Video Salient Object Detection Using Spatiotemporal Deep Features

Trung-Nghia Le and Akihiro Sugimoto

***Abstract*— This paper presents a method for detecting salient objects in videos, where temporal information in addition to spatial information is fully taken into account. Following recent reports on the advantage of deep features over conventional handcrafted features, we propose a new set of spatiotemporal deep (STD) features that utilize local and global contexts over frames. We also propose new spatiotemporal conditional random field (STCRF) to compute saliency from STD features. STCRF is our extension of CRF to the temporal domain and describes the relationships among neighboring regions both in a frame and over frames. STCRF leads to temporally consistent saliency maps over frames, contributing to accurate detection of salient objects' boundaries and noise reduction during detection. Our proposed method first segments an input video into multiple scales and then computes a saliency map at each scale level using STD features with STCRF. The final saliency map is computed by fusing saliency maps at different scale levels. Our experiments, using publicly available benchmark datasets, confirm that the proposed method significantly outperforms the state-of-the-art methods. We also applied our saliency computation to the video object segmentation task, showing that our method outperforms existing video object segmentation methods.**

***Index Terms*— Video saliency, salient object detection, spatiotemporal deep feature, spatiotemporal CRF, video object segmentation.**

I. Introduction

SALIENT object detection from videos plays an important role as a pre-processing step in many computer vision applications such as video re-targeting [1], object detection [2], person re-identification [3], and visual tracking [4]. Conventional methods for salient object detection often segment each frame into regions and artificially combine low-level (bottom-up) features (e.g., intensity [5], color [5], edge orientation [6]) with heuristic (top-down) priors (e.g., center prior [7], boundary prior [5], objectness [6]) detected from the regions.

Manuscript received August 4, 2017; revised January 15, 2018, March 7, 2018, May 5, 2018, and June 5, 2018; accepted June 8, 2018. Date of publication June 22, 2018; date of current version July 9, 2018. This work was supported in part by JST CREST under Grant JPMJCR14D1 and in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant 16H02851. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Khan M. Iftekharuddin. *(Corresponding author: Trung-Nghia Le.)*

T.-N. Le is with the Department of Informatics, SOKENDAI (The Graduate University for Advanced Studies), Tokyo 101-8430, Japan (e-mail: ltnghia@nii.ac.jp).

A. Sugimoto is with the National Institute of Informatics, Tokyo 101-8430, Japan (e-mail: sugimoto@nii.ac.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2849860

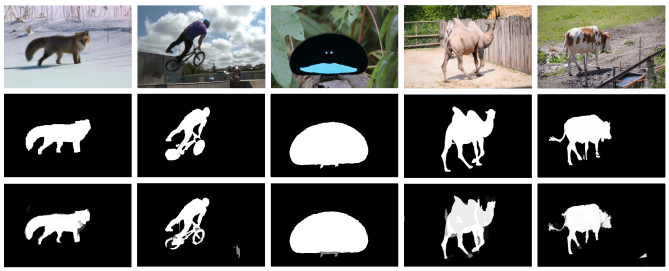


Fig. 1. Examples of results obtained by our proposed method. Top row images are original video frames, followed by the ground truth and corresponding saliency maps obtained using our method.

Low-level features and priors used in the conventional methods are hand-crafted and are not sufficiently robust for challenging cases, especially when the salient object is presented in a low-contrast and cluttered background. Although machine learning based methods have been recently developed [8]–[10], they are primary for integrating different hand-crafted features [9], [11] or fusing multiple saliency maps generated from various methods [8]. Accordingly, they usually fail to preserve object details when the salient object intersects with the image boundary or has similar appearance with the background where hand-crafted features are often unstable.

Recent advances in deep learning using Deep Neural Network (DNN) enable us to extract visual features, called deep features, directly from raw images/videos. They are more powerful for discrimination and, furthermore, more robust than hand-crafted features [12]–[14]. Indeed, saliency models for videos using deep features [15]–[17] have demonstrated superior results over existing works utilizing only hand-crafted features. However, they extract deep features from each frame independently and employ frame-by-frame processing to compute saliency maps, leading to inaccuracy for dynamically moving objects. This is because temporal information over frames is not taken into account in computing either deep features or saliency maps. Incorporating temporal information in such computations should lead to better performance.

Computed saliency maps do not always accurately reflect the shapes of salient objects in videos. To segment salient objects as accurately as possible while reducing noise, dense Conditional Random Field (CRF) [15], [18], a powerful graphical model to globally capture the contextual information, has been applied to the computed saliency maps, which results in improvement in spatial coherence and contour localization. However, dense CRF is applied to each frame of a video separately, meaning that only spatial contextual information is

considered. Again, temporal information over frames should be taken into account for better performance.

Motivated by the above observation, we propose a novel framework using spatiotemporal information as fully as possible for salient object detection in videos. We introduce a new set of SpatioTemporal Deep (STD) features that utilize both local and global contexts over frames. Our STD features consist of local and global features. The local feature is computed by aggregating over frames deep features, which are extracted from each frame using a region-based Convolutional Neural Network (CNN) [13]. The global feature is computed from a temporal-segment of a video using a block-based[1] CNN [14]. We also introduce the SpatioTemporal CRF (STCRF), in which the spatial relationship between regions in a frame as well as temporal consistency of regions over frames is formally described using STD features. Our proposed method first segments an input video into multi-scale levels, and then at each scale level, extracts STD features and computes a saliency map. The method then fuses saliency maps at different scale levels into the final saliency map. Extensive experiments on public benchmark datasets for video saliency confirm that our proposed method significantly outperforms the state-of-the-arts. Examples of saliency maps obtained by our method are shown in Fig.1. We also apply our method to video object segmentation and observe that our method outperforms existing methods.

The rest of this paper is organized as follows. We briefly review and analyze related work in Section II. Then, we present in detail our proposed method in Section III. Our experiments are discussed in Sections IV and V. In Section VI, we present an application of our proposed method to video object segmentation. Section VII presents conclusion and future work. We remark that this paper extends the work reported in [19]. Our extensions in this paper are building a new STCRF model utilizing CNN instead of Random Forest (Section III-C.1), adding more experiments (Section V), and an application of our method to video object segmentation (Section VI).

II. Related Work

Here we briefly survey features used for salient object detection in videos, and saliency computation methods.

A. Features for Salient Object Detection

Saliency computation methods for videos using hand-crafted features are mostly developed from traditional saliency models for still images by incorporating motion features to deal with moving objects [6], [7], [10], [20]. Motion features commonly used include optical flow [6], [7], [20], trajectories of local features [10], [21], gradient flow field [22], and temporal motion boundary [23]; they are utilized to detect salient objects in videos. Xue *et al.* [24], on the other hand, sliced a video along X–T and Y–T planes to separate foreground moving objects from backgrounds. However, hand-crafted features have limitation in capturing the semantic concept of objects. Accordingly, these methods often fail when the salient object crosses the image boundary or has similar appearance with the background.

Several existing methods [15], [25] for saliency computation using deep features, on the other hand, utilize superpixel segmentation to extract region-level deep features in different ways (e.g., feeding regions into a CNN individually to compute deep features [25] or pooling a pixel-level feature map into regions to obtain region-level deep features [15]). To exploit the context of a region in multiple scales, multi-scale deep features of the region are extracted by changing the window size [25]. Li and Yu [25] fused multi-scale deep features of a region of interest to compute the saliency score for the region using a two-layer DNN. Lee *et al.* [17] integrated hand-crafted features into deep features to improve accuracy for salient object detection. More precisely, they concatenated an encoded low-level distance map and a high-level feature map from CNN to enrich information included in the extracted feature map. The region-level feature map and the pixel-level feature map are also integrated into the saliency model to enhance accuracy of detected object boundaries [15]. In end-to-end deep saliency models [16], [26], pixel-based deep features are enhanced by their context information through recurrent CNNs.

Saliency models using deep features have demonstrated state-of-the-art performance in salient object detection and significantly outperformed existing works utilizing only hand-crafted features. However, in almost all existing saliency models, temporal information over frames is not taken into account in deep features, leading to inaccuracy for dynamically moving objects. Though Wang *et al.* [27] very recently proposed a fully convolutional network (FCN) having a pair of frames as its input for video saliency computation, a pair of frames is too short to exploit the temporal domain. Therefore, effectively mining correlation inherent in the spatial and temporal domains into powerful deep features for saliency computation is still an open problem.

B. Saliency Computation Methods

The salient object detection approach using deep models [15], [16], [18], [26], [28] computes saliency scores directly from FCNs. In these deep models, recurrent layers [16], [26] and skip connections [16], [18] are utilized to enhance the contextual information of deep feature maps to improve the accuracy of saliency computation. However, these methods focus on frame-by-frame processing without considering any temporal information in videos. In addition, they still do not detect boundaries of salient objects accurately. A refinement post-processing step is usually required to improve accuracy of detected object boundaries.

Spatial CRF has the capability to relate local regions in order to capture global context, and has been commonly used for refinement in semantic segmentation [29] and for saliency computation [15], [18]. Dense CRF [30] is used as a post-processing step to refine the label map generated from CNN to improve the performance of semantic segmentation [29]. Shimoda and Yanai [29] developed a weakly supervised semantic segmentation method using a dense CRF to refine

[1]In contrast to the region-based CNN working on spatial segments in each frame, the block-based CNN works on a sequence of frames of a video.

