Temporal Superpixels Based on Proximity-Weighted Patch Matching

Se-Ho Lee
Korea University
seholee@mcl.korea.ac.kr

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

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

Abstract

A temporal superpixel algorithm based on proximity-weighted patch matching (TS-PPM) is proposed in this work. We develop the proximity-weighted patch matching (PPM), which estimates the motion vector of a superpixel robustly, by considering the patch matching distances of neighboring superpixels as well as the target superpixel. In each frame, we initialize superpixels by transferring the superpixel labels of the previous frame using PPM motion vectors. Then, we update the superpixel labels of boundary pixels, based on a cost function, composed of color, spatial, contour, and temporal consistency terms. Finally, we execute superpixel splitting, merging, and relabeling to regularize superpixel sizes and reduce incorrect labels. Experiments show that the proposed algorithm outperforms the state-of-the-art conventional algorithms significantly.

1. Introduction

Superpixel segmentation divides an image into a set of meaningful regions. It reduces the number of image primitives greatly to execute computer vision techniques efficiently. Superpixels are used in various vision tasks, *e.g.* image segmentation [16], video object segmentation [10], and saliency detection [23]. While many superpixel methods for static images have been proposed [1, 8, 12–15, 17, 19, 25, 33, 34], there are a relatively small number of temporal superpixel methods for video processing [1, 5, 12, 22, 26].

In general, both superpixel and temporal superpixel methods divide an image into superpixels by minimizing cost functions, which make each superpixel belong to a single object and not overlap with other objects. However, if a superpixel method is applied to each frame in a video sequence independently, the resultant superpixels may be temporally inconsistent. Thus, temporal superpixel methods [1, 5, 12, 22, 26] use color or motion information of neighboring frames to generate temporally consistent superpixels. In particular, to reflect object movements, [5, 12, 22] exploit optical flow information between frames. However, extracting optical flow information requires high computational complexity.

In this work, we propose a temporal superpixel algorithm based on proximity-weighted patch matching (TS-PPM). First, we perform the proximity-weighted patch matching (PPM) to estimate superpixel motion vectors efficiently. In PPM, to determine the motion vector of a superpixel, we consider the patch matching distances of neighboring superpixels, as well as that of the target superpixel. We then sum up those matching distances using the proximity between the target and neighboring superpixels. Second, we initialize the superpixel label of each pixel in a frame, by mapping the superpixel labels in the previous frames using the PPM motion vectors. Third, we refine the initial superpixels by updating the superpixel labels of boundary pixels iteratively based on a cost function. Finally, we perform superpixel splitting, merging, and relabeling to prevent irregular superpixel sizes and incorrect labeling. Experimental results show that the proposed TS-PPM algorithm outperforms the conventional algorithms [1, 5, 8, 12–15, 17, 19, 22, 25, 26, 33, 34], while demanding lower computational complexity than most of the conventional algorithms. Furthermore, we demonstrate that the proposed algorithm is applicable to video object segmentation [10, 11, 20] and saliency detection [27, 32].

This paper has three main contributions.

- We propose PPM that robustly estimates the motion vector of a superpixel using the information in neighboring superpixels, as well as the target superpixel.
- We develop the temporal superpixel labeling scheme, based on the cost function, which enforces superpixels to be temporally consistent.
- The proposed algorithm shows remarkable performances on the superpixel datasets [6, 18, 24] and is applicable to many computer vision tasks.

2. Related Work

2.1. Superpixel Methods

Many superpixel methods have been proposed. Levinshtein *et al.* [13] proposed Turbopixels, which initializes seeds based on the gradient information and propagates them using the level-set methods.

K-means optimization methods, which assign each pixel to the nearest cluster and update the cluster centroids iteratively, have been proposed [1, 14, 17]. Achanta *et al*. [1] represented each pixel with spatial and color coordinates. Li and Chen [14] adopted the cost function of the normalized cuts in their K-means optimization. Liu *et al*. [17] proposed content-sensitive superpixels, by measuring the areas of Voronoi cells based on the content density. Thus, they assigned small superpixels in content-rich regions and large superpixels in homogeneous regions.

Liu *et al*. [15] proposed an entropy-based superpixel method. They first constructed a graph on an image and formulated a cost function, composed of the entropy rate of a random walker on the graph and a balancing term. While the entropy rate enforces superpixels to be compact and homogeneous, the balancing term imposes superpixels to have similar sizes.

Coarse-to-fine superpixel methods, which update the labels of superpixel boundary regions in multiple scales from block to pixel levels, have been proposed [25, 33]. Van den Bergh *et al*. [25] updated superpixel labels to enhance the color homogeneity within each superpixel. Yao *et al*. [33] employed the shape regularization term to make superpixels regular in shape and the boundary length term to reduce superpixel boundary lengths.

Also, contour-based superpixel methods have been proposed in [8, 12, 19, 34]. Moore *et al*. [19] and Fu *et al*. [8] determined paths containing many object contour pixels and divided superpixels along the paths. Zeng *et al*. [34] proposed a superpixel method using geodesic distances. They computed geodesic distances, based on the contour information, obtained from gradient magnitudes. Then, they assigned each pixel to the seed that has the smallest geodesic distance and updated the position of each seed iteratively. Lee *et al*. [12] adopted the contour constraint to compel superpixel boundaries to be compatible with object contours, extracted by the learning-based contour detector [29]. Specifically, they determined the contour constraint by matching the extracted contour map with the contour pattern set, constructed from the ground-truth contour maps of training images.

2.2. Temporal Superpixel Methods

Compared with superpixel methods for image processing, only a small number of temporal superpixel methods have been proposed for video processing. Achanta *et al*. [1] and Van den Bergh *et al*. [26] extended their superpixel methods for video processing, respectively. Achanta *et al*. [1] simply carried out the K-means optimization on a three-dimensional signal, obtained by concatenating 2D images along the time axis. Van den Bergh *et al*. [26] also extended their superpixel method [25]. They constructed color histograms, by considering previous frames as well as the current frame, to yield superpixels with temporally consistent colors. They also created and terminated superpixel labels to cope with color variation and object occlusion. These extended schemes [1, 26], however, may be ineffective when there exists fast motion in the scene since they do not consider object movements.

