Contour-Constrained Superpixels for Image and Video Processing

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 novel contour-constrained superpixel (CCS) algorithm is proposed in this work. We initialize superpixels and regions in a regular grid and then refine the superpixel label of each region hierarchically from block to pixel levels. To make superpixel boundaries compatible with object contours, we propose the notion of contour pattern matching and formulate an objective function including the contour constraint. Furthermore, we extend the CCS algorithm to generate temporal superpixels for video processing. We initialize superpixel labels in each frame by transferring those in the previous frame and refine the labels to make superpixels temporally consistent as well as compatible with object contours. Experimental results demonstrate that the proposed algorithm provides better performance than the state-of-the-art superpixel methods.

1. Introduction

Superpixel segmentation is a preprocessing task to partition an input image into smaller meaningful regions. In comparison with the pixel representation of an image, the superpixel representation can reduce the number of image primitives or units greatly. Recently, superpixel methods have been widely used in many computer vision algorithms, including image segmentation [15], video object segmentation [9], semantic segmentation [12], saliency detection [21], and stereo matching [31].

Many superpixel methods have been proposed [1, 8, 10, 11, 13, 16, 18, 24, 34, 35], which achieve superpixel partitioning by optimizing objective functions in general. Since each superpixel is employed as a minimal unit in applications, it should belong to a single object without overlapping with multiple objects. In other words, superpixels should adhere to image contours. Thus, a few superpixel methods [8, 18, 35] use contour information in their objective functions. On the other hand, several advanced contour detection techniques, based on deep learning, have been proposed recently [28, 33], which can detect object contours faithfully with relatively low complexity. In this work, we attempt to exploit learning-based contour information explicitly to achieve accurate superpixel segmentation.

Also, temporal superpixel (or supervoxel) methods for video processing have been proposed [1, 5, 20, 25]. If a superpixel method is applied to each frame in a video sequence independently, it will lead to flickering artifacts. Therefore, a temporal superpixel method should consider temporal correlation to label the same regions consistently in consecutive frames while making superpixel boundaries compatible with object contours.

We propose a novel superpixel algorithm, referred to as contour-constrained superpixel (CCS). We initialize superpixels and regions in a regular grid and then refine the superpixel label of each region hierarchically from block to pixel levels. At each level, we use a cost function to explicitly enforce the contour constraint that two neighboring regions should belong to different superpixels if there is an object contour between them. To this end, we propose the notion of contour pattern matching. Moreover, we extend the proposed CCS algorithm to generate temporal superpixels. We initialize superpixel labels in each frame by transferring those in the previous frame using optical flows. Then, we perform the temporal superpixel labeling to make superpixels temporally consistent, as well as compatible with object contours. Experimental results show that the proposed algorithm outperforms the conventional superpixel [1, 8, 10, 11, 13, 16, 24, 34] and temporal superpixel [1, 5, 20, 30] methods and can be applied to object segmentation [9] and saliency detection [14, 32] effectively. To summarize, this paper has three main contributions.

- Introduction of the contour constraint to compel superpixel boundaries to be compatible with object contours, by adopting the contour pattern matching.
- Extension of the proposed CCS algorithm for video processing, which yields temporally consistent and spatially accurate superpixels.
- Remarkable performance achievement on superpixel and temporal superpixel datasets and improvement of many computer vision algorithms by applying the proposed CCS.

2. Related Work

2.1. Superpixel Methods

A variety of superpixel methods have been proposed. Levinshtein *et al.* [10] proposed Turbopixels. They initialized seeds and propagated them using the level set method to obtain superpixels.

Achanta *et al.* [1] proposed the simple linear iterative clustering (SLIC), which is a K-means optimization method. SLIC represents each pixel with a 5-dimensional feature vector, composed of spatial coordinates and colors. It assigns each pixel to the nearest cluster and updates the cluster centers iteratively. Li and Chen [11] proposed the linear spectral clustering (LSC), based on a weighted K-means scheme. However, these K-means-based methods may not preserve the connectedness of each superpixel, and thus they should perform postprocessing. Liu *et al.* [16] extended SLIC to yield small superpixels in detailed regions and large superpixels in flat regions.