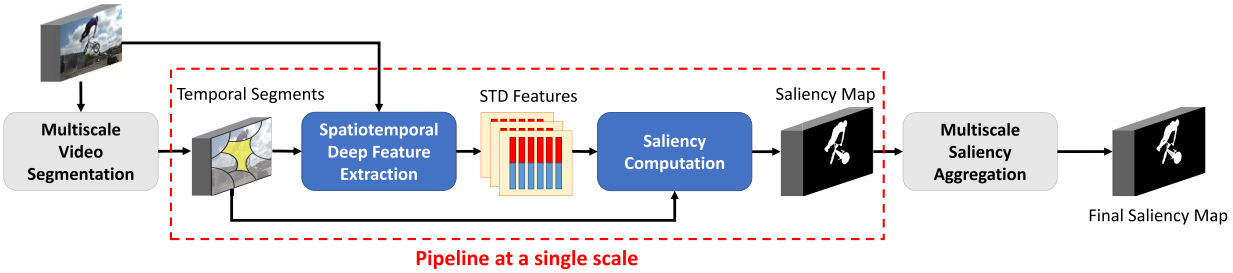


Fig. 2. Pipeline of the proposed method (brighter means more salient in the final saliency map).

results from distinct class saliency maps. The dense CRF is incorporated into the saliency map computed from the CNN to improve spatial coherence and contour localization [15], [18]. Though spatial information is successfully utilized using CRFs in these methods, how to deal with temporal information is left unanswered, which is crucial for videos.

Dynamic CRF (DCRF) [31] is an extension of the spatial CRF toward to the spatiotemporal domain to exploit both spatial and temporal information in videos. DCRF is constructed from consecutive video frames, where each pixel connects to its neighboring pixels in both space (i.e., the same frame) and time (i.e., the next frame and the previous frame). DCRF has been used to enhance both spatial accuracy and temporal coherence for object segmentation [31]–[33] and saliency computation [10] in videos. Yi *et al.* [33] proposed a framework using DCRF to improve fence segmentation in videos. Wang and Ji [31] and Wang *et al.* [32] applied DCRF to object segmentation and moving shadow segmentation in indoor scenes in videos. SIFT flow features were incorporated into DCRF to detect salient objects from videos [10]. However, DCRF is a pixel-level dense graph; thus it is usually constructed using only two successive frames due to large memory consumption. In addition, since the energy function of DCRF is defined using the combination of classical hand-crafted features such as color and optical flow, DCRF is not capable of exploiting spatial and temporal information semantically. Our proposed STCRF differs from DCRF in that STCRF is defined over regions using STD features only, so that it is capable of dealing with more successive frames and exploiting spatial and temporal information semantically with less computational cost.

Different from these existing methods, our proposed method utilizes spatiotemporal information as much as possible when both extracting deep features and computing saliency maps. More precisely, our method uses STD features computed from the spatiotemporal domain together with STCRF constructed in the spatiotemporal domain to produce accurate saliency maps. Our method thus accurately detects boundaries of salient objects by removing irrelevant small regions.

III. Proposed Method

A. Overview

Our goal is to compute a saliency map to accurately segment salient objects in every frame from an input video while fully utilizing information along the temporal dimension. Figure 2 illustrates the pipeline of our proposed method.

We segment an input video at multiple scale levels and compute a saliency map at each scale level at each frame, and then aggregate all saliency maps at different scale levels at each frame into a final saliency map. This follows our intuition that objects in a video contain various salient scale patterns and an object at a coarser scale level may be composed of multiple parts at a finer scale level.

In this work, we employ the video segmentation method [34] at multiple scale levels. We first specify the number of initial superpixels to define a scale level. For each scale level, we then segment each frame into initial superpixels using entropy-rate superpixel segmentation [34]. Similar superpixels at consecutive frames are then grouped and connected across frames to have temporal segments using parametric graph partitioning [35]. By specifying different numbers of initial superpixels, we obtain multiple scale temporal segments (we set four numbers to have four scale levels in our experiments as discussed later). We remark that each scale level has a different number of segments, which are defined as (non-overlapping) regions.

The final saliency map is computed by taking the average value of saliency maps over different scale levels. In the following subsections, we explain how to compute a saliency map at a scale level. We remark that a saliency map in this section indicates the saliency map at a scale level unless explicitly stated with "final."

B. Spatiotemporal Deep Feature Extraction

For each region (segment) at each frame, our proposed STD feature is computed by concatenating a local feature and a global feature. The local feature is extracted using a region-based CNN followed by aggregation over frames, while the global feature is computed using a block-based CNN whose input is a sequence of frames of the video. The STD feature extraction for a region is illustrated in Fig. 3.

1) Local Feature Extraction: A region at each frame, which is defined from a temporal segment at a frame, is fed into a region-based CNN to extract its region-based feature which is with a dimension of 4096. As our region-based CNN, we use the publicly available R-CNN model[2] [13] that was pre-trained on the ImageNet ILSVRC-2013 challenge dataset [39].

The region-based feature contains the local context of the region but does not contain temporal information because it

[2]R-CNN runs at the original resolution of its input region while Fast R-CNN [36], Faster R-CNN [37], and Mask R-CNN [38] require to reduce the resolution of the region to adapt their architectures. This resolution reduction may eliminate small regions. We thus used R-CNN.

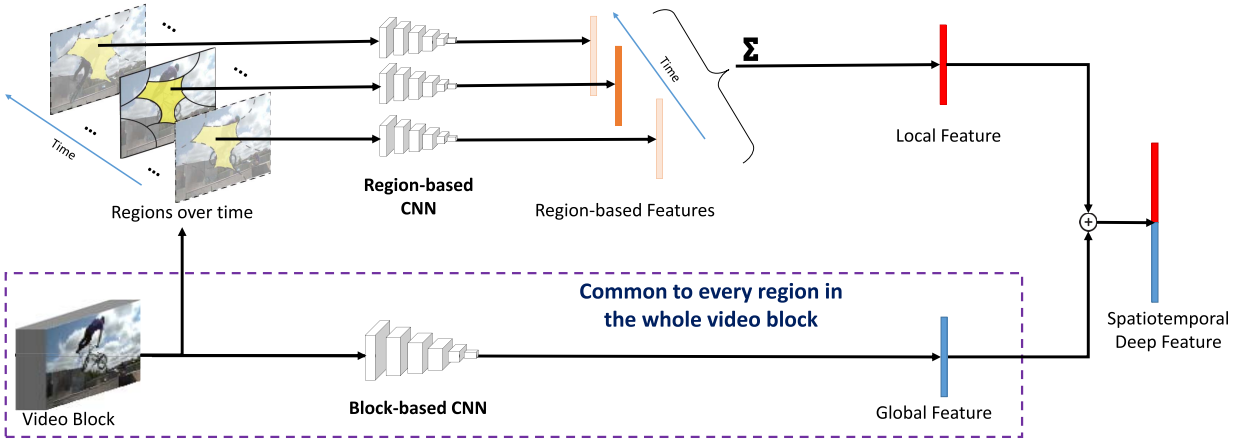


Fig. 3. Spatiotemporal deep (STD) feature extraction for a region. A region (yellow) in a frame of a video block is fed to the region-based CNN to have the region-based feature of the (yellow) region in the frame. Region-based features over the frames of the video block are aggregated to have the local feature of the region. On the other hand, the video block is fed to the block-based CNN to have the global feature. The local feature of the region and the global feature are concatenated to form the spatiotemporal deep (STD) feature of the region.

is computed frame-by-frame. In order to incorporate temporal information, for a region, we aggregate its region-based features over a sequence of frames, resulting in the consistent local feature over time for the region. It is important to remark that we use only neighboring frames whenever a region-of-interest is present. Thus, the number of frames used for this aggregation may change depending on the region.

Just uniformly averaging region-based features over frames is not wise because pixels vary over time due to lossy compression, degrading accuracy of corresponding regions across frames. This degradation increases with larger time increments across frames. We thus linearly combine region-based features at neighboring frames, similarly to [6], using weights modeled by a Gaussian distribution centered at the frame from which we compute local features. With these weights, region-based features at frames with large temporal distance to a frame of interest will contribute less to the computation of local features of the frame: the local feature $F_{\mathrm{L}}(i,t)$ of a region i at frame t is extracted by

$$F_{\mathrm{L}}(i,t) = \frac{1}{\Psi} \sum_{t'=t-k/2}^{t+k/2} \mathcal{G}\left(t'|t,\sigma^2\right) f(i,t'), \qquad (1)$$

where $\mathcal{G}\left(t'|t,\sigma^2\right)$ is a Gaussian distribution with mean t and standard deviation $\sigma = 2$ expressing distribution of temporal weights, $f(i,t')$ is the region-based feature of region i at frame t', and $\Psi = \sum_{t'=t-k/2}^{t+k/2} \mathcal{G}\left(t'|t,\sigma^2\right)$ (normalizing factor). $k+1$ is the number of frames where the region i is always present.

In this work, we set $k = 16$ by default. This is because almost all regions at a frame are present in the next (previous) 8 successive frames. For a region that disappears during the next 7 successive frames or that newly appears during the previous 7 successive frames, we first identify the maximum number of successive frames in which the region is always present in the previous and the next directions and use this number as k for the region. For example, if a region in a frame appears from 3 frames before and disappears in 2 frames after, then we set $k = 4 (= 2 \times 2)$ for this region.

2) Global Feature Extraction: To compute a global feature, we feed a video block (sequence of frames) of a video into a block-based CNN. The global feature obtained in this way takes its temporal consistency into account in its nature. As our block-based CNN, we employ the C3D model [14] pre-trained on the Sports-1M dataset [40], which is known to be effective for extracting spatiotemporal features for action recognition. As an input video block, frame t is expanded into both directions in the temporal domain to obtain a 16-frame sequence as suggested by Tran *et al.* [14]. For each input block, we feed it into the pre-trained C3D model only once and assign the extracted global feature $F_{\mathrm{G}}(t)$ with a dimension of 4096 identical to all the regions in the block. This distributes the global context to each region and, at the same time, reduces the computational cost.