To address object motion, [5, 12, 22] use optical flow information to determine the center of each superpixel. Reso *et al*. [22] performed the K-means optimization, as in [1], to assign superpixel labels. When calculating the average color of each superpixel, they adopted a temporal sliding window, containing the past and future frames as well as the current frame. Thus, their method can yield temporally consistent superpixels. Chang *et al*. [5] proposed another temporal superpixel method, by employing a generative probabilistic model. They initialized motion vectors of superpixels using optical flow information and then refined the motion vectors based on a bilateral kernel. Lee *et al*. [12] initialized the superpixel labels of a frame by transferring those of the previous frame using the optical flow information. Thus, they can label the same regions in consecutive frames consistently. Although [5, 12, 22] can track objects based on the motion information, they demand high computational complexities for the motion estimation.

3. Proposed Superpixel Algorithm

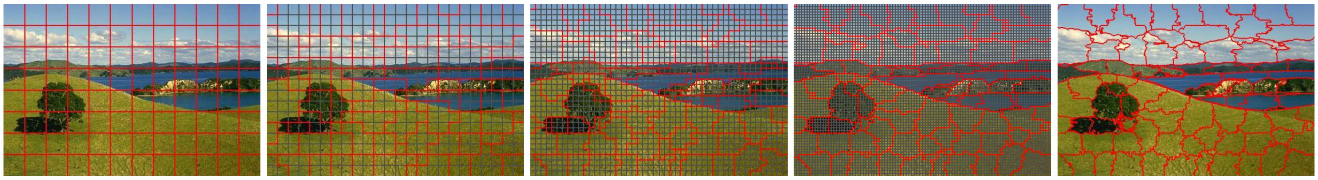
This section proposes a superpixel algorithm for image processing. For video processing, the proposed superpixel algorithm is applied to the first frame. Then, based on high correlations between frames, the resultant superpixels are used to generate superpixels for subsequent frames, as will be described in Section 4.

Let $S_1, \ldots, S_K$ denote superpixels in an image, and $l(\mathbf{p}) \in \{1, \ldots K\}$ be the superpixel label of pixel $\mathbf{p}$. Thus, notice that pixel $\mathbf{p}$ belongs to superpixel $S_{l(\mathbf{p})}$. For over-segmenting an image into superpixels, we iteratively update the superpixel labels of boundary pixels, which are located at superpixel boundaries.

We update the superpixel label of a boundary pixel $\mathbf{p}$ from $l(\mathbf{p})$ to $l(\mathbf{q})$ of a neighboring pixel $\mathbf{q} \in \mathcal{N}_\mathbf{p}$, which has the smallest cost $E(\mathbf{p}, \mathbf{q})$. Here, $\mathcal{N}_\mathbf{p}$ denotes the set of the 4-neighbors of $\mathbf{p}$. To preserve the topological relationship among superpixels, as in [25, 33], changing the superpixel label of $\mathbf{p}$ is allowed only when $\mathbf{p}$ is a simple point [4]. We formulate the cost function $E(\mathbf{p}, \mathbf{q})$ for updating the superpixel label of $\mathbf{p}$ from $l(\mathbf{p})$ to $l(\mathbf{q})$ as

$$E(\mathbf{p}, \mathbf{q}) = [E_\mathrm{C}(\mathbf{p}, \mathbf{q}) + E_\mathrm{L}(\mathbf{p}, \mathbf{q})] \times E_\mathrm{O}(\mathbf{p}, \mathbf{q}). \quad (1)$$

Let us describe each term in (1) subsequently.



(a) Initial superpixels (b) Level 1 (c) Level 2 (d) Level 3 (e) Level 4

Figure 1. Illustration of the coarse-to-fine superpixel labeling ($K = 96$). Red lines are superpixel boundaries, while black lines in (b)∼(d) are block boundaries. The superpixel labeling is done at block levels in (b)∼(d), and at the pixel level in (e).

3.1. Color Distance

We calculate the color distance between pixel $\mathbf{p}$ and superpixel $S_{l(\mathbf{q})}$ by

$$E_{\mathrm{C}}(\mathbf{p},\mathbf{q}) = \|\mathbf{c}(\mathbf{p}) - \boldsymbol{\mu}_{\mathrm{C}}(l(\mathbf{q}))\|^2 \tag{2}$$

where $\mathbf{c}(\mathbf{p})$ is the color of $\mathbf{p}$, and $\boldsymbol{\mu}_{\mathrm{C}}(l(\mathbf{q}))$ denotes the average color of pixels within superpixel $S_{l(\mathbf{q})}$. In this work, the CIELAB color space is used. The color distance $E_{\mathrm{C}}(\mathbf{p},\mathbf{q})$ is included in the overall cost function in (1) to enforce pixel $\mathbf{p}$ to be assigned to a superpixel with a similar color.

3.2. Spatial Distance

We also use the spatial distance between $\mathbf{p}$ and $S_{l(\mathbf{q})}$,

$$E_{\mathrm{L}}(\mathbf{p},\mathbf{q}) = \|\mathbf{p} - \boldsymbol{\mu}_{\mathrm{L}}(l(\mathbf{q}))\|^2 \tag{3}$$

where $\boldsymbol{\mu}_{\mathrm{L}}(l(\mathbf{q}))$ is the average position of pixels within $S_{l(\mathbf{q})}$. Thus, the reduction of $E_{\mathrm{L}}(\mathbf{p},\mathbf{q})$ has the effect of assigning $\mathbf{p}$ to a neighboring superpixel $S_{l(\mathbf{q})}$ whose centroid is near to $\mathbf{p}$. $E_{\mathrm{L}}(\mathbf{p},\mathbf{q})$ is incorporated into (1) to yield compactly shaped superpixels.

3.3. Contour Term

Superpixels should adhere to object contours, since they are used as the smallest meaningful units in applications. In other words, a superpixel should not overlap with multiple objects. We adopt the contour term $E_{\mathrm{O}}(\mathbf{p},\mathbf{q})$ in (1) to make superpixel boundaries compatible with object contours.