Liu *et al.* [13] proposed an entropy-based superpixel method. Their method constructs a graph on an input image and formulates a cost function, which consists of the entropy rate of a random walker on the graph and a balancing term. The entropy rate enforces each superpixel to be compact and homogeneous, while the balancing term constrains the size of each superpixel to be similar.

Also, coarse-to-fine methods have been proposed. Van den Bergh *et al.* [24] proposed the superpixels extracted via energy-driven sampling (SEEDS) method, which changes the superpixel label of each region to refine superpixel boundaries in a coarse-to-fine manner. The superpixel labels are updated to improve the homogeneity of colors within each superpixel. However, SEEDS may fail to obtain compact superpixels. Thus, Yao *et al.* [34] proposed another coarse-to-fine method. They defined the cost function based on the distances from the centers of superpixels to achieve compactness.

However, the aforementioned methods [1, 10, 11, 13, 16, 24, 34] do not exploit contour information, and thus their superpixel boundaries may be incompatible with image contours. Only a few contour-based superpixel methods have been proposed [8, 18, 35]. Moore *et al.* [18] and Fu *et al.* [8] determined superpixel boundaries, by finding paths containing many image contour pixels. However, both methods should maintain a regular grid structure of superpixels, which limits their clustering performance. Zeng *et al.* [35] proposed a superpixel method using geodesic distances. Their algorithm assigns each pixel to the seed that has the smallest geodesic distance and updates the position of each seed alternately. For the geodesic distance computation, gradient magnitudes are utilized. However, note that the gradient information is not sufficient for detecting true image contours.

2.2. Temporal Superpixel Methods

For video processing, temporal superpixel methods have been proposed. Achanta *et al.* [1] and Van den Bergh *et al.* [25], respectively, modified superpixel methods to process video sequences. Achanta *et al.* [1] extended their SLIC algorithm for 2D images straightforwardly to obtain temporal superpixels, by considering a video sequence as the 3D signal. Van den Bergh *et al.* [25] extended SEEDS [24], by considering previous frames when constructing color histograms. They also created and terminated labels to reflect color changes in different frames.

Reso *et al.* [20] proposed temporally consistent superpixels (TCS). They labeled each superpixel using the K-means optimization as in [1]. However, they adopted a temporal sliding window to improve temporal consistency. Specifically, to calculate the average color of a superpixel, they considered not only the pixels in the current frame but also those in the other frames in the sliding window. Chang *et al.* [5] proposed another temporal superpixel method, called TSP, which is allowed to change the superpixel label of each pixel only if the topological relationship of superpixels is maintained. Both TCS and TSP use optical flow information from the previous frame to initialize the partitioning of a current frame to achieve temporal consistency.

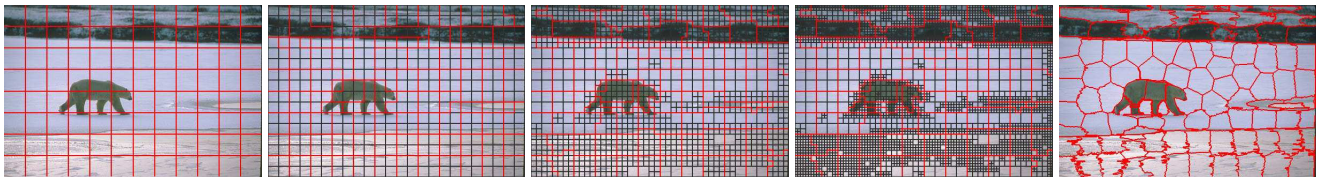
3. Contour-Constrained Superpixels

This section proposes a novel superpixel algorithm, referred to as CCS. We first initialize K superpixels $S_1, \ldots, S_K$ in a regular grid, as shown in Figure 1(a). Then, we refine those superpixels hierarchically. Specifically, we divide regions and update their superpixel labels hierarchically at three block levels in Figures 1(b)∼(d) and perform the finest update at the pixel level in Figure 1(e).