Finally, for a region i of a frame t, we concatenate its local and global feature vectors to obtain its STD feature vector $F(i,t)$ whose dimension is 4096×2: $F(i,t) = F_{\mathrm{L}}(i,t) \oplus F_{\mathrm{G}}(t)$ (cf. Fig. 3).

C. Saliency Computation Using Spatiotemporal CRF

CRF is used to improve accuracy (particularly in object boundaries) of the saliency map while reducing noise because CRF captures the spatial relationship between regions in a frame. We extend CRF toward the temporal domain to have the ability to capture temporal consistency of regions over frames as well. We call our extended CRF, SpatioTemporal CRF (STCRF in short).

1) STCRF Graph Construction: For temporal segments of a video block, we construct a STCRF graph. Each vertex of the graph represents a region, which is defined from a temporal segment at a frame, in the block. Each edge of the graph, on the other hand, represents the neighboring relationship between regions in space or in time. Considering all the neighboring relationships, however, leads to a dense graph especially when the video volume is large, and the constructed graph becomes practically useless in considering memory consumption and processing time in the inference process.

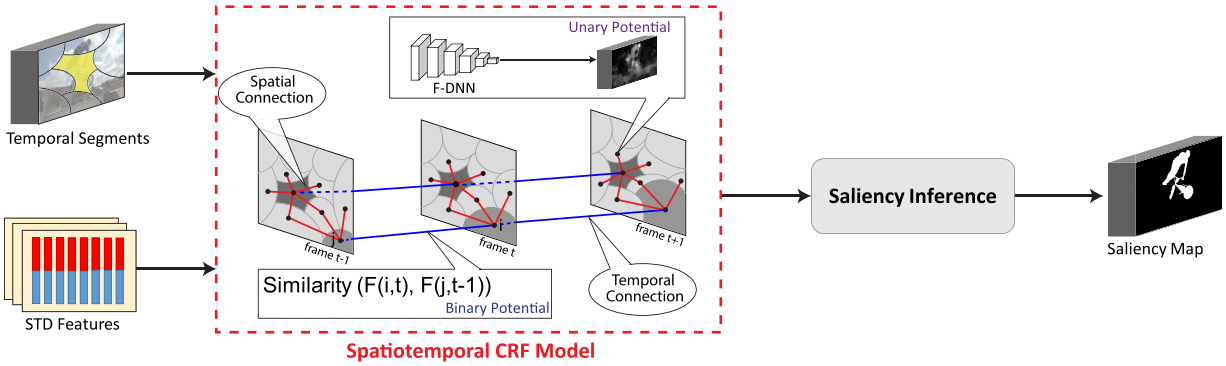


Fig. 4. Saliency computation pipeline for a video block based on a graphical model.

We therefore employ edges that only represent adjacency relationship (cf. Fig. 4). Furthermore, we partition the video into a sequence of consecutive blocks so that inference in each block is performed separately.

In the experiments, an input video is decomposed into overlapping blocks with a fixed size where the overlapping rate is 50%. We note that each block length is equal to 16 frames (see Section V-C.4). The saliency score of a region is refined by uniformly averaging saliency scores of the region over all the blocks that contain the region. This reduces processing time while keeping accuracy.

2) Energy Function for STCRF: We define the energy function of the STCRF so that probabilistic inference is realized by minimizing the function. The energy function E has a video block (with its temporal segments) $\boldsymbol{x}$ as its input. E is defined by the unary and the binary terms with labels representing foreground/background $\boldsymbol{l} = \{l_i \in \{0, 1\} | i \in \mathcal{V}\}$ where l_i is the label for region i, and $\mathcal{V}$ is the set of vertices, i.e., regions in $\boldsymbol{x}$:

$$E(\boldsymbol{l}, \boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i \in \mathcal{V}} \psi_{\mathrm{u}}(l_i; \theta_{\mathrm{u}}) + \sum_{(i,j) \in \mathcal{E}} \psi_{\mathrm{b}}(l_i, l_j; \theta_{\mathrm{b}}), \quad (2)$$

where ψ_{u} and ψ_{b} are the unary and binary potentials given below. $\mathcal{E}$ is the set of edges of the STCRF graph. $\boldsymbol{\theta} = (\theta_{\mathrm{u}}, \theta_{\mathrm{b}})$ is the model parameter.

a) Unary potential: The unary potential for region i is defined using the label of the region:

$$\psi_{\mathrm{u}}(l_i; \theta_{\mathrm{u}}) = \theta_{\mathrm{u}}\, \omega(F(i, t_i)), \quad (3)$$

where t_i is the frame in which region i exists, and ω is a function estimating the probability of the region being the foreground.

To compute ω, i.e., the probability of the region being the foreground, we employ the DNN proposed by Wang *et al.* [41] and modify it for our problem (cf. Table I). Namely, right before the last fully connected layer of the original network, we add a dropout layer and a fully connected layer followed by a Rectified Linear Unite (ReLU) [42] layer (Nos. 15, 16, and 17 in Table I) to increase the depth of the network. We then appropriately change the output channel of the first fully connected layer (Nos. 1, 2, and 3).

Hence, our used network, called Foreground-Deep Neural Network (F-DNN in short), consists of 7 fully connected layers. Each layer executes a linear transformation followed by the ReLU operator. Dropout operations with the ratio of 0.5 are applied after ReLU layers during the training process to avoid overfitting. To the input STD feature having 8192 channels, the numbers of output channels gradually reduce to 2048 at the first three fully connected layers and to 1024 at the next three layers. The last fully connected layer has two output channels representing foreground and background classes.

TABLE I
ARCHITECTURE OF THE OUR F-DNN

No	Layer	Output Channel
0	STD Feature Input	8192
1	Fully Connected	2048
2	ReLU	2048
3	Dropout	2048
4	Fully Connected	2048
5	ReLU	2048
6	Dropout	2048
7	Fully Connected	2048
8	ReLU	2048
9	Dropout	2048
10	Fully Connected	1024
11	ReLU	1024
12	Dropout	1024
13	Fully Connected	1024
14	ReLU	1024
15	Dropout	1024
16	Fully Connected	1024
17	ReLU	1024
18	Fully Connected	2

b) Binary potential: The binary potential provides the deep feature based smoothing term that assigns similar labels to regions with similar deep features. Depending on spatial adjacency or temporal adjacency, the potential is differently formulated with further separation of θ_{b} into θ_{bs} and θ_{bt}:

$$\psi_{\mathrm{b}}(l_i, l_j; \theta_{\mathrm{b}}) = \begin{cases} \theta_{\mathrm{bs}}\, \Phi_{\mathrm{bs}}(l_i, l_j) & (i, j) \in \mathcal{E}_s \\ \theta_{\mathrm{bt}}\, \Phi_{\mathrm{bt}}(l_i, l_j) & (i, j) \in \mathcal{E}_t, \end{cases} \quad (4)$$

where $\mathcal{E}_{\mathrm{s}}$ and $\mathcal{E}_{\mathrm{t}}$ respectively denote the set of edges representing spatial adjacency and that representing temporal adjacency. Note that $\mathcal{E} = \mathcal{E}_{\mathrm{s}} \cup \mathcal{E}_{\mathrm{t}}$ and $\mathcal{E}_{\mathrm{s}} \cap \mathcal{E}_{\mathrm{t}} = \emptyset$.

Φ_{bs} and Φ_{bt} are spatial smoothness and temporal smoothness between two regions:

$$\Phi_{\text{bs}}(l_i, l_j) = (1-\delta_{l_i l_j}) D(i,j)^{-1} \exp\left(-\beta_s \left\| F(i,t_i) - F(j,t_j) \right\|^2\right), \quad (5)$$

$$\Phi_{\text{bt}}(l_i, l_j) = \left(1-\delta_{l_i l_j}\right) \phi(i,j) \exp\left(-\beta_t \left\| F(i,t_i) - F(j,t_j) \right\|^2\right), \quad (6)$$

where δ is the Kronecker delta and $D(i, j)$ is the Euclidean distance between the two centers of regions i and j. ϕ is the ratio of the area matched by the optical flow inside the two temporally different regions [43]. $F(i, t_i)$ is the STD feature of region i (which exists in frame t_i). The parameters β_s and β_t are chosen similarly to [44] to ensure the exponential term switches appropriately between high and low contrasts:

$$\beta_s = \frac{1}{2}\Big(\sum_{(i,j)\in\mathcal{E}_s} \left\| F(i,t_i) - F(j,t_j) \right\|^2 \Big)^{-1}, \quad (7)$$

$$\beta_t = \frac{1}{2}\Big(\sum_{(i,j)\in\mathcal{E}_t} \left\| F(i,t_i) - F(j,t_j) \right\|^2 \Big)^{-1}. \quad (8)$$

We remark that to compute the weight ϕ, we first count the area transformed from a temporal segment (region) at a frame to its corresponding region at the next frame via optical flow and vice versa, and then take the average of ratios of the areas. In the temporal domain, this weight is better than the Euclidean distance because it is independent of the speed of the motion [43]. In this work, we employ the deep flow method [45] to transfer pixels in the temporal segment.