We employ the holistically-nested edge detection (HED) scheme [29] to obtain a contour map of the input image, as done in [12]. Whereas [12] performs the highly complicated pattern matching to exploit contour information, we simply define the contour term $E_{\mathrm{O}}(\mathbf{p},\mathbf{q})$, based on the contour map, as

$$E_{\mathrm{O}}(\mathbf{p},\mathbf{q}) = h(\mathbf{q}) + \|\mathbf{c}(\mathbf{p}) - \mathbf{c}(\mathbf{q})\|^2 \tag{4}$$

where $h(\mathbf{q})$ denotes the HED contour response for pixel $\mathbf{q}$. The contour term $E_{\mathrm{O}}(\mathbf{p},\mathbf{q})$ gets larger when $\mathbf{q}$ has a higher contour response or the color difference between $\mathbf{p}$ and $\mathbf{q}$ is larger. Therefore, $E_{\mathrm{O}}(\mathbf{p},\mathbf{q})$ amplifies the cost function in (1), when there is an object contour on $\mathbf{q}$. The contour term prevents each superpixel from overlapping with multiple objects in general.

Algorithm 1 Superpixel Algorithm

```
Initialize superpixels in a regular grid
for level = 1 to 4 do
    repeat for all simple points p
        q* ← arg min_q E(p, q) and l(p) ← l(q*)        ▷ (1)
        Update the average colors and positions of superpixels
    until convergence or pre-defined number of iterations
end for
```

3.4. Coarse-to-Fine Superpixel Labeling

We adopt a coarse-to-fine scheme, as in [25, 33]. We first initialize K superpixels in a regular grid in Figure 1(a). Then, we update superpixel labels at four levels. While the update is carried out using blocks at levels 1∼3, it is done using pixels at level 4. At the coarsest level 1 in Figure 1(b), we divide each regular superpixel in Figure 1(a) into four blocks and update their superpixel labels. We further divide the blocks at levels 2 and 3 and perform the block label update in Figures 1(c) and (d). Finally, at level 4, we update the labels of boundary pixels in Figure 1(e).

For the block label update at levels 1∼3, we modify the cost function in (1) accordingly. In (2)∼(4), we replace the colors and positions of pixels with the average colors and positions of blocks. Also, in (4), we replace the contour response of a pixel with the maximum response of pixels in a block.

Algorithm 1 summarizes the proposed superpixel algorithm. At each level, we terminate the iterative update, when there is no label change or the maximum number of iterations are performed. The maximum number is set to 10.

4. Proposed Temporal Superpixel Algorithm

We extend the superpixel algorithm in Section 3 to generate temporal superpixels. The proposed TS-PPM algorithm generates superpixels for frame $I^{(t)}$ based on the superpixel label information for frame $I^{(t-1)}$.

4.1. Proximity-Weighted Patch Matching

In [5, 22], dense optical flow estimation is performed to determine the motion of each superpixel, but it is computationally expensive to obtain dense optical flow vectors for all pixels. Moreover, dense vectors are not necessary for

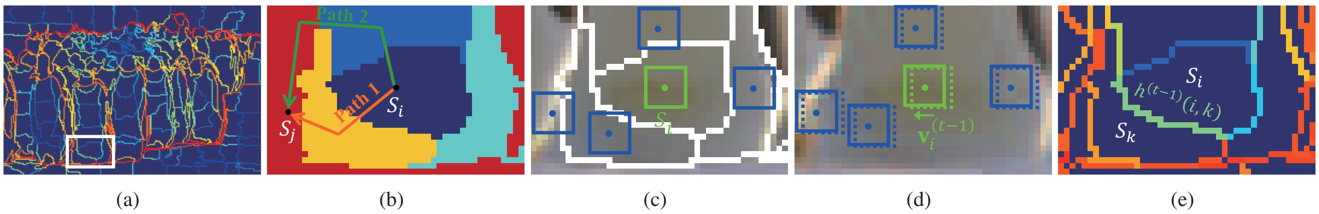


Figure 2. An example of PPM: (a) contour map for $I^{(t-1)}$, (b) paths between S_i and S_j in $I^{(t-1)}$, (c) patches in $I^{(t-1)}$, (d) displaced patches in $I^{(t)}$, and (e) boundary contour strength $h^{(t-1)}(i,k)$. Note that (b), (c), and (e) show the enlarged regions of the white box in (a).

extracting the motion of a superpixel in this work. This is because the superpixels in $I^{(t-1)}$ are already divided using the contour term in (4). Thus, each superpixel in $I^{(t-1)}$ belongs to a single object in general, and all pixels in the superpixel tend to have the same motion. Hence, instead of the dense estimation, we estimate the motion vector of each superpixel, by performing patch matching [3, 9] sparsely. For robust motion estimation, we develop PPM in this work.

Figure 2 illustrates PPM. In Figure 2(b), colors represent the superpixel labels of regions within the white box in Figure 2(a). As shown in Figure 2(c), we consider patches centered at the average positions of the superpixels in Figure 2(b). Suppose that we estimate the motion vector of a target superpixel S_i from frame $I^{(t-1)}$ to frame $I^{(t)}$. To this end, we use patches for neighboring superpixels, as well as that for the target superpixel S_i. The neighboring patches are depicted by blue squares in Figure 2(c). Given a candidate motion vector $\mathbf{v}$ of S_i, the target and neighboring patches are matched to the displaced patches in $I^{(t)}$ in Figure 2(d). Then, the patch matching distances for the target and neighboring patches are summed up with weights. A high weight is assigned to a neighboring patch that is more similar to the target patch. To quantify the similarity, we compute the proximity between superpixels, using the patch matching distance and the contour distance, as follows.

First, the intra-frame patch matching distance $d_{\mathrm{P}}(i,j)$ between superpixels S_i and S_j in frame $I^{(t-1)}$ is given by

$$d_{\mathrm{P}}(i,j) = \sum_{\boldsymbol{\Delta}} \|\mathbf{c}^{(t-1)}(\boldsymbol{\mu}_{\mathrm{L}}^{(t-1)}(i)+\boldsymbol{\Delta}) - \mathbf{c}^{(t-1)}(\boldsymbol{\mu}_{\mathrm{L}}^{(t-1)}(j)+\boldsymbol{\Delta})\|^2 \tag{5}$$

where $\mathbf{c}^{(t-1)}(\mathbf{p})$ is the color of $\mathbf{p}$ in $I^{(t-1)}$, and $\boldsymbol{\mu}_{\mathrm{L}}^{(t-1)}(i)$ is the average position of pixels within S_i in $I^{(t-1)}$. In (5), $\boldsymbol{\Delta} = (\Delta x, \Delta y)$ and the summation is over $-m \leq \Delta x, \Delta y \leq m$. We set $m = 3$ to use 7×7 patches.

