\mathbf{p}$ is inconsistent.

Next, we should detect reappearing foreground pixels from the set of inconsistent pixels. To this end, we use the foreground and background GMMs at the first and $(t-1)$th frames. The reappearing parts are more likely to be represented by the foreground GMMs than by the background ones. Therefore, we determine that an inconsistent pixel belongs to the reappearing parts, if its foreground Gaussian cost is lower than the background Gaussian cost. Note that the Gaussian costs are defined in (3). Figure 6 shows an example of the reappearing object detection result. We include the reappearing pixels into the set $\mathcal{L}$ of foreground pixels before the MRF optimization.

Table 2. Performance comparison of the video object segmentation algorithms on the DAVIS dataset [36]. The best and the second best results are boldfaced and underlined, respectively.

	Region similarity (RS)			Contour accuracy (CA)		
Algorithm	Mean ↑	Recall ↑	Decay ↓	Mean ↑	Recall ↑	Decay ↓
A. Unsupervised algorithms						
NLC [11]	0.641	0.731	0.086	0.593	0.658	0.086
CVOS [48]	0.514	0.581	0.127	0.490	0.578	0.138
TRC [14]	0.501	0.560	0.050	0.478	0.519	0.066
MSG [33]	0.543	0.636	0.028	0.525	0.613	0.057
KEY [27]	0.569	0.671	0.075	0.503	0.534	0.079
SAL [53]	0.426	0.386	0.084	0.383	0.264	0.072
FST [35]	0.575	0.652	0.044	0.536	0.579	0.065
ACO [23]	0.531	0.611	0.093	0.504	0.558	0.088
B. Semi-supervised algorithms						
TSP [5]	0.358	0.388	0.385	0.346	0.329	0.388
SEA [40]	0.556	0.606	0.355	0.533	0.559	0.339
JMP [12]	0.607	0.693	0.372	0.586	0.656	0.373
FCP [37]	0.631	0.778	**0.031**	0.546	0.604	**0.039**
BVS [31]	0.665	0.764	0.260	0.656	0.774	0.236
Prop-Q	**0.755**	**0.890**	0.144	**0.714**	**0.848**	0.140
Prop-F	$\underline{0.734}$	$\underline{0.865}$	$\underline{0.123}$	$\underline{0.680}$	$\underline{0.799}$	$\underline{0.123}$

Table 3. Jaccard indices on the SegTrack dataset [49]. Higher values are better. The best and the second best results are boldfaced and underlined, respectively.

Sequence	[15]	[52]	[40]	[54]	[50]	[31]	Prop-Q
Girl	0.54	0.52	0.62	0.84	$\underline{0.88}$	**0.89**	0.86
Birdfall	0.56	0.33	0.09	**0.78**	0.57	$\underline{0.66}$	0.61
Parachute	0.86	0.70	0.93	$\underline{0.94}$	**0.95**	$\underline{0.94}$	$\underline{0.94}$
Cheetah	$\underline{0.46}$	0.33	0.18	**0.63**	0.34	0.10	0.40
Monkeydog	$\underline{0.61}$	0.22	0.05	**0.82**	0.54	0.41	0.57
Average	0.60	0.42	0.37	**0.80**	0.66	0.60	$\underline{0.68}$

4. Experimental Results

We evaluate the proposed algorithm on the state-of-the-art DAVIS benchmark dataset [36], composed of 30 training videos, 20 validation videos, and the corresponding ground-truth label maps. We use all 50 videos for the evaluation, since they are not used for training the proposed algorithm. The spatial resolutions of these videos are 854×480, and the number of frames in each video is from 25 to 104. The videos are very challenging due to fast motion, occlusion, and object deformation.

We use the performance measures introduced in [36]. To quantify the *region similarity* (RS), we use the Jaccard index, which is the intersection-over-union ratio of a predicted segmentation label map and the ground-truth mask. Also, the *contour accuracy* (CA) is reported in terms of an F-measure, which is the combination of the precision and recall rates of contour pixels. For these metrics, we report the three statistics: mean, recall, and decay. The mean averages the scores over all frames. The recall computes the proportion of frames, the segmentation scores of which are higher than 0.5. The decay first divides all frames into four clips, and then computes the score difference between the last quarter and the first quarter.

Table 4. Segmentation scores of the proposed algorithm in various settings.