3) Saliency Inference: Saliency scores for regions are obtained in terms of labels by minimizing the energy function:

$$\hat{\boldsymbol{l}} = \arg\min_{\boldsymbol{l}} E(\boldsymbol{l}, \boldsymbol{x}; \boldsymbol{\theta}), \quad (9)$$

We minimize E in Eq. (2) by iterating the Graph Cut method [46], which shows the effectiveness in CRF-based energy minimization [47], and is popularly used for object segmentation [48], [49]. The inputs are initial label $\boldsymbol{l}$, block (with its temporal segments) $\boldsymbol{x}$, and model parameter $\boldsymbol{\theta}$. The minimization is then executed as the iterative expectation-maximization [50] scheme until convergence. In each iteration, the Graph Cut algorithm [46] is used to solve the ‘ “Min-Cut/Max-Flow” problem [51] of the graph, resulting in a new label for each vertex (region). The updated labels are used for the next iteration. After the saliency inference process, we obtain (binary) saliency maps for frames in $\boldsymbol{x}$.

IV. Experimental Settings

A. Benchmark Datasets

We evaluated the performance of our method on three public benchmark datasets: 10-Clips dataset [52], SegTrack2 dataset [53], and DAVIS dataset [54].

The 10-Clips dataset [52] has ten video sequences, each of which contains a single salient object. Each sequence in the dataset has the spatial resolution of 352×288 and consists of about 75 frames.

The SegTrack2 dataset [53] contains 14 video sequences and is originally designed for video object segmentation. A half of the videos in this dataset have multiple salient objects. This dataset is challenging in that it has background-foreground color similarity, fast motion, and complex shape deformation. Sequences in the dataset consist of about 76 frames with various resolutions.

The DAVIS dataset [54] consists of 50 high quality 854×480 spatial resolution and Full HD 1080p video sequences with about 70 frames per video, each of which has one single salient object or two spatially connected objects either with low contrast or overlapping with image boundary. This is also a challenging dataset because of frequent occurrences of occlusions, motion blur, and appearance changes. In this work, we used only 854×480 resolution video sequences.

All the datasets contain manually annotated pixel-wise ground-truth for every frame.

B. Evaluation Criteria

We evaluated the performance using Precision-Recall Curve (PRC), F-measure [55], and Mean Absolute Error (MAE).

The first two evaluation metrics are computed based on the overlapping areas between obtained results and provided ground-truth. Using a fixed threshold between 0 and 255, pairs of $(Precision, Recall)$ scores are computed and then combined to form a PRC. F-measure is a balanced measurement between $Precision$ and $Recall$ as follows:

$$F_\beta = \frac{\left(1+\beta^2\right) Precision \times Recall}{\beta^2 \times Precision + Recall}. \quad (10)$$

We remark that we set $\beta^2 = 0.3$ for F-measure, as suggested by Achanta *et al.* [55] so that precision is weighted more heavily.

For a given threshold, we binarize the saliency map to compute *Precision* and *Recall* at each frame in a video and then take the average over frames in the video. After that, the mean of the averages over the videos in a dataset is computed. F-measure is computed from the final precision and recall. When binarizing results for the comparison with the ground truth, we used F-Adap [56], which uses an adaptive threshold $\theta = \mu + \eta$ where μ and η are the mean value and the standard deviation of the saliency scores of the obtained map, and F-Max [57], which describes the maximum of F-measure scores for different thresholds from 0 to 255.

MAE, on the other hand, is the average over the frame of pixel-wise absolute differences between the ground truth GT and obtained saliency map SM:

$$\text{MAE} = \frac{1}{W \cdot H} \sum_{x=1}^{W} \sum_{y=1}^{H} \left\| SM(x,y) - GT(x,y) \right\|, \quad (11)$$

where W and H are the width and the height of the video frame. We note that MAE is also computed from the mean average value of the dataset in the same way as F-measure.

C. Implementation Details

We implemented region-based CNN, block-based CNN, and F-DNN in C/C++ using Caffe [58], and we implemented the

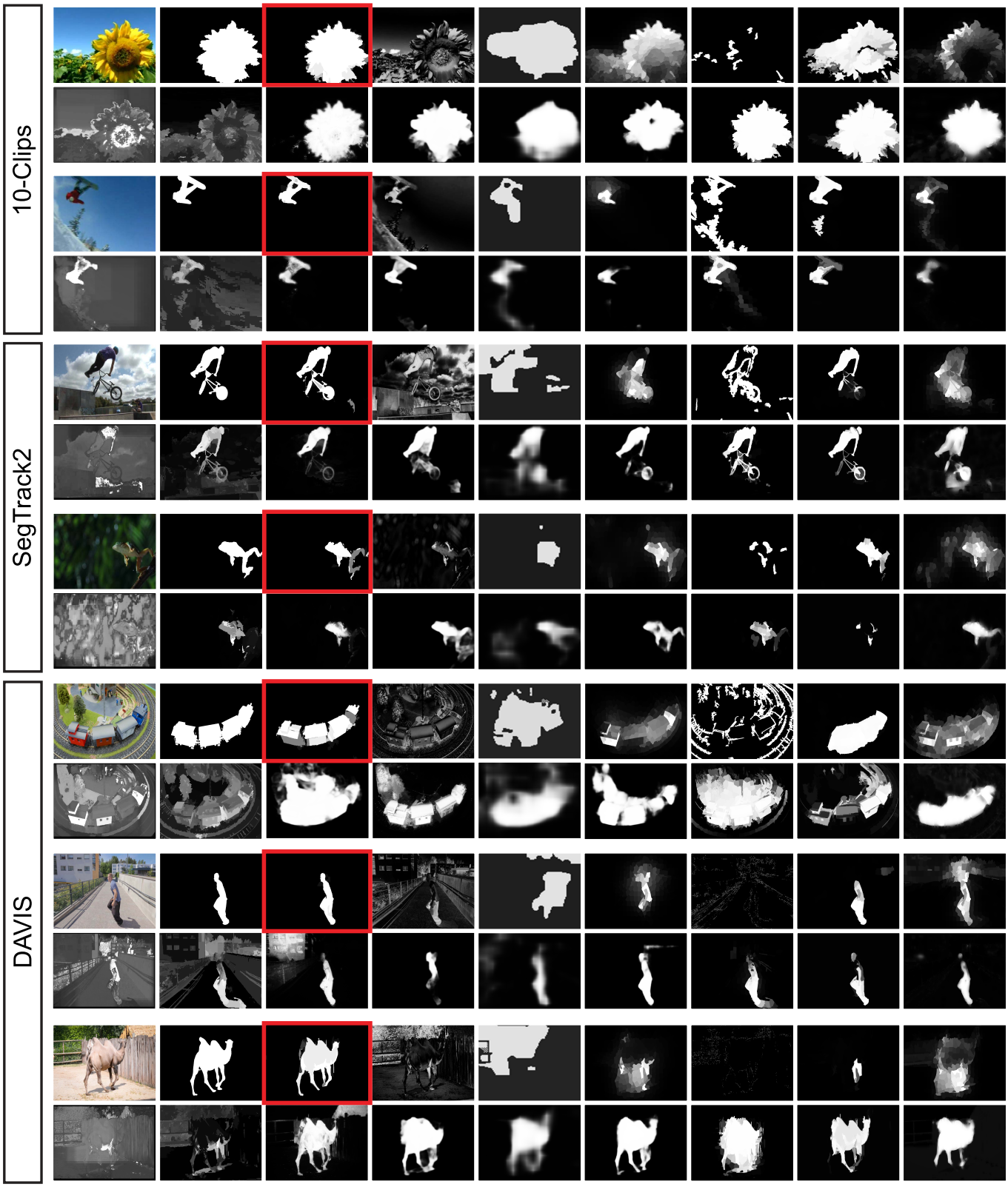
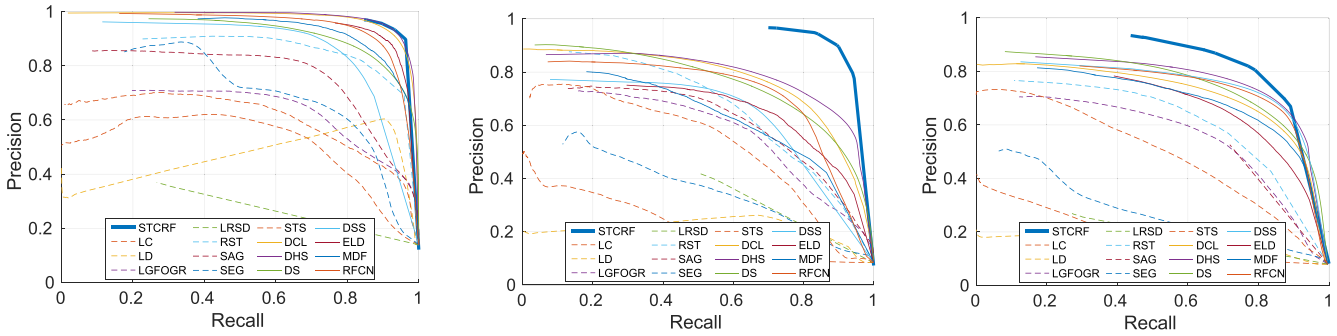


Fig. 5. Visual comparison of our method against the state-of-the-art methods. From top-left to bottom-right, original video frame and ground-truth are followed by outputs obtained using our method (STCRF), LC [21],LD [10], LGFOGR [22], LRSD [24], RST [6], SAG [23], SEG [20], STS [7], DCL [15], DHS [16], DS [28], DSS [18], ELD [17], MDF [25], and RFCN [26], in this order. Our method surrounded with red rectangles achieves the best results.

other parts in Matlab. All experiments were conducted on a PC with a Core i7 3.6GHz processor, 32GB of RAM, and GTX 1080 GPU. We remark that the region-based CNN and the block-based CNN were used without any fine-tuning.