Second, we compute the contour distance $d_{\mathrm{O}}(i,j)$ between S_i and S_j in $I^{(t-1)}$. Suppose that there are two adjacent superpixels, sharing a boundary. Then, we define the boundary contour strength between them as the average HED contour response of the pixels on the shared boundary. Figure 2(e) is the magnified contour map of Figure 2(a) and illustrates how to compute the boundary contour strength $h^{(t-1)}(i,k)$ between S_i and S_k. There are many paths from S_i to S_j, as shown in Figure 2(b). Among them, we find the optimal path that yields the minimum sum of the boundary contour strengths. Similar to the geodesic distance, the minimum sum becomes the contour distance $d_{\mathrm{O}}(i,j)$. Specifically, let an index sequence $\mathbf{u} = (u_1 = i, u_2, \ldots u_{N-1}, u_N = j)$ denote a path from S_i to S_j, where S_{u_n} and $S_{u_{n+1}}$ are adjacent superpixels for $n = 1, \ldots, N-1$. Then, we have the contour distance

$$d_{\mathrm{O}}(i,j) = \min_{\mathbf{u}=(u_1=i,\ldots,u_N=j)} \sum_{n=1}^{N-1} h^{(t-1)}(u_n, u_{n+1}). \tag{6}$$

If there exist object boundaries in every path connecting superpixels S_i and S_j, $d_{\mathrm{O}}(i,j)$ has a high value.

Then, the proximity $w(i,j)$ between superpixels S_i and S_j is computed as

$$w(i,j) = \exp\left(-\frac{d_{\mathrm{P}}(i,j)}{\sigma_{\mathrm{P}}^2} - \frac{d_{\mathrm{O}}(i,j)}{\sigma_{\mathrm{O}}^2}\right) \tag{7}$$

where the scaling parameters are $\sigma_{\mathrm{P}}^2 = 2.0$ and $\sigma_{\mathrm{O}}^2 = 0.15$. S_j is declared to be a neighboring superpixel of S_i if $w(i,j)$ is higher than a threshold $\tau_{\mathrm{w}} = 0.01$.

Finally, we estimate the motion vector $\mathbf{v}_i^{(t-1)}$ of superpixel S_i to yield the minimum PPM distance from $I^{(t-1)}$ to frame $I^{(t)}$, given by

$$\mathbf{v}_i^{(t-1)} = \arg\min_{\mathbf{v}} \sum_j \sum_{\boldsymbol{\Delta}} \{w(i,j) \times \|\mathbf{c}^{(t-1)}(\boldsymbol{\mu}_{\mathrm{L}}^{(t-1)}(j)+\boldsymbol{\Delta}) - \mathbf{c}^{(t)}(\boldsymbol{\mu}_{\mathrm{L}}^{(t-1)}(j)+\mathbf{v}+\boldsymbol{\Delta})\|^2\} \tag{8}$$

where the summation over j is for the neighboring superpixels S_j of S_i. Using the neighboring superpixels, PPM estimates the motion of S_i robustly.

4.2. Temporal Superpixel Labeling

We initialize superpixel labels of frame $I^{(t)}$, by transferring those of frame $I^{(t-1)}$ using the motion vectors from $I^{(t-1)}$ to $I^{(t)}$, estimated by PPM. During the initialization, we do not assign superpixel labels to occluded or disoccluded pixels $I^{(t)}$. We regard a pixel mapped from multiple pixels in frame $I^{(t-1)}$ as occluded, and a pixel mapped from no pixel as disoccluded. For example, Figure 3(c) shows the initial superpixel label map of frame $I^{(t)}$, transferred from

(a) (b) (c)

Figure 3. Initialization of a superpixel label map: (a) frame $I^{(t-1)}$, (b) superpixel label map for $I^{(t-1)}$, and (c) initial label map for $I^{(t)}$.

the label map of frame $I^{(t-1)}$ in Figure 3(b). In Figure 3(c), occluded or disoccluded pixels are colored in black.

After the initialization, we perform the temporal superpixel labeling in a similar manner to Section 3. However, the labels are updated at the pixel level only. The cost function $E(\mathbf{p}, \mathbf{q}, t)$ for updating the superpixel label of a boundary pixel $\mathbf{p}$ from $l(\mathbf{p})$ to $l(\mathbf{q})$ in frame $I^{(t)}$ is defined as

$$E(\mathbf{p},\mathbf{q},t) = [E_{\mathrm{C}}(\mathbf{p},\mathbf{q},t) + E_{\mathrm{L}}(\mathbf{p},\mathbf{q},t)] \times E_{\mathrm{O}}(\mathbf{p},\mathbf{q},t) \times E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t) \tag{9}$$

where $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ is the temporal consistency term. The color distance $E_{\mathrm{C}}(\mathbf{p},\mathbf{q},t)$, the spatial distance $E_{\mathrm{L}}(\mathbf{p},\mathbf{q},t)$, and the contour term $E_{\mathrm{O}}(\mathbf{p},\mathbf{q},t)$ are defined in the same way as (2), (3), and (4), respectively. However, when calculating the color distance $E_{\mathrm{C}}(\mathbf{p},\mathbf{q},t)$, we use the average color of $S_{l(\mathbf{q})}$ from frame $I^{(1)}$ to frame $I^{(t-1)}$ to make superpixels contain temporally consistent color information.