Let $l(R_i) \in \{1, \ldots, K\}$ denote the superpixel label of the ith region R_i, which can be either a block or a pixel according to the refinement level. Note that R_i constitutes the $l(R_i)$-th superpixel, and thus $R_i \subset S_{l(R_i)}$. At each refinement level, we iteratively update the superpixel label of a boundary region R_i from $l(R_i)$ to $l(R_j)$ of a neighboring region $R_j \in \mathcal{N}_{R_i}$, which has the smallest cost $E(i,j)$. Here, $\mathcal{N}_{R_i}$ denotes the set of neighboring regions of R_i, which are adjacent to R_i. We update the superpixel label of the boundary region R_i, only if R_i is a simple point [3], to preserve the topological relationship among superpixels, as done in [24, 34]. We formulate the cost function $E(i,j)$, for updating the superpixel label of R_i from $l(R_i)$ to $l(R_j)$, as

$$E(i,j) = [E_{\mathrm{D}}(i,j) + \gamma E_{\mathrm{L}}(i,j) + \eta E_{\mathrm{I}}(i,j)] \times E_{\mathrm{C}}(i,j) \tag{1}$$

where parameters γ and η control the relative contributions of the feature distance E_{D}, the boundary length cost E_{L}, and the inter-region color cost E_{I}. In all experiments, $\gamma = 2$ and $\eta = 10$. We amplify the cost in (1) when there is a



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

Figure 1. Hierarchical superpixel labeling ($K = 96$). Red lines depict superpixel boundaries, while black lines in (b)∼(d) are block boundaries. (b)∼(d) hierarchical superpixel labeling at three block levels and (e) the finest pixel level labeling.

contour between R_i and R_j, by adopting the contour constraint $E_{\mathrm{C}}(i,j)$. In other words, we constrain superpixels to be compatible with image contours.

Let us describe each term in the cost function $E(i,j)$ in (1) subsequently in more detail.

3.1. Feature Distance from Superpixel Centroid

We use the feature distance between a boundary region R_i and the $l(R_j)$-th superpixel $S_{l(R_j)}$, which a neighboring region R_j constitutes. We adopt color and position as features and define the feature distance $E_{\mathrm{D}}(i,j)$ as

$$\begin{aligned} E_{\mathrm{D}}(i,j) &= \left\|\mathbf{c}(R_i) - \mathbf{c}(S_{l(R_j)})\right\|^2 \\ &\quad + \left\|\mathbf{p}(R_i) - \mathbf{p}(S_{l(R_j)})\right\|^2 \end{aligned} \tag{2}$$

where $\mathbf{c}(R_i)$ and $\mathbf{c}(S_{l(R_j)})$ denote the average CIELAB colors of region R_i and superpixel $S_{l(R_j)}$, respectively. Similarly, $\mathbf{p}(R_i)$ and $\mathbf{p}(S_{l(R_j)})$ are the average positions of R_i and $S_{l(R_j)}$. The color distance makes that a superpixel consists of homogenous colors, while the spatial distance imposes that a superpixel is composed of nearby pixels.

3.2. Boundary Length Cost

To yield superpixels of compact shapes, we minimize the boundary lengths of superpixels explicitly. To this end, we define the boundary length cost $E_{\mathrm{L}}(i,j)$, by counting the changed number of boundary regions when the superpixel label of region R_i is updated from $l(R_i)$ to $l(R_j)$, *i.e.*,

$$E_{\mathrm{L}}(i,j) = \lambda(R_i, l(R_j)) - \lambda(R_i, l(R_i)) \tag{3}$$

where $\lambda(R_i,k)$ denotes the total number of boundary regions in the image when the superpixel label of R_i is k. In the implementation, we only consider the set of neighboring regions $\mathcal{N}_{R_i}$ of R_i, since the states (boundary or not) of the other regions are not affected by the superpixel label of R_i. If $E_{\mathrm{L}}(i,j)$ is positive, the label change of R_i from $l(R_i)$ to $l(R_j)$ increases the total boundary length. Thus, by minimizing $E_{\mathrm{L}}(i,j)$, we constrain each superpixel to have a small boundary length and thus have a compact shape.