(a) (b) (c)

Fig. 6. Quantitative comparison of precision-recall curve with state-of-the-art methods under different thresholds. Our method is denoted by STCRF (thick blue). (a) 10-Clips Dataset. (b) SegTrack2 Dataset. (c) DAVIS Dataset

TABLE II
NUMBER OF VIDEOS USED IN OUR EXPERIMENTS

Dataset	10-Clips [52]	SegTrack2 [53]	DAVIS [54]	Total
Training	6	8	30	44
Testing	4	6	20	30

To segment a video, we follow [35] as described above. We set the number of initial superpixels at each frame as $\{100, 200, 300, 400\}$ to have four scale levels. The other required parameters are set similarly to [35]. For parameters in STCRF, we empirically set $\theta = (\theta_u, \theta_{bs}, \theta_{bt}) = (50, 0.05, 1000)$. All these parameters are fixed throughout experiments.

D. Training F-DNN for Foreground Probability Prediction

In training our F-DNN (see Section III-C.2), we took an approach where we use all three datasets together rather than training our F-DNN for each dataset. This is because each dataset is too small to train a reliable model. Our approach also enables the trained model not to over-fit to a specific dataset.

From each video dataset except for the DAVIS dataset, we chose randomly 60% (in number) of videos and mixed them into a larger dataset for training while the remaining videos were used for testing each dataset (cf. Table II). For the DAVIS dataset, we used the training set and the testing set as in the DAVIS Benchmark [54].[3] We thus used 44 videos for training.

The model was fine-tuned from the network proposed in [41] using randomly initialized weights for new layers. We trained the network for 300k iterations, using the Stochastic Gradient Descent (SGD) optimization [59] with a moment $\beta = 0.9$ and a weight decay of 0.005. The size of each mini-batch is set 500. A base learning rate was initially set to 0.001 and divided by 10 at every 50k iterations.

V. EXPERIMENTAL RESULTS

A. Comparison With the State-of-the-Arts

We compared the performance of our method (denoted by STCRF) with several state-of-the-art methods for salient object detection such as LC [21], LD [10], LGFOGR [22], LRSD [24], RST [6], SAG [23], SEG [20], STS [7], DCL [15], DHS [16], DS [28], DSS [18], ELD [17], MDF [25], and RFCN [26]. Compared methods are classified in Table III. We remark that we run their original codes provided by the authors with the recommended parameter settings for obtaining results. We also note that we applied the methods developed for the still image to videos frame-by-frame.

TABLE III
COMPARED STATE-OF-THE-ART METHODS AND CLASSIFICATION

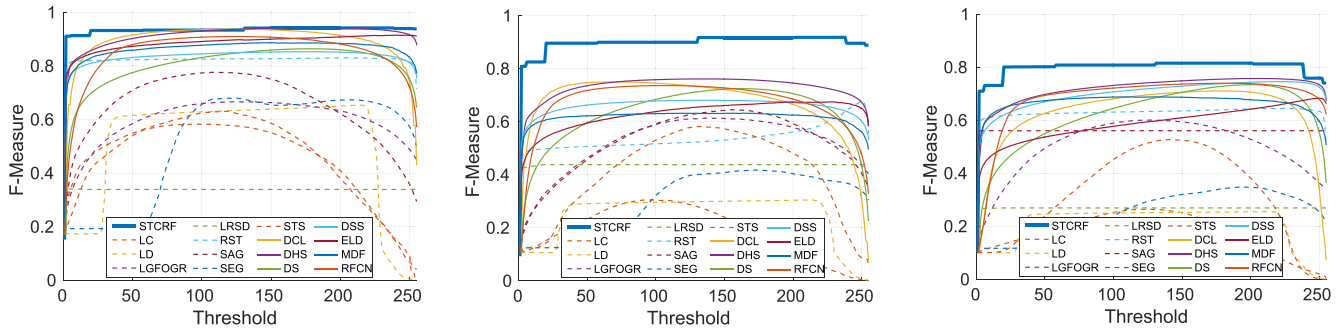
Target	Hand-crafted feature	Deep feature
Video	LC[21],LD[10], LGFOGR[22], LRSD[24], RST[6], SAG[23], SEG[20], STS[7]	None
Image	None	DCL[15], DHS[16], DS[28], DSS[18], ELD[17], MDF[25], RFCN[26]

Figure 5 shows examples of obtained results. Qualitative evaluation confirms that our method produces the best results on each dataset. Our method can handle complex foreground and background with different details, giving accurate and uniform saliency assignment. In particular, object boundaries are clearly kept with less noise, compared with the other methods.

To quantitatively evaluate the obtained results, we first computed PRC and F-measure curves, which are shown in Figs. 6 and 7.

It can be seen that our method achieves the highest precision in almost the entire recall ranges on all the datasets. Especially on the two most challenging datasets (i.e., SegTrack2 and DAVIS), the performance gains of our method against the other methods are more remarkable (results with higher recall values are less important because achieving higher recall values is easy). When compared with the second best method, i.e., DHS, we see that (1) both the methods have comparable results on 10-Clips dataset, that (2) our method is significantly better than DHS on SegTrack2 dataset, and that (3) on DAVIS dataset, the precision of our method is larger than that of DHS when recall values are small (higher binarization thresholds) while it is smaller for large recall values (lower binarization thresholds). Salient object detection at higher thresholds is more

[3] http://davischallenge.org/browse.html



(a) (b) (c)

Fig. 7. Quantitative comparison of F-measure with state-of-the-art methods under different thresholds. Our method is denoted by STCRF (thick blue). (a) 10-Clips Dataset. (b) SegTrack2 Dataset. (c) DAVIS Dataset.

TABLE IV

THE WALL-CLOCK TIME AVERAGE FOR EACH FRAME

Method	Code	Platform	Time (seconds)	FPS
STCRF	Matlab	CPU+GPU	**4.596**	**0.218**
STCRF-full	Matlab	CPU+GPU	**10.300**	**0.097**
LGFOGR [22]	Matlab	CPU	16.096	0.062
RST [6]	Matlab	CPU	19.903	0.050
SAG [23]	Matlab	CPU	17.613	0.057
STS [7]	Matlab	CPU	10.924	0.092
MDF [25]	Matlab	CPU+GPU	12.300	0.081
LD [10]	Matlab	CPU	8.318	0.120
LRSD [24]	Matlab	CPU	0.755	1.325
SEG [20]	Matlab	CPU	4.856	0.206
RFCN [26]	Matlab	CPU+GPU	1.840	0.543
LC [21]	C/C++	CPU	0.131	7.634
DCL [15]	C/C++	GPU	0.183	5.464
DHS [16]	C/C++	GPU	0.080	12.500
DS [28]	C/C++	GPU	0.109	9.174
DSS [18]	C/C++	GPU	0.178	5.618
ELD [17]	C/C++	GPU	2.030	0.493

TABLE V

THE WALL-CLOCK TIME AVERAGE OF EACH STEP FOR EACH FRAME IN THE PROPOSED METHOD. BOTTLENECKS ARE SHOWN IN RED. (THE LOCAL FEATURE EXTRACTION STREAM AND THE GLOBAL FEATURE EXTRACTION STREAM RUN IN PARALLEL)

Part	Small step	Time (seconds)
Optical flow [45]		1.265
Video segmentation [35]		4.439
STD feature extraction		
	Region-based feature extraction	2.323
	Local feature computation	0.383
	Global feature extraction	0.027
	(sub-total)	(2.706)
Saliency computation		
	Unary potential prediction	0.180
	Binary potential computation	1.461
	Saliency inference	0.249
	(sub-total)	(1.890)
Total		**10.300**

practical and effective than that at lower thresholds because with low thresholds, more pixels are segmented regardless of salient objects or background.

F-measure indicates that our method significantly outperforms the other methods at every threshold on all the datasets. Since the 10-Clips dataset is easiest among the three datasets, any methods can achieve good results while the other two datasets are challenging, meaning that the effectiveness of methods becomes discriminative. Indeed, compared with the second best method (DHS), our method is comparable on the 10-Clips dataset and significantly better on the other datasets.

Table VI illustrates the evaluations in terms of F-Adap, F-Max, and MAE. Our proposed method achieves the best performance under all the metrics on all the datasets. In particular, the outperformance of our method even against the second best method (DHS) is significant on SegTrack2 and DAVIS datasets.

B. Computational Efficiency

We further evaluated the computational time of all the methods. We compared the running-time average of our method with that of the other methods. The wall-clock time average for each frame in our method and the compared methods is given in Table IV. Our methods are denoted by STCRF for the pipeline without counting optical flow computation and video segmentation, and by STCRF-full for the full pipeline. We note that all videos were resized to the resolution of 352×288 for the fair comparison. Performances of all the methods are compared based on the implementations in C/C++ and Matlab. We classify all the methods into three categories: Matlab-region-based methods, Matlab-pixel-based methods, and C/C++ based methods.

Since it is obvious that codes implemented in C/C++ run faster than those in Matlab, we cannot directly compare the run-time of all the methods. However, we see that our method runs in the competitive speed with the others. Indeed, our method is fastest among the Matlab-region-based methods. We remark that Matlab-region-based methods run more slowly than Matlab-pixel-based ones because treating regions individually in a sequential manner and then integrating results are time-consuming.