4.3. Temporal Consistency Term

We adopt the temporal consistency term $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ in (9) to label the same region consistently between $I^{(t-1)}$ and $I^{(t)}$. Let $\mathcal{L}_{\mathbf{p}}$ be the set of the superpixel labels that are mapped to pixel $\mathbf{p}$ in $I^{(t)}$ from $I^{(t-1)}$ using the superpixel motion vectors. Notice that $\mathcal{L}_{\mathbf{p}}$ consists of a single label in a normal case, and multiple labels in an occluded case. Also, $\mathcal{L}_{\mathbf{p}}$ is empty when $\mathbf{p}$ is disoccluded. The consistency term $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ is lowered, when the superpixel label $l(\mathbf{q})$ belongs to $\mathcal{L}_{\mathbf{p}}$. In other words, it is encouraged to update the label of $\mathbf{p}$ with an element in $\mathcal{L}_{\mathbf{p}}$.

However, the motion estimation is not perfect, and a superpixel label $k \in \mathcal{L}_{\mathbf{p}}$ may be mapped from $I^{(t-1)}$ with an incorrect motion vector $\mathbf{v}_k^{(t-1)}$. We hence determine the reliability of $\mathbf{v}_k^{(t-1)}$ at $\mathbf{p}$ as follows. First, we compute the backward patch matching distance of $\mathbf{v}_k^{(t-1)}$ at $\mathbf{p}$ in $I^{(t)}$,

$$\rho(\mathbf{p},k,t) = \sum_{\mathbf{\Delta}} \|\mathbf{c}^{(t)}(\mathbf{p}+\mathbf{\Delta}) - \mathbf{c}^{(t-1)}(\mathbf{p}-\mathbf{v}_k^{(t-1)}+\mathbf{\Delta})\|^2. \tag{10}$$

A higher $\rho(\mathbf{p},k,t)$ indicates that the motion vector $\mathbf{v}_k^{(t-1)}$ and the corresponding label k are less reliable. Second, we cross-check $\mathbf{v}_k^{(t-1)}$, as in [7]. Let $\tilde{\mathbf{v}}_k^{(t)}$ be the estimated backward motion vector of superpixel S_k from $I^{(t)}$

Algorithm 2 Temporal Superpixel Algorithm (TS-PPM)

```
Apply Algorithm 1 to I^(1)
for t = 2 to t_end do
    Perform PPM from I^(t−1) to I^(t)
    Initialize superpixels using the superpixel results in I^(t−1)
    repeat for all simple points p in I^(t)
        q* = arg min_q E(p, q, t)                    ▷ (9)
        l(p) ← l(q*)
        Update the average positions of superpixels
    until convergence or pre-defined number of iterations
    Perform superpixel merging, splitting, and relabeling
end for
(Optional) Perform backward refinement
```

to $I^{(t-1)}$. In the ideal case, $\tilde{\mathbf{v}}_k^{(t)}$ should be equal to $-\mathbf{v}_k^{(t-1)}$. Thus, we define the cross-check discrepancy $\zeta(k,t)$ as

$$\zeta(k,t) = \|\mathbf{v}_k^{(t-1)} + \tilde{\mathbf{v}}_k^{(t)}\|^2. \tag{11}$$

A higher $\zeta(k,t)$ also indicates that the superpixel label k is less reliable. By combining (10) and (11), we define the reliability $r(\mathbf{p},k,t)$ of assigning label $k \in \mathcal{L}_{\mathbf{p}}$ to pixel $\mathbf{p}$ as

$$r(\mathbf{p},k,t) = \exp\left(-\frac{\rho(\mathbf{p},k,t)}{\sigma_\rho^2} - \frac{\zeta(k,t)}{\sigma_\zeta^2}\right) \tag{12}$$

where $\sigma_\rho^2 = 2$ and $\sigma_\zeta^2 = 12$ in all experiments.

Based on the reliability in (12), the temporal consistency term $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ is defined as

$$E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t) = \begin{cases} \frac{1}{1+r(\mathbf{p},l(\mathbf{q}),t)} & \text{if } l(\mathbf{q}) \in \mathcal{L}_{\mathbf{p}}, \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

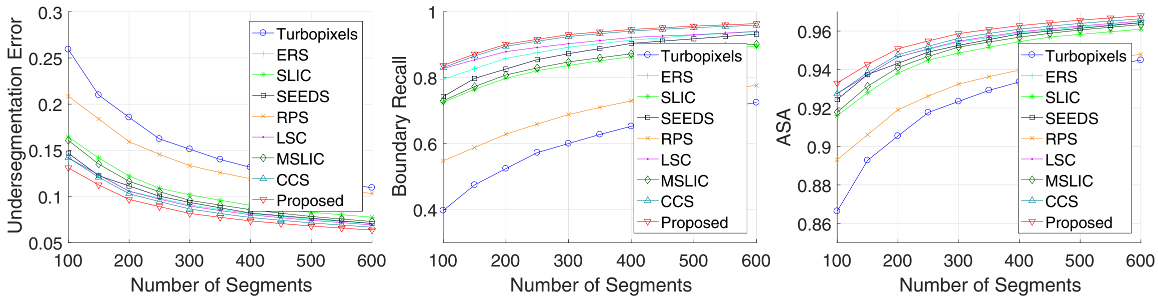
$E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ decreases the overall cost $E(\mathbf{p},\mathbf{q},t)$ in (9) when the reliability $r(\mathbf{p},l(\mathbf{q}),t)$ is high. Note that, when $l(\mathbf{q})$ is not mapped from $I^{(t-1)}$ using any motion vector, $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ has the maximum value of 1. Thus, $E_{\mathrm{T}}(\mathbf{p},\mathbf{q},t)$ facilitates temporally consistent labeling.

4.4. Splitting, Merging, and Relabeling

As the superpixel labeling is performed frame by frame, some superpixels may grow or shrink. Splitting and merging superpixels are required to regularize superpixel sizes. Also, some superpixels can be mislabeled due to occlusion or imperfect motion estimation. They should be relabeled as new ones. To address these issues, we follow the superpixel splitting, merging, and relabeling strategies in [12].

4.5. Optional Backward Refinement

After generating temporal superpixels sequentially from the first frame $I^{(1)}$ to the last frame $I^{(t_{\mathrm{end}})}$ in a video, we optionally refine the generated superpixels backwardly from $I^{(t_{\mathrm{end}})}$ to $I^{(1)}$ to further improve temporal consistency.