3.3. Inter-Region Color Cost

We assign each region to a superpixel by considering the color difference between the region and its neighboring regions. It is more likely that an object boundary exists between the two regions when they have different colors. Therefore, we attempt to assign different superpixel labels to neighboring regions with dissimilar color information. Moreover, we adopt the notion of the internal difference [7] to consider the texture information in each superpixel. Specifically, we define the internal difference $\kappa(S_{l(R_i)})$ of superpixel $S_{l(R_i)}$ as the maximum color difference between neighboring regions within $S_{l(R_i)}$,

$$\kappa(S_{l(R_i)}) = \max_{\substack{R_m, R_n \in S_{l(R_i)}, \\ R_n \in \mathcal{N}_{R_m}}} \left\|\mathbf{c}(R_m) - \mathbf{c}(R_n)\right\|^2. \tag{4}$$

A large $\kappa(S_{l(R_i)})$ indicates that $S_{l(R_i)}$ has complex texture in general.

Then, we compute the inter-region color cost $E_{\mathrm{I}}(i,j)$, by comparing the color distance between neighboring regions R_i and R_j and the internal difference $\kappa(S_{l(R_j)})$,

$$E_{\mathrm{I}}(i,j) = \max\left\{0, \left\|\mathbf{c}(R_i) - \mathbf{c}(R_j)\right\|^2 - \kappa(S_{l(R_j)})\right\}. \tag{5}$$

We hence impose the inter-region color cost only when the color difference between R_i and R_j is larger than $\kappa(S_{l(R_j)})$. Hence, superpixel $S_{l(R_j)}$ can include a new region R_i with high tolerance of color difference, if it has complex texture. In contrast, a superpixel with flat texture cannot include a new region with a large color difference.

3.4. Contour Constraint

Superpixels should adhere to object contours since they are mainly used as processing units to detect and segment objects. In other words, object contours should be composed of superpixel boundaries, although superpixel boundaries are not necessarily object contours. In this work, we adopt the contour constraint E_{C} in (1) to form superpixels so that their boundaries are compatible with object contours.

Given an input image, we obtain its contour map by employing the holistically-nested edge detection (HED) scheme [28]. HED can extract faithful contours with relatively low computational complexity, by adopting a convolutional neural network. However, it is hard to determine the existence of a contour between distant pixels based on the primitive contour map. Thus, we match each patch in

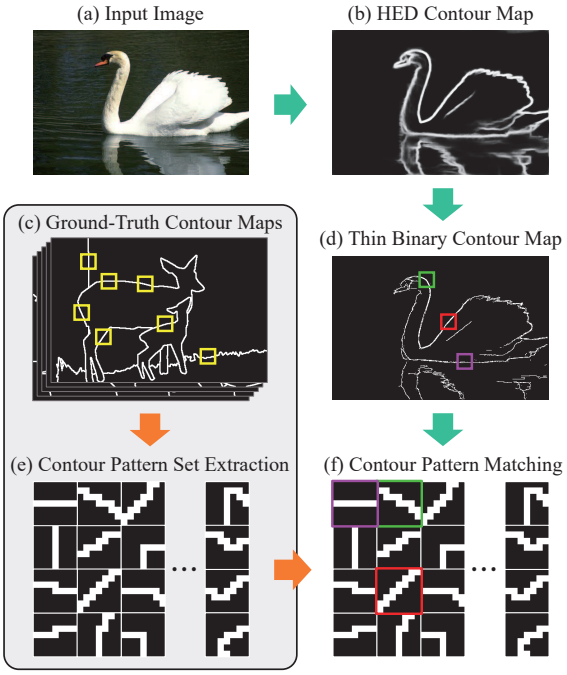


Figure 2. Contour pattern set extraction and contour pattern matching. In the contour pattern set, the patterns are ordered by occurring frequencies from the left to the right.

the contour map to a pre-extracted contour pattern. By examining these matching patterns, we can estimate the probability that two pixels are separated by a contour.