It can be seen in Table IV that in our method, the time required for computing optical flow and video segmentation is a bottleneck: it takes 5.704 ($=10.300 - 4.596$) seconds per frame. To identify bottleneck steps in our pipeline, we broke down running-time into individual steps in our

TABLE VI

QUANTITATIVE COMPARISON WITH STATE-OF-THE-ART METHODS, USING F-MEASURE (F-ADAP AND F-MAX) (HIGHER IS BETTER) AND MEAN ABSOLUTE ERRORS (MAE) (SMALLER IS BETTER). THE BEST AND THE SECOND BEST RESULTS ARE SHOWN IN **BLUE** AND **GREEN**, RESPECTIVELY. OUR METHOD (STCRF) MARKED IN **BOLD** IS FOLLOWED BY METHODS FOR VIDEOS AND THOSE FOR STILL IMAGES

Dataset	10-Clips			SegTrack2			DAVIS		
Metric	F-Adap⇑	F-Max⇑	MAE⇓	F-Adap⇑	F-Max⇑	MAE⇓	F-Adap⇑	F-Max⇑	MAE⇓
STCRF	**0.936**	**0.942**	**0.016**	**0.899**	**0.919**	**0.014**	**0.803**	**0.816**	**0.033**
LC [21]	0.577	0.583	0.166	0.244	0.306	0.173	0.201	0.265	0.191
LD [10]	0.637	0.654	0.197	0.286	0.305	0.281	0.252	0.256	0.302
LGFOGR [22]	0.629	0.667	0.207	0.500	0.614	0.117	0.537	0.601	0.102
LRSD [24]	0.339	0.342	0.164	0.438	0.438	0.102	0.269	0.269	0.116
RST [6]	0.827	0.831	0.055	0.510	0.677	0.125	0.627	0.645	0.077
SAG [23]	0.755	0.777	0.117	0.504	0.646	0.106	0.494	0.548	0.103
SEG [20]	0.687	0.680	0.298	0.388	0.418	0.321	0.305	0.348	0.323
STS [7]	0.591	0.631	0.177	0.471	0.583	0.147	0.379	0.527	0.183
DCL [15]	**0.935**	0.937	0.031	**0.734**	0.750	0.060	0.664	0.711	0.067
DHS [16]	0.923	**0.947**	**0.022**	0.733	**0.762**	**0.050**	**0.715**	**0.758**	**0.048**
DS [28]	0.832	0.864	0.050	0.636	0.725	0.069	0.616	0.734	0.076
DSS [18]	0.838	0.853	0.049	0.662	0.681	0.054	0.690	0.746	0.049
ELD [17]	0.893	0.915	0.023	0.611	0.675	0.065	0.572	0.683	0.081
MDF [25]	0.884	0.887	0.041	0.627	0.633	0.077	0.684	0.688	0.063
RFCN [26]	0.901	0.910	0.046	0.716	0.737	0.062	0.710	0.740	0.067

TABLE VII

COMPARISON OF STD FEATURES AND LOCAL FEATURES. THE BEST RESULTS ARE SHOWN IN BLUE (HIGHER IS BETTER FOR F-ADAP AND F-MAX, AND LOWER IS BETTER FOR MAE)

Used feature	10-Clips			SegTrack2			DAVIS		
	F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓
STD feature	**0.936**	**0.942**	**0.016**	**0.899**	**0.919**	**0.014**	**0.803**	**0.816**	**0.033**
Local feature alone	0.683	0.727	0.079	0.692	0.780	0.043	0.648	0.744	0.067
RGB feature	0.882	0.913	0.044	0.366	0.454	0.080	0.160	0.186	0.199

pipeline (see Table V). We note that in our pipeline, the step of region-based feature extraction followed by local feature computation, and the step of global feature extraction run in parallel. Table V indicates that region-based feature extraction and binary potential computation are also bottlenecks. Because the bottleneck steps except for region-based feature extraction are implemented in Matlab, re-implementing such steps in C/C++ and using Cuda for parallel processing for regions will improve the speed of our method. We note that speed-up of the computational time for salient object detection is not the scope of this paper.

C. Detailed Analysis of the Proposed Method

To demonstrate the effectiveness of utilizing local and global features, utilizing spatiotemporal information in computing the saliency map, and the effectiveness of multi-level analysis, we performed experiments under controlled settings and compared results.

1) Effectiveness of Combination of Local and Global Features: To evaluate the effectiveness of combining local and global features, we compared results using STD features with those using local features alone, which is illustrated in Table VII.

We see that the combination of local and global features brings more gains than using only local features. This can be explained as follows. Local features exploit the meaning of an object in term of saliency but only in a local context, while global features can model a global context in the whole video block. Thus, STD features are more powerful. We remark that we also present results using RGB features just to confirm that the deep feature outperforms RGB features.

2) Effectiveness of Spatiotemporal Potential in STCRF: To demonstrate the effectiveness of utilizing spatiotemporal information into the energy function in STRCF, we performed experiments under four different controlled settings. We changed the binary term: setting $\theta_{\text{bt}} = 0$ to use spatial information alone (denoted by SP), setting $\theta_{\text{bs}} = 0$ to use temporal information alone (denoted by TP), and setting $\theta_{\text{bt}} = \theta_{\text{bs}} = 0$ to use the unary term alone (denoted by U). We compared the proposed (complete) method (denoted by STP) with these three baseline methods (cf. Table VIII).

Table VIII indicates that STP exhibits the best performance on all the metrics on the three datasets. We see that using both spatial and temporal information effectively works and brings more gains than using spatial information alone or using temporal information alone. This suggests that our method captures spatial contexts in a frame and temporal information over frames to produce saliency maps.

3) Effectiveness of Multiple-Scale Approach: To demonstrate the effectiveness of our multiple-scale approach, we compared methods that use different numbers of scale levels in computing the saliency map. More precisely, starting with only the coarsest scale level (level 1), we fused finer levels (levels 2, 3, 4) one by one to compute the saliency

TABLE VIII

COMPARISON OF DIFFERENT POTENTIALS IN STCRF. THE BEST RESULTS ARE SHOWN IN BLUE (HIGHER IS BETTER FOR F-ADAP AND F-MAX, AND LOWER IS BETTER FOR MAE). OUR COMPLETE METHOD ARE MARKED IN **BOLD**

Setting description	Unary term	Spatial information	Temporal information	10-Clips			SegTrack2			DAVIS		
				F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓
STP	∨	∨	∨	**0.936**	**0.942**	**0.016**	**0.899**	**0.919**	**0.014**	**0.803**	**0.816**	**0.033**
SP	∨	×	∨	0.930	0.941	0.018	0.871	0.918	0.017	0.759	0.814	0.038
TP	∨	∨	×	0.930	0.940	0.019	0.859	0.901	0.019	0.750	0.805	0.039
U	∨	×	×	0.876	0.940	0.044	0.703	0.912	0.072	0.537	0.804	0.169

TABLE IX

COMPARISON OF DIFFERENT NUMBERS OF SCALE LEVELS IN PROCESSING. THE BEST RESULTS ARE SHOWN IN BLUE (HIGHER IS BETTER FOR F-ADAP AND F-MAX, AND LOWER IS BETTER FOR MAE). OUR COMPLETE METHOD IS MARKED IN **BOLD**

Setting description	10-Clips			SegTrack2			DAVIS		
	F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓	F-Adap ⇑	F-Max ⇑	MAE ⇓
1-level	0.928	0.930	0.017	0.880	0.889	0.016	0.750	0.757	0.038
2-levels	0.929	0.940	0.017	0.876	0.908	0.015	0.763	0.800	0.035
3-levels	0.935	0.941	**0.016**	**0.909**	0.912	**0.014**	0.798	**0.816**	**0.033**
4-levels	**0.936**	**0.942**	**0.016**	0.899	**0.919**	**0.014**	**0.803**	**0.816**	**0.033**

TABLE X

COMPARISON UNDER DIFFERENT LENGTHS OF THE VIDEO BLOCK

Length of video block	10-Clips		SegTrack2		DAVIS	
	F-Adap ⇑	MAE ⇓	F-Adap ⇑	MAE ⇓	F-Adap ⇑	MAE ⇓
1 ($=2^0$)	0.934	0.017	0.890	0.016	0.791	0.035
2 ($=2^1$)	0.935	0.017	0.892	0.015	0.792	0.034
4 ($=2^2$)	0.935	0.017	0.895	0.015	0.794	0.034
8 ($=2^3$)	0.936	0.017	0.897	0.015	0.799	0.033
16 ($=2^4$)	**0.936**	**0.016**	**0.899**	**0.014**	**0.803**	**0.033**
32 ($=2^5$)	0.936	0.016	0.899	0.014	0.803	0.032
64 ($=2^6$)	0.936	0.016	0.899	0.014	0.803	0.032

map. The methods are denoted by 1-level, 2-levels, 3-levels, and 4-levels (our complete method).

The results are illustrated in Table IX. Table IX shows that the multiple-scale approach outperforms the single-scale approach. It also indicates that using more scales produces better results. Indeed, as the number of scales in the saliency computation increases, we have more accurate results. Table IX also shows that employing 4 scale levels seems to be sufficient because 3-levels and 4-levels have almost similar performances.

4) Effective Length of Video Block: We investigated the effectiveness of the size of the video block to feed to STCRF by changing the window size from 1 to 64 by twice: $1, 2, 2^2, \ldots, 2^6$ (cf. Table X).

Table X shows that as the window size becomes larger, we have more accurate results. However, the improvement in accuracy is saturated around a size of 16. On the other hand, the processing time for a larger window size is slower because the size of the graphical model becomes larger. To balance the performance between accuracy and the processing time, we observe that the appropriate window size of the video block is 16.

VI. APPLICATION TO VIDEO OBJECT SEGMENTATION

Video object segmentation (VOS) is a binary labeling problem aiming to separate foreground objects from the background of a video [54]. On the other hand, salient object detection (SOD) aims to detect and segment salient objects in natural scenes. Although VOS and SOD are different tasks, SOD methods are beneficial for VOS when salient objects are foreground objects in scenes. In this section, we demonstrate the applicability of our proposed method to VOS.