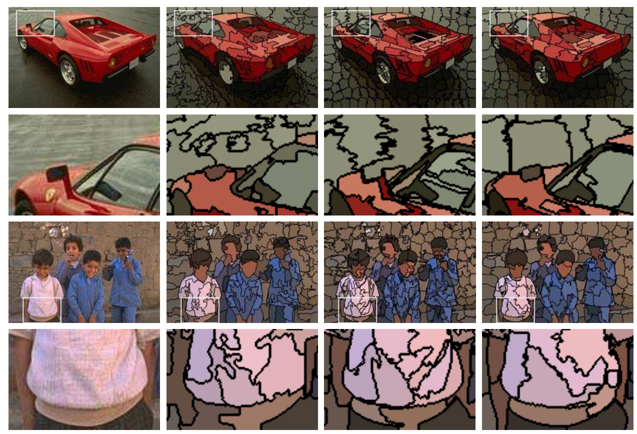


(a) UE ↓ (b) BR ↑ (c) ASA ↑

Figure 4. Quantitative comparison of superpixel algorithms.

The backward refinement is performed similarly to the forward pass, by updating the superpixel label of $\mathbf{p}$ in $I^{(t)}$ using the cost function in (9). However, when computing the color distance $E_{\mathrm{C}}(\mathbf{p}, \mathbf{q}, t)$, we use the average color of each superpixel from $I^{(t+1)}$ to $I^{(t_{\text{end}})}$, instead of that from $I^{(1)}$ to $I^{(t-1)}$. Also, for the consistency term $E_{\mathrm{T}}(\mathbf{p}, \mathbf{q}, t)$, we use the backward motion vector $\tilde{\mathbf{v}}_k^{(t+1)}$ from $I^{(t+1)}$ to $I^{(t)}$, instead of the forward motion vector. $E_{\mathrm{L}}(\mathbf{p}, \mathbf{q}, t)$ and $E_{\mathrm{O}}(\mathbf{p}, \mathbf{q}, t)$ are not modified.

Algorithm 2 summarizes the proposed TS-PPM algorithm. Without the optional backward refinement, the proposed algorithm operates online in that it generates superpixels for a frame causally using the information in the current and past frames only. With the optional refinement, it becomes an offline approach using the entire sequence.



(a) Input (b) SEEDS (c) LSC (d) Proposed

Figure 5. Visual comparison of superpixel results. Each image consists of about 200 superpixels. The second and last rows show the enlarged parts of the images in the first and third rows, respectively. In (b)∼(d), a superpixel is represented by the average color.

5. Experimental Results

5.1. Superpixel Algorithm

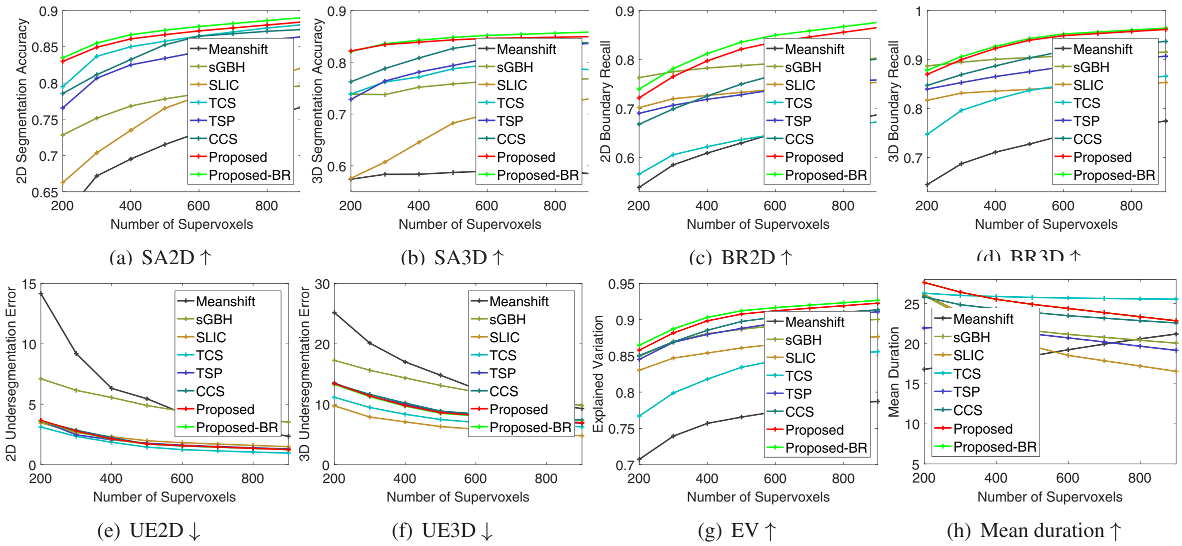
We evaluate the proposed superpixel algorithm on the 200 test images in the BSDS500 dataset [18]. All parameters are fixed in all experiments. We compare the proposed algorithm with eight conventional algorithms: Turbopixels [13], ERS [15], SLIC [1], SEEDS [25], RPS [8], LSC [14], MSLIC [17], and CCS [12].

As in [15], we assess the performance of the superpixel algorithms using three evaluation metrics: undersegmentation error (UE), boundary recall (BR), and achievable segmentation accuracy (ASA). UE is the fraction of pixels leaking across the ground-truth boundaries. BR is the percentage of the ground-truth boundaries recovered by the superpixel boundaries. ASA is the highest accuracy achievable for object segmentation when utilizing generated superpixels as units. A lower UE corresponds to better performance while higher BR and ASA are better. Figure 4 compares the performances. The proposed algorithm outperforms all conventional algorithms in all metrics. Specifically, when the number of segments K is 200, the proposed algorithm yields 6.3% lower UE, 1.0% higher BR, and 0.3% higher ASA than the state-of-the-art algorithm CCS [12].

Figure 5 compares superpixel results qualitatively. We see that the proposed algorithm successfully separates the car and the boy from the background regions, while the conventional algorithms fail to divide them correctly.