Figure 2 shows the processes of contour pattern set extraction and contour pattern matching, which are used to define the contour constraint. To construct a set of contour patterns, we use 200 training images in the BSDS500 dataset [17], in which each training image has ground-truth contour maps, as shown in Figure 2(c). We use the 7×7 patch, centered at each contour pixel, as a contour pattern. Since each patch is binary and the center pixel value is 1, there can be $2^{7\times7-1}$ contour patterns. However, we only consider the patterns, whose elements are divided into two mutually exclusive regions by the contours. Also, we construct the contour pattern set by selecting only top 1,000 frequently occurring patterns in the contour maps as in Figure 2(e). These 1,000 patterns cover 90.5% of the patches in the training contour maps.

On the HED contour map in Figure 2(b), we perform the non-maximum suppression [4] and then thresholding to yield a thin binary contour map, as shown in Figure 2(d). Then, we employ the pattern matching process in Figure 2(f). We consider the 7×7 patch P_m, centered at a contour pixel m in an input image, as shown in Figure 3. Let $\mathcal{M}$ denote the contour pixel set. For each patch P_m, $m \in \mathcal{M}$, we compute its Hamming distances from the contour patterns and select the best matching pattern Q_m with the shortest distance. Then, the contour probability $\phi(u, v)$

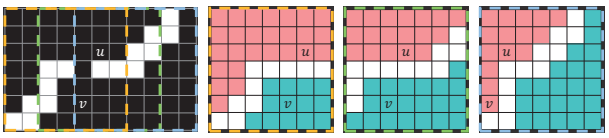


Figure 3. An example of the contour pattern matching: (a) input contour map, including three patches depicted in orange, green blue dashed boxes, and (b)∼(d) the matching contour patterns for the three patches.

between pixels u and v is modeled as

$$\phi(u, v) = \frac{\sum_{m\in\mathcal{M}} \delta_m^{(1)}(u, v) \times \delta_m^{(2)}(u, v)}{\sum_{m\in\mathcal{M}} \delta_m^{(1)}(u, v)} \tag{6}$$

where the binary function $\delta_m^{(1)}(u, v) = 1$ only if both u and v are within patch P_m as in Figures 3(b)∼(d). So the denominator in (6) counts the number of patches that include both u and v. Also, the binary function $\delta_m^{(2)}(u, v) = 1$ only if u and v belong to different components in the matching pattern Q_m as in Figures 3(b) and (c). Hence, $\phi(u, v)$ measures the proportion of patches whose matching patterns separate u from v. In the case of Figure 3, out of the three matching patterns, two separate u and v successfully. Thus, by considering all the patches containing both pixels u and v, we can obtain the contour probability $\phi(u, v)$ faithfully, although the contour is not closed in Figure 3(a).

Then, we determine the contour probability $\psi(R_i, R_j)$ between regions R_i and R_j, by finding the maximum contour probability between the pixels in R_i and R_j,

$$\psi(R_i, R_j) = \max_{u\in R_i, v\in R_j} \phi(u, v). \tag{7}$$

We then compute the contour constraint $E_{\mathrm{C}}(i, j)$ by

$$E_{\mathrm{C}}(i, j) = \exp\left(\beta \times \psi(R_i, R_j)\right) \tag{8}$$

where $\beta = 3$. The exponential function is used in (8) so as to amplify the cost function $E(i, j)$ in (1) significantly when there is an object contour between regions R_i and R_j. By adopting the contour constraint in (1), we can make superpixels compatible with image contours.

3.5. Hierarchical Superpixel Refinement

As shown in Figure 1, we refine superpixels hierarchically from block to pixel levels. At the coarsest level (level 1) in Figure 1(b), each initial rectangular superpixel is divided into four blocks, and then each block is regarded as a refinement unit, *i.e.* region R_i. We update the superpixel label of each region to minimize the cost $E(i, j)$ in (1) iteratively. Then, we use the divisive algorithm [22] to determine the block structure at the finer level (level 2) in Figure 1(c).