Figure 8 illustrates the framework for VOS using the saliency map. In one pass, the output saliency map is binarized using the adaptive threshold mentioned in Section IV-B to obtain the foreground mask. In the other pass, we implemented the object segmentation method based on boundary snapping [60]. We first detect contours of foreground objects using CEDN [67] and then apply the combinatorial grouping method [68] to compute the Ultrametric Contour Map (UCM) [68], which presents hierarchical segmentation. Superpixels are aligned by binarizing the UCM using a threshold $\tau = 0.3$. From the foreground mask and superpixels, we perform the majority voting to segment foreground objects.

VOS methods are classified into two groups: one that is requiring the initial object mask at the first frame, and the other that is not. In the DAVIS Benchmark [54], the former group is called semi-supervised while the latter one is unsupervised. Since an initial object mask becomes a strong prior for accurately segmenting objects in subsequent frames, we chose most recent unsupervised methods for the fair comparison. We compared our method with the state-of-the-art saliency method (DHS [16]), and most recent unsupervised VOS methods: ACO [61], CVOS [62], FST [43], KEY [63],

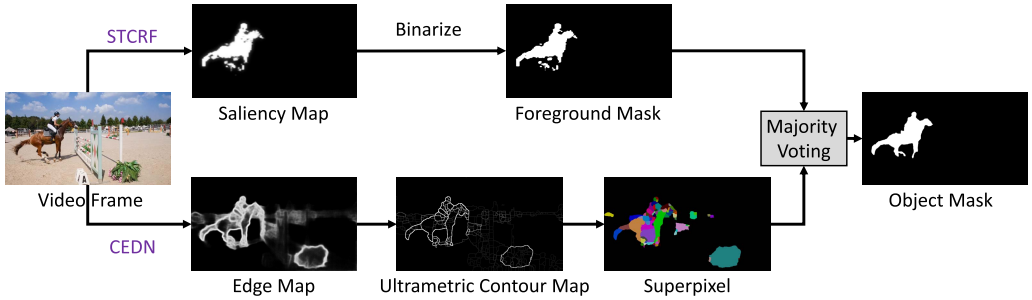


Fig. 8. Boundary snapping [60] based video object segmentation framework using the saliency map.

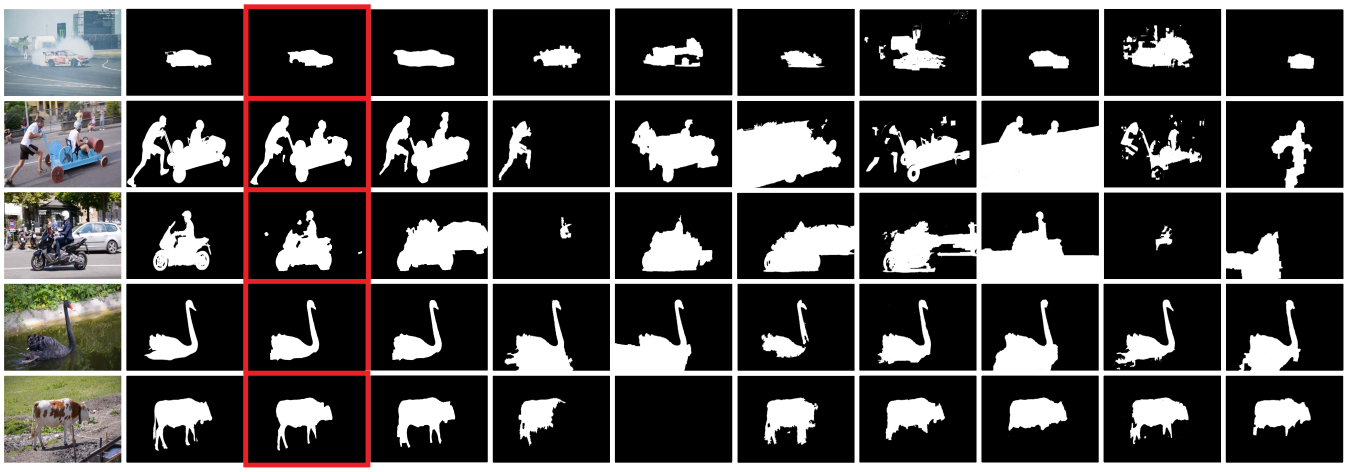


Fig. 9. Visual comparison of our method against the state-of-the-art video object segmentation methods. From left to right, original video frame and ground-truth are followed by outputs obtained using our method (STCRF*), DHS* [16], ACO [61], CVOS [62], FST [43], KEY [63], MSG [64], NLC [65], and TRC [66], in this order. Our STCRF surrounded with red rectangles achieves the best results.

TABLE XI

QUANTITATIVE COMPARISON WITH STATE-OF-THE-ART VIDEO OBJECT SEGMENTATION METHODS ON THE DAVIS DATASET, USING REGION SIMILARITY, CONTOUR ACCURACY, AND OVERALL PERFORMANCE METRICS. THE BEST THREE RESULTS ARE SHOWN IN **BLUE**, **GREEN**, AND **RED**, RESPECTIVELY. OUR METHOD, DENOTED BY STCRF*, IS MARKED IN **BOLD**

Methods	Region similarity ($\mathcal{J}$)			Contour accuracy ($\mathcal{F}$)			Overall performance ($\mathcal{O}$)
	Mean⇑	Recall⇑	Decay⇓	Mean⇑	Recall⇑	Decay⇓	Mean⇑
STCRF*	**0.714**	**0.851**	**-0.019**	**0.674**	**0.790**	**-0.019**	**0.694**
DHS* [16]	**0.701**	**0.840**	0.032	**0.656**	**0.779**	0.036	**0.679**
ACO [61]	0.503	0.572	**-0.006**	0.467	0.494	**-0.022**	0.485
CVOS [62]	0.482	0.540	0.105	0.447	0.526	0.117	0.465
FST [43]	**0.558**	**0.649**	**0.000**	0.511	0.516	**0.029**	0.535
KEY [63]	0.498	0.591	0.141	0.427	0.375	0.106	0.463
MSG [64]	0.533	0.626	0.024	0.508	**0.600**	0.051	0.521
NLC [65]	0.552	0.558	0.126	**0.523**	0.519	0.114	**0.537**
TRC [66]	0.473	0.493	0.083	0.441	0.436	0.129	0.457

MSG [64], NLC [65], and TRC [66]. We remark that two SOD methods (i.e., our method and DHS) segment objects using the framework in Fig. 8. We denote their by STCRF* and DHS* individually.

We tested all the methods on the DAVIS dataset [54], the newest dataset for VOS, and evaluated results using measures in the 2017 DAVIS Challenge [69] (i.e., region similarity $\mathcal{J}$, contour accuracy $\mathcal{F}$, and overall performance $\mathcal{O}$). For a given error measure, we computed three different statistics as in [54]. They are the mean error, the object recall (measuring the fraction of sequences scoring higher than a threshold $\tau = 0.5$), and the decay (quantifying the performance loss (or gain) over time). Note that we used the results in the DAVIS Benchmark [54][4] for the compared state-of-the-art VOS techniques. We also note that we run the source code of ACO [61], which is not mentioned in the DAVIS Benchmark, provided by the authors with the recommended parameter settings.

Figure 9 shows some examples of the obtained results. The quantitative comparison of these methods is shown in Table XI, indicating that our proposed method STCRF* exhibits the best performance on all the metrics at all the statistics. STCRF* achieves 0.714 for $\mathcal{J}$(Mean), 0.674 for $\mathcal{F}$(Mean), and 0.694 for $\mathcal{O}$(Mean), while the best VOS

[4]http://davischallenge.org/soa_compare.html

methods achieve 0.558 (for FST [43]), 0.523 (for NLC [65]), and 0.537 (for NLC [65]), respectively. STCRF* outperforms the compared VOS methods by a large margin on all the metrics. We can thus conclude that our proposed SOD method works even for VOS. We note that DHS* is second best.

VII. Conclusion

Different from the still image, the video has temporal information and how to incorporate temporal information as effectively as possible is the essential issue for dealing with the video. This paper focused on detecting salient objects from a video and proposed a framework using STD features together with STCRF. Our method takes into account temporal information in a video as much as possible in different ways, namely, feature extraction and saliency computation. Our proposed STD feature utilizes local and global contexts in both spatial and temporal domains. The proposed STCRF is capable to capture temporal consistency of regions over frames and spatial relationship between regions.

Our experiments show that the proposed method significantly outperforms state-of-the-art methods on publicly available datasets. We also applied our method to the video object segmentation task, showing that our method outperforms existing unsupervised VOS methods on the DAVIS dataset.

Visual saliency is also used for estimating human gaze [70]–[72]. For salient object detection, object boundaries should be kept as accurately as possible while for human gaze estimation, they are not. Rather, gaze fixation point should be precisely identified and the area nearby the fixation point had better be blurred to present saliency using a Gaussian kernel, for example. Applying our method directly to gaze estimation is thus not suitable. However, the idea of combining local and global features will be interesting even to gaze estimation. Adapting our proposed method to gaze estimation in videos is left for future work.

Acknowledgment

The authors are thankful to Gene Cheung for his valuable comments to improve the presentation of this paper.

References

[1] T. Lu, Z. Yuan, Y. Huang, D. Wu, and H. Yu, "Video retargeting with nonlinear spatial-temporal saliency fusion," in *Proc. ICIP*, Sep. 2010, pp. 1801–1804.

[2] M. Guo, Y. Zhao, C. Zhang, and Z. Chen, "Fast object detection based on selective visual attention," *Neurocomputing*, vol. 144, pp. 184–197, Nov. 2014.

[3] R. Zhao, W. Ouyang, and X. Wang, "Unsupervised salience learning for person re-identification," in *Proc. CVPR*, Jun. 2013, pp. 3586–3593.

[4] S. Stalder, H. Grabner, and L. Van Gool, "Dynamic objectness for adaptive tracking," in *Proc. ACCV*, 2013, pp. 43–56.