5.2. Temporal Superpixel Algorithm

Using the LIBSVX 3.0 benchmark [30], we compare the proposed TS-PPM algorithm with the Meanshift [21], sGBH [31], SLIC [1], TCS [22], TSP [5], and CCS [12] algorithms. We use the evaluation metrics in [30]: 2D segmentation accuracy (SA2D), 3D segmentation accuracy (SA3D), BR2D, BR3D, UE2D, UE3D, explained variation (EV), and mean duration. SA2D, BR2D, and UE2D are measured by calculating ASA, BR, and UE in Section 5.1 for each frame and averaging them. Also, SA3D, BR3D, and UE3D are obtained by regarding a video as a three-dimensional signal and then computing ASA, BR, and UE. EV assesses how well original pixels can be reproduced with temporal superpixels. Mean duration is the average duration of a superpixel in terms of the number of frames.



(a) SA2D ↑ (b) SA3D ↑ (c) BR2D ↑ (d) BR3D ↑

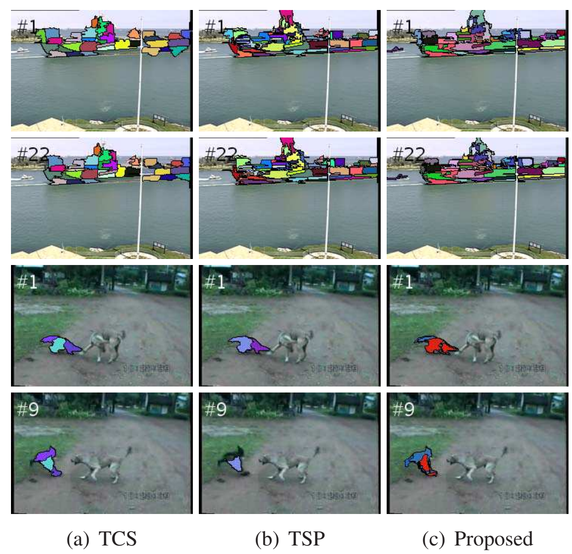
(e) UE2D ↓ (f) UE3D ↓ (g) EV ↑ (h) Mean duration ↑

Figure 6. Quantitative evaluation of temporal superpixel algorithms on the SegTrack dataset [24].

Figure 6 compares the quantitative results on the SegTrack dataset [24]. 'Proposed-BR' represents the results when the optional backward refinement is performed, while 'Proposed' those without the refinement. Even without the refinement, the proposed TS-PPM algorithm provides significantly better SA2D, SA3D, BR2D, BR3D, and EV results than the conventional algorithms and yields comparable UE2D, UE3D, and mean duration results. For example, when the number of superpixels is 400, the SA3D score of the proposed algorithm is 0.839, while those of the state-of-the-art TCS [22], TSP [5], and CCS [12] are 0.772, 0.781, and 0.808, respectively. Furthermore, when the refinement is performed, the segmentation results of the proposed algorithm are further improved, especially in the SA2D, SA3D, BR2D, BR3D, and EV metrics. The improvements are minor in UE2D and UE3D. The mean duration results are unchanged, since the backward refinement does not change the number of superpixels. As mentioned in Section 4.5, the proposed algorithm without the refinement operates online, but the refinement makes it an offline approach. In the remaining experiments, the refinement is not done unless otherwise specified.

We also assess the proposed algorithm on the Chen dataset [6] and provide the results in the supplementary materials. The results exhibit similar tendencies to Figure 6.

To test the efficacy of PPM in the proposed TS-PPM algorithm, we replace it with the state-of-the-art optical flow method, DeepFlow [28]. In the optical flow version, we calculate the motion vector of a superpixel by averaging the pixel-wise flow results within the superpixel. When the number of superpixels is 400, the proposed algorithm provides 6.3%, 8.6%, and 4.6% better UE2D, UE3D, and SA3D scores than the optical flow version, even though DeepFlow requires higher computational complexity. In the other metrics as well, the proposed algorithm yields better performances by about 0.5% on average.



(a) TCS (b) TSP (c) Proposed

Figure 7. Comparison of temporal superpixels. Each frame consists of about 200 superpixels. Colored regions in the first and third rows represent the superpixels containing objects in the first frames. The second and last rows show the superpixels that still contain the objects in the later frames.

Figure 7 compares temporal superpixels of the proposed algorithm with those of TCS and TSP qualitatively. The proposed algorithm segments and tracks the objects more faithfully. For instance, the proposed algorithm successfully tracks both ship and monkey, while TCS fails to track parts of the ship and TSP misses the monkey's upper body.

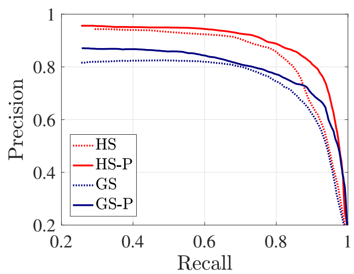


Figure 8. Precision-recall curves of two saliency detection techniques, HS [32] and GS [27]. HS-P and GS-P are the post-processing results of HS and GS, respectively.

5.3. Applications

The proposed superpixel and temporal superpixel algorithms are applicable to many computer vision tasks. In this work, we apply the proposed algorithms to video object segmentation (VOS) and saliency detection.

First, we incorporate the proposed superpixel algorithm in Section 3 into the semi-supervised VOS technique in [10]. We modify the technique by replacing SLIC with the proposed algorithm and then compare the segmentation performances on the SegTrack dataset [24]. The proposed algorithm increases the average intersection over union (IoU) score significantly from 0.532 to 0.579.

Second, we employ the proposed TS-PPM algorithm to post-process VOS results. Given VOS results, we average binary segmentation values of pixels within each temporal superpixel in all frames. Then, we binarize the average values for temporal superpixels via thresholding to obtain binary segmentation results. We post-process segmentation results of the alternate convex optimization (ACO) [11] and fast object segmentation (FOS) [20] techniques on the SegTrack dataset. The average IoU scores are increased from 0.574 to 0.595 for ACO, and from 0.438 to 0.446 for FOS.

Third, we also perform post-processing on saliency detection results. If an image saliency detection technique is applied to each frame in a video independently, the resultant saliency maps are temporally incompatible. Thus, we use the proposed temporal superpixels to obtain temporally compatible maps. We average the saliency values of pixels within each temporal superpixel in all frames and replace the original saliency values with the average value. This post-processing is applied to two saliency detection techniques, hierarchical saliency detection (HS) [32] and geodesic saliency detection (GS) [27]. Figure 8 compares the precision-recall curves on the NTT dataset [2]. The post-processing improves the saliency detection performances of both HS and GS considerably.