	Region similarity (RS)			Contour accuracy (CA)		
Setting	Mean ↑	Recall ↑	Decay ↓	Mean ↑	Recall ↑	Decay ↓
A. Ablation studies						
w/o DDs	0.684	0.756	0.185	0.677	0.790	0.159
w/o SD	0.663	0.797	0.248	0.665	0.792	0.216
B. Efficacy of MRF optimization						
Before MRF	0.715	0.857	0.131	0.663	0.791	0.142
After MRF	0.755	0.890	0.144	0.714	0.848	0.140

Table 2 compares the proposed algorithm with 13 conventional algorithms: unsupervised ones [11, 14, 23, 27, 33, 35, 48, 53] and semi-supervised ones [5, 12, 31, 37, 40]. Note that [12] is a supervised algorithm, but it operates as a semi-supervised one when user annotations are given at the first frame only. The scores of the conventional algorithms are from [31, 36] except for ACO [23]. For ACO, we use the source codes, which are available online. We report the scores of two versions of the proposed algorithm: 'Prop-Q' uses the state-of-the-art optical flow technique [19], whereas 'Prop-F' adopts a much faster optical flow technique [26]. Both Prop-Q and Prop-F surpass all conventional algorithms. Especially, in terms of the RS recall, Prop-Q outperforms the second best algorithm FCP [37] by a considerable margin, about 14.4%. Also, Prop-Q is superior to BVS [31] in terms of the RS mean by about 13.5%. FCP yields better decay scores than the proposed algorithm. This is because, while the proposed algorithm performs the segmentation sequentially from the first to last frames, FCP discovers a target object by considering object proposals in all frames simultaneously.

Figure 7 shows that the proposed algorithm yields spatially accurate and temporally coherent segment tracks, even when the target objects undergo fast motion ("Dog-agility"), occlusion ("Dog-agility" and "Motorbike"), and deformation ("Dance-twirl" and "Dog-agility").

For the sake of complete evaluation, we test the semi-supervised algorithms on SegTrack dataset [49], which is widely used to assess video object segmentation techniques. Table 3 lists the Jaccard indices of the segmentation results. In average, the proposed algorithm outperforms the conventional algorithms except for [54]. However, [54] uses manually tuned parameters for each sequence, while the others use fixed parameters.

Ablation Study: We perform two ablation studies. Prop-Q is used in these studies. First, we remove the DFD and DBD, and thus $\mathcal{F} = \mathcal{B} = \varnothing$ in (4). Second, we use the propagation map $H^{(t)}$ to select initial foreground pixels in the MRF optimization, instead of employing the set $\mathcal{L}$ of foreground pixels that are detected by the SD. Let us refer to the first and the second settings as 'w/o DDs' and 'w/o

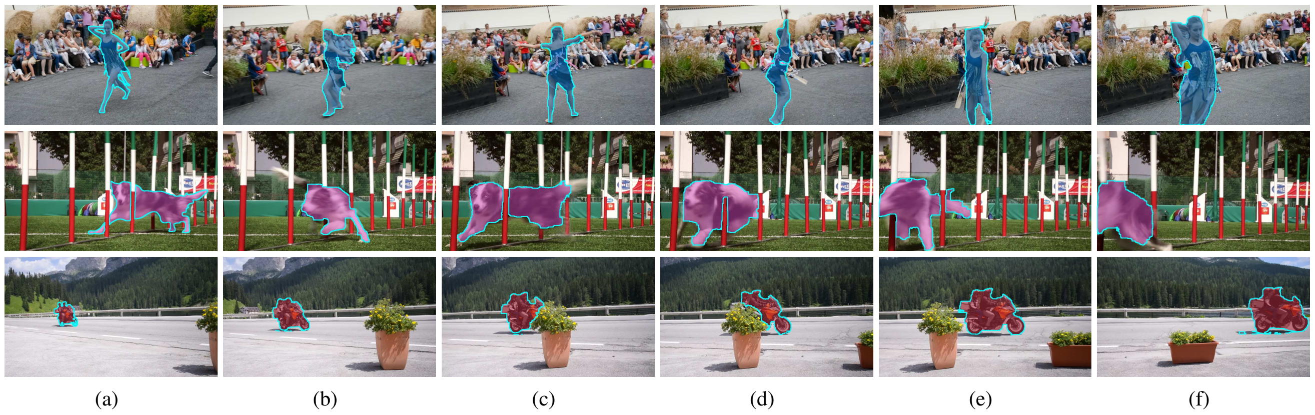

Figure 7. Segmentation results of the proposed algorithm. (a) shows the user-annotated target objects at the first frame. (b)$\sim$(f) are the segmentation results at subsequent frames. From top to bottom, the frames are from “Dance-twirl,” “Dog-agility,” and “Motorbike” in the DAVIS dataset [36].