[5] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *Proc. CVPR*, Jun. 2014, pp. 2814–2821.

[6] T.-N. Le and A. Sugimoto, "Contrast based hierarchical spatial-temporal saliency for video," in *Proc. Pacific-Rim Symp. Image Video Technol. (PSIVT)*, vol. 9431, 2015, pp. 734–748.

[7] F. Zhou, S. B. Kang, and M. F. Cohen, "Time-mapping using space-time saliency," in *Proc. CVPR*, Jun. 2014, pp. 3358–3365.

[8] L. Mai, Y. Niu, and F. Liu, "Saliency aggregation: A data-driven approach," in *Proc. CVPR*, Jun. 2013, pp. 1131–1138.

[9] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, "Salient object detection: A discriminative regional feature integration approach," in *Proc. CVPR*, Jun. 2013, pp. 2083–2090.

[10] T. Liu *et al.*, "Learning to detect a salient object," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 353–367, Feb. 2011.

[11] P. Jiang, H. Ling, J. Yu, and J. Peng, "Salient region detection by ufo: Uniqueness, focusness and objectness," in *Proc. ICCV*, Dec. 2013, pp. 1976–1983.

[12] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Proc. ECCV*, 2010, pp. 140–153.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, Jun. 2014, pp. 580–587.

[14] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. ICCV*, Dec. 2015, pp. 4489–4497.

[15] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *Proc. CVPR*, Jun. 2016, pp. 478–487.

[16] N. Liu and J. Han, "DHSNet: Deep hierarchical saliency network for salient object detection," in *Proc. CVPR*, Jun. 2016, pp. 660–668.

[17] G. Lee, Y.-W. Tai, and J. Kim, "Deep saliency with encoded low level distance map and high level features," in *Proc. CVPR*, Jun. 2016, pp. 660–668.

[18] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. S. Torr, "Deeply supervised salient object detection with short connections," in *Proc. CVPR*, Jul. 2017, pp. 5300–5309.

[19] T.-N. Le and A. Sugimoto, "SpatioTemporal utilization of deep features for video saliency detection," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2017, pp. 465–470.

[20] E. Rahtu, J. Kannala, M. Salo, and J. Heikkilä, "Segmenting salient objects from images and videos," in *Proc. ECCV*, 2010, pp. 366–379.

[21] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *Proc. ACM MM*, 2006, pp. 815–824.

[22] W. Wang, J. Shen, and L. Shao, "Consistent video saliency using local gradient flow optimization and global refinement," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4185–4196, Nov. 2015.

[23] W. Wang, J. Shen, and F. Porikli, "Saliency-aware geodesic video object segmentation," in *Proc. CVPR*, Jun. 2015, pp. 3395–3402.

[24] Y. Xue, X. Guo, and X. Cao, "Motion saliency detection using low-rank and sparse decomposition," in *Proc. ICASSP*, Mar. 2012, pp. 1485–1488.

[25] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. CVPR*, Jun. 2015, pp. 5455–5463.

[26] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan, "Saliency detection with recurrent fully convolutional networks," in *Proc. ECCV*, 2016, pp. 825–841.

[27] W. Wang, J. Shen, and L. Shao, "Video salient object detection via fully convolutional networks," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 38–49, Jan. 2018.

[28] X. Li *et al.*, "DeepSaliency: Multi-task deep neural network model for salient object detection," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3919–3930, Aug. 2016.

[29] W. Shimoda and K. Yanai, "Distinct class-specific saliency maps for weakly supervised semantic segmentation," in *Proc. ECCV*, 2016, pp. 218–234.

[30] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. NIPS*, 2011, pp. 109–117.

[31] Y. Wang and Q. Ji, "A dynamic conditional random field model for object segmentation in image sequences," in *Proc. CVPR*, vol. 1, Jun. 2005, pp. 264–270.

[32] Y. Wang, K.-F. Loe, and J.-K. Wu, "A dynamic conditional random field model for foreground and shadow segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 2, pp. 279–289, Feb. 2006.

[33] R. Yi, J. Wang, and P. Tan, "Automatic fence segmentation in videos of dynamic scenes," in *Proc. CVPR*, Jun. 2016, pp. 705–713.

[34] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Proc. CVPR*, Jun. 2011, pp. 2097–2104.

[35] C.-P. Yu, H. Le, G. Zelinsky, and D. Samaras, "Efficient video segmentation using parametric graph partitioning," in *Proc. ICCV*, Dec. 2015, pp. 3155–3163.

[36] R. Girshick, "Fast R-CNN," in *Proc. ICCV*, Apr. 2015, pp. 1440–1448.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.

[38] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, Oct. 2017, pp. 2980–2988.
[39] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[40] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. CVPR*, Jun. 2014, pp. 1725–1732.
[41] L. Wang, H. Lu, X. Ruan, and M.-H. Yang, "Deep networks for saliency detection via local estimation and global search," in *Proc. CVPR*, Jun. 2015, pp. 3183–3192.
[42] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.
[43] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video," in *Proc. ICCV*, Dec. 2013, pp. 1777–1784.
[44] C. Rother, V. Kolmogorov, and A. Blake, "'GrabCut': Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.
[45] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. ICCV*, Dec. 2013, pp. 1385–1392.
[46] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.
[47] M. M. Cheng, V. A. Prisacariu, S. Zheng, P. H. S. Torr, and C. Rother, "Densecut: Densely connected CRFs for realtime grabcut," *Comput. Graph. Forum*, vol. 34, no. 7, pp. 193–201, 2015.
[48] T.-N. Le *et al.*, "Instance re-identification flow for video object segmentation," *DAVIS Challenge Video Object Segmentation CVPR Workshops*, 2017, pp. 1–6.
[49] Y.-H. Tsai, M.-H. Yang, and M. J. Black, "Video segmentation via object flow," in *Proc. CVPR*, Jun. 2016, pp. 3899–3908.
[50] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.
[51] L. R. Foulds, *Graph Theory Applications*. New York, NY, USA: Springer, 2012.
[52] K. Fukuchi, K. Miyazato, A. Kimura, S. Takagi, and J. Yamato, "Saliency-based video segmentation with graph cuts and sequentially updated priors," in *Proc. ICME*, Jun./Jul. 2009, pp. 638–641.
[53] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *Proc. ICCV*, Dec. 2013, pp. 2192–2199.
[54] 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. CVPR*, Jun. 2016, pp. 724–732.
[55] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *Proc. CVPR*, Jun. 2009, pp. 1597–1604.
[56] Y. Jia and M. Han, "Category-independent object-level saliency detection," in *Proc. ICCV*, Dec. 2013, pp. 1761–1768.
[57] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, "Salient object detection: A benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5706–5722, Dec. 2015.
[58] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM MM*, 2014, pp. 675–678.
[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699.
[60] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *Proc. CVPR*, Jul. 2017, pp. 5320–5329.
[61] W.-D. Jang, C. Lee, and C.-S. Kim, "Primary object segmentation in videos via alternate convex optimization of foreground and background distributions," in *Proc. CVPR*, Jun. 2016, pp. 696–704.
[62] B. Taylor, V. Karasev, and S. Soattoc, "Causal video object segmentation from persistence of occlusions," in *Proc. CVPR*, Jun. 2015, pp. 4268–4276.
[63] Y. J. Lee, J. Kim, and K. Grauman, "Key-segments for video object segmentation," in *Proc. ICCV*, Nov. 2011, pp. 1995–2002.
[64] P. Ochs and T. Brox, "Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions," in *Proc. ICCV*, Nov. 2011, pp. 1583–1590.
[65] A. Faktor and M. Irani, "Video segmentation by non-local consensus voting," in *Proc. BMVC*, vol. 2, no. 7, 2014, p. 8.
[66] K. Fragkiadaki, G. Zhang, and J. Shi, "Video segmentation by tracing discontinuities in a trajectory embedding," in *Proc. CVPR*, Jun. 2012, pp. 1846–1853.
[67] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proc. CVPR*, Jun. 2016, pp. 193–202.
[68] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 128–140, Jan. 2017.
[69] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. (2017). "The 2017 DAVIS challenge on video object segmentation." [Online]. Available: https://arxiv.org/abs/1704.00675
[70] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.
[71] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *Proc. NIPS*, 2006, pp. 545–552.
[72] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Proc. CVPR*, Jun. 2007, pp. 1–8.

Trung-Nghia Le received the B.S. and M.S. degrees in computer science from Vietnam National University–Ho Chi Minh City in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Informatics, SOKENDAI (The Graduate University for Advanced Studies), Tokyo, Japan.

Akihiro Sugimoto received the B.S., M.S., and Dr.Eng. degrees in mathematical engineering from The University of Tokyo. He was with Hitachi, ATR, and Kyoto University. From 2006 to 2007, he was a Visiting Professor with the University of Paris-Est, France. He is currently a Full Professor with the National Institute of Informatics, Tokyo, Japan. He is interested in mathematical methods in engineering. He has authored or co-authored over 100 peer-reviewed journal/international conference papers. His current main research interests include discrete mathematics, optimization algorithm, vision geometry, and modeling of human vision. He received the Best Paper Award from the Information Processing Society of Japan in 2001 and the Best Paper Award from the Institute of Electronics, Information and Communication Engineers in 2011. He served for several international conferences, including PSIVT2009, PSIVT2010, ACCV2012 (General Chair), ACCV2010 (Program Chair), ACCV2009, 3DV2018 (Area Chair), ICCV2009 (Tutorial Chair), ICCV2011 (Industrial Liaison Chair), 3DV2014, PSIVT2013, and PSIVT2015 (Workshop Chair). He is a regular reviewer for international conferences/journals in computer vision.