More experimental results are available in the supplementary materials.

Table 1. Run-times of the superpixel algorithms.

	[13]	[1]	[15]	[14]	[25]	[17]	[12]	Proposed
Time (s)	8.09	0.26	1.52	0.34	0.06	0.36	0.97	0.20

Table 2. Run-times of the temporal superpixel algorithms (per frame).

	[31]	[1]	[22]	[5]	[12]	Proposed	Proposed-BR
Time (s)	5.71	0.08	7.83	2.39	1.70	0.29	0.38

5.4. Computational Complexity

We measure the run-times of the superpixel and temporal superpixel algorithms using a PC with a 2.2 GHz CPU. We use the available source codes for [1, 5, 12–15, 17, 25, 31], and implement TCS using the parameters in the paper [22]. Table 1 compares the run-times of the superpixel algorithms for dividing a 481×321 image into about 200 superpixels. The proposed algorithm is faster than the conventional algorithms, except for SEEDS [25]. Table 2 compares the run-times of the temporal superpixel algorithms to segment a 240×160 frame into about 200 superpixels. The proposed TS-PPM algorithm without or with the backward refinement is faster than than the conventional algorithms, except for SLIC [1].

6. Conclusions

We proposed the TS-PPM algorithm, which generates temporal superpixels using PPM. For robust motion estimation, PPM estimates the motion vector of a superpixel by combining the patch matching distances of the target superpixel and the neighboring superpixels. The proposed algorithm initializes superpixels in a frame, by mapping the superpixels labels in the previous frames using the PPM motion vectors. Then, it refines the initial superpixels by updating the superpixel labels of boundary pixels. It then performs the superpixel splitting, merging, and relabeling. Optionally, the backward refinement is executed. Experimental results showed that the proposed TS-PPM algorithm outperforms the conventional superpixel algorithms, while demanding low computational complexity, and can be applied to many computer vision tasks.

Acknowledgements

This work was supported partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF2015R1A2A1A10055037), and partly by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2017-2016-0-00464) supervised by the Institute for Information & communications Technology Promotion (IITP).

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, 2012. 1, 2, 6, 8

[2] K. Akamine, K. Fukuchi, A. Kimura, and S. Takagi. Fully automatic extraction of salient objects from videos in near real time. *Comput. J.*, 55(1):3–14, 2012. 8

[3] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patch-match for large displacement optical flow. In *ICCV*, pages 3534–3541, 2014. 4

[4] G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recogn. Lett.*, 15(10):1003–1011, 1994. 2

[5] J. Chang, D. Wei, and J. W. Fisher III. A video representation using temporal superpixels. In *CVPR*, pages 2051–2058, 2013. 1, 2, 3, 6, 7, 8

[6] A. Chen and J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proc. Western New York Image Processing Workshop*, pages 14–17, 2010. 1, 7

[7] G. Egnal and R. P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1127–1133, 2002. 5

[8] H. Fu, X. Cao, D. Tang, Y. Han, and D. Xu. Regularity preserved superpixels and supervoxels. *IEEE Trans. Multimedia*, 16(4):1165–1175, 2014. 1, 2, 6

[9] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patch match for large displacement optical flow. In *CVPR*, pages 5704–5712, 2016. 4

[10] W.-D. Jang and C.-S. Kim. Semi-supervised video object segmentation using multiple random walkers. In *BMVC*, pages 1–13, 2016. 1, 8

[11] 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. 1, 8

[12] S.-H. Lee, W.-D. Jang, and C.-S. Kim. Contour-constrained superpixels for image and video processing. In *CVPR*, 2017. 1, 2, 3, 5, 6, 7, 8

[13] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2290–2297, 2009. 1, 6, 8

[14] Z. Li and J. Chen. Superpixel segmentation using linear spectral clustering. In *CVPR*, pages 1356–1363, 2015. 1, 2, 6, 8

[15] M. Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, pages 2097–2104, 2011. 1, 2, 6, 8

[16] T. Liu, M. Zhang, M. Javanmardi, and N. Ramesh. SSHMT: Semi-supervised hierarchical merge tree for electron microscopy image segmentation. In *ECCV*, pages 144–159, 2016. 1

[17] Y.-J. Liu, C.-C. Yu, M.-J. Yu, and Y. He. Manifold SLIC: A fast method to compute content-sensitive superpixels. In *CVPR*, pages 651–659, 2016. 1, 2, 6, 8

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, 2001. 1, 6

[19] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*, pages 1–8, 2008. 1, 2

[20] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, pages 1777–1784, 2013. 1, 8

[21] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *CVPR*, pages 1–8, 2007. 6

[22] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Temporally consistent superpixels. In *ICCV*, pages 385–392, 2013. 1, 2, 3, 6, 7, 8

[23] Y. Tang and X. Wu. Saliency detection via combining region-level and pixel-level predictions with CNNs. In *ECCV*, pages 809–825, 2016. 1

[24] D. Tsai, M. Flagg, and J. M. Rehg. Motion coherent tracking with multi-label MRF optimization. In *BMVC*, pages 56.1–56.11, 2010. 1, 7, 8

[25] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. In *ECCV*, pages 13–26, 2012. 1, 2, 3, 6, 8

[26] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool. Online video SEEDS for temporal window objectness. In *ICCV*, pages 377–384, 2013. 1, 2

[27] Y. Wei, F. Wen, W. Zhu, and J. Sun. Geodesic saliency using background priors. In *ECCV*, pages 29–42, 2012. 1, 8

[28] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 7

[29] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, 2015. 2, 3

[30] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, pages 1202–1209, 2012. 6

[31] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, pages 626–639, 2012. 6, 8

[32] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *CVPR*, pages 1155–1162, 2013. 1, 8

[33] J. Yao, M. Boben, S. Fidler, and R. Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *CVPR*, pages 2947–2955, 2015. 1, 2, 3

[34] G. Zeng, P. Wang, J. Wang, R. Gan, and H. Zha. Structure-sensitive superpixels via geodesic distance. In *ICCV*, pages 447–454, 2011. 1, 2