Table 5. Comparison of computational times. The fastest and the second fast algorithms are boldfaced and underlined, respectively.

	NLC [11]	SEA [40]	JMP [12]	BVS [31]	Prop-Q	Prop-F
Time (SPF)	45.62	13.69	27.37	**0.84**	29.95	<u>1.33</u>

SD,’ respectively. Table 4 lists the RS and CA scores for each ablation setting. In both settings, the performance is degraded severely, which indicates that all three decoders are necessary for accurate video object segmentation.

Efficacy of MRF Optimization: Table 4 also measures the qualities of segmentation maps of the proposed algorithm before and after the MRF optimization. It is observable that the MRF optimization further refines segmentation maps. Notice that the reappearing object detection technique is adopted after the CTN to augment the set of initial foreground pixels in the MRF optimization. Since many objects in the DAVIS dataset [36] suffer from occlusion and reappearance, the MRF optimization improves the performance considerably.

Running Time Analysis: We measure the running times of the segmentation algorithms in seconds per frame (SPF). We test the proposed algorithm on the “Blackswan” sequence in the DAVIS dataset [36] using a PC with a Titan X GPU and a 3.0 GHz CPU. Table 5 shows that the faster version of the proposed algorithm, Prop-F, is faster than most algorithms, while providing superior performance.

Parameter Selection: For balancing the unary and pairwise costs, we set γ to 25 in (2). The proposed algorithm has two controllable parameters θ_{sep} and θ_{def}. θ_{sep} thresholds the separative map of the SD, while θ_{def} binarizes the two definite maps of the DFD and DBD. Figure 8 shows the RS and CA scores for various combinations of θ_{sep} and θ_{def}. It is observable that the proposed algorithm provides the best results at $\theta_{\text{sep}} = 0.3$ and $\theta_{\text{def}} = 0.9$. Thus, these parameters are fixedly used in all experiments.

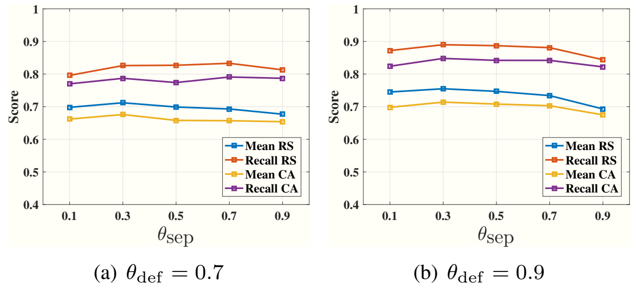

Figure 8. Segmentation qualities according to the two parameters, θ_{sep} and θ_{def}.

5. Conclusions

We proposed a semi-supervised online video object segmentation algorithm. First, a segmentation label map is propagated from the previous frame to the current frame. Then, the CTN yields three probability maps, tailored for the binary labeling problem. To delineate a target object, we performed the MRF optimization by adopting the tailored probability maps. Experimental results demonstrated that the proposed algorithm significantly outperforms the state-of-the-art conventional algorithms [12, 31, 37, 40] on the DAVIS benchmark dataset [36].

Acknowledgements

This work was supported partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2015R1A2A1A10055037), and partly by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00464) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: Robust video object cutout using localized classifiers. *ACM Trans. Graphics*, 28(3):70, 2009. 1

[2] P. Bideau and E. Learned-Miller. Its moving! A probabilistic model for causal motion segmentation in moving camera videos. In *ECCV*, pages 433–449, 2016. 2

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. 6

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, pages 282–295. 2010. 2, 5

[5] J. Chang, D. Wei, and J. W. Fisher. A video representation using temporal superpixels. In *CVPR*, pages 2051–2058, 2013. 2, 7

[6] P. Chockalingam, N. Pradeep, and S. Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *ICCV*, pages 1530–1537, 2009. 2

[7] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, pages 3150–3158, 2016. 1, 3, 4, 5

[8] S. Duffner and C. Garcia. PixelTrack: A fast adaptive algorithm for tracking non-rigid objects. In *ICCV*, pages 2480–2487, 2013. 2

[9] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, pages 575–588. 2010. 2

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 5

[11] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, pages 1–12, 2014. 1, 2, 7, 8

[12] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. JumpCut: Non-successive mask transfer and interpolation for video cutout. *ACM Trans. Graphics*, 34(6):195, 2015. 1, 5, 7, 8

[13] A. Fathi, M. F. Balcan, X. Ren, and J. M. Rehg. Combining self training and active learning for video segmentation. In *BMVC*, pages 1–11, 2011. 1

[14] K. Fragkiadaki, G. Zhang, and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *CVPR*, pages 1846–1853, 2012. 2, 7

[15] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *ICCV*, pages 81–88, 2011. 1, 2, 7

[16] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, pages 3129–3136, 2010. 3

[17] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998, 2011. 5

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[19] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine PatchMatch for large displacement optical flow. In *CVPR*, pages 5704–5712, 2016. 3, 7

[20] F. J. Huang, Y.-L. Boureau, Y. LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, pages 1–8, 2007. 1, 2

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 5

[22] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *ECCV*, pages 656–671. 2014. 2

[23] W.-D. Jang, C. Lee, and C.-S. Kim. Primary object segmentation in videos via alternate convex optimization of foreground and background distributions. In *CVPR*, pages 696–704, 2016. 2, 5, 7

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM Multimedia*, pages 675–678, 2014. 5

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[26] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *ECCV*, pages 471–488, 2016. 3, 7

[27] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, pages 1995–2002, 2011. 1, 2, 7

[28] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, pages 2192–2199, 2013. 1, 2, 5

[29] N. Liu and J. Han. DHSNet: Deep hierarchical saliency network for salient object detection. In *CVPR*, pages 678–686, 2016. 1, 3, 4

[30] T. Ma and L. J. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, pages 670–677, 2012. 2

[31] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung. Bilateral space video segmentation. In *CVPR*, pages 743–751, 2016. 1, 2, 7, 8

[32] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, pages 1520–1528, 2015. 1, 3

[33] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, pages 1583–1590, 2011. 1, 2, 7

[34] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, pages 614–621, 2012. 2

[35] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, pages 1777–1784, 2013. 1, 2, 7

[36] F. Perazzi, J. P.-T. B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, pages 724–732, 2016. 1, 7, 8

[37] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, pages 3227–3234, 2015. 1, 2, 7, 8

[38] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, pages 1990–1998, 2015. 1, 3

[39] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, pages 75–91, 2016. 4

[40] S. A. Ramakanth and R. V. Babu. SeamSeg: Video object segmentation using patch seams. In *CVPR*, pages 376–383, 2014. 1, 2, 7, 8

[41] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1, 4

[42] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Trans. Graphics*, volume 23, pages 309–314, 2004. 3

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. 1

[44] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, pages 1154–1160, 1998. 2

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 3

[46] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, pages 438–451, 2010. 6

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1

[48] B. Taylor, V. Karasev, and S. Soattoc. Causal video object segmentation from persistence of occlusions. In *CVPR*, pages 4268–4276, 2015. 7

[49] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label MRF optimization. In *BMVC*, pages 1–11, 2010. 2, 7

[50] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, pages 3899–3908, 2016. 2, 7

[51] D. Varas and F. Marques. Region-based particle filter for video object segmentation. In *CVPR*, pages 3470–3477, 2014. 2

[52] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011. 2, 7

[53] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, pages 3395–3402, 2015. 1, 2, 7

[54] L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. JOTS: Joint online tracking and segmentation. In *CVPR*, pages 2226–2234, 2015. 1, 2, 7

[55] F. Xiao and Y. J. Lee. Track and segment: An iterative unsupervised approach for video object proposals. In *CVPR*, pages 933–942, 2016. 2

[56] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang. Deep interactive object selection. In *CVPR*, pages 373–381, 2016. 3

[57] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, pages 193–202, 2016. 1, 3, 5

[58] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014. 4

[59] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, pages 628–635, 2013. 2