Algorithm 1 Contour-Constrained Superpixels

Initialize superpixels and regions in a regular grid
for level $=1$ to 4 **do**
  repeat for all boundary simple regions R_i
    $j^* \leftarrow \arg\min_j E(i,j)$ and $l(R_i) \leftarrow l(R_{j^*})$ ▷ (1)
    Update the average colors and positions of superpixels
    Update the internal differences of superpixels
  until convergence or pre-defined number of iterations
  if level $=1$ or 2 **then**
    Divide regions into blocks ▷ (9)
  else if level $=3$ **then**
    Divide all regions into pixels
  end if
end for

To decide whether to divide region R_i or not, we compute its inhomogeneity

$$\theta(R_i) = \max_{u,v\in R_i, v\in\mathcal{N}_u} \|\mathbf{c}(u)-\mathbf{c}(v)\|^2 + \exp(\beta \times \max_{u,v\in R_i, v\in\mathcal{N}_u} \phi(u,v)), \quad (9)$$

where $\mathcal{N}_u$ is the set of 4-adjacent pixels to pixel u, and $\mathbf{c}(u)$ is the CIELAB color of u. In (9), the first term measures the maximum color difference between adjacent pixels in R_i, and the second term computes the maximum contour strength between adjacent pixels in R_i similarly to (8). When $\theta(R_i)$ is higher than a threshold $\tau_{\text{div}} = 100$, we divide R_i into four blocks for level 2.

This division process is repeated once more to refine superpixels at level 3. Notice that unlike the conventional coarse-to-fine methods [24, 34], the proposed hierarchical refinement divides only inhomogeneous regions containing complicated texture and contours. Thus, homogeneous regions are not divided, and the corresponding superpixels can maintain relatively regular and compact shapes.

Finally, we conduct the superpixel labeling at the pixel level (level 4). At level 4, contrary to levels 1∼3, we divide all blocks into pixels to perform the finest scale superpixel labeling. **Algorithm 1** summarizes the proposed CCS algorithm. The iteration terminates when there is no label change or the maximum number of iterations are performed. We set the maximum number of iterations to 20.

4. Contour-Constrained Temporal Superpixels

We extend the proposed CCS algorithm to generate temporal superpixels for video processing.

4.1. Initialization

We perform the CCS algorithm to obtain the superpixel labels of the first frame $I^{(1)}$ in a video sequence. Then, for each frame $I^{(t)}$, $t \geq 2$, we estimate optical flows [26] from $I^{(t-1)}$ to $I^{(t)}$. We transfer the label of each superpixel in $I^{(t-1)}$ to $I^{(t)}$ by employing the average optical flow of the superpixel. By initializing the labels with the optical flow information, we can label the same regions in consecutive frames consistently. During the initialization, we do not assign any superpixel labels to occluded or disoccluded pixels. Note that we refer to a pixel mapped from multiple superpixels in the previous frame as an occluded pixel, and a pixel mapped from no superpixel as a disoccluded pixel.

Algorithm 2 Contour-Constrained Temporal Superpixels

Apply **Algorithm 1** to $I^{(1)}$
for $t=2$ to t_{end} **do**
  Initialize superpixels using the results in $I^{(t-1)}$
  repeat for all boundary simple pixels $R_i^{(t)}$
    $j^* = \arg\min_j E(i,j,t)$ ▷ (10)
    $l(R_i^{(t)}) \leftarrow l(R_{j^*}^{(t)})$
    Update the average positions of superpixels
    Update the internal differences of superpixels
  until convergence or pre-defined number of iterations
  Perform superpixel merging, splitting, and relabeling
end for

4.2. Temporal Superpixel Labeling

After the initialization, we perform the temporal superpixel labeling in a similar manner to Section 3. However, the temporal superpixel labeling is performed at the pixel level only. Thus, the cost function $E(i,j,t)$ for updating the superpixel label of a boundary pixel $R_i^{(t)}$ from $l(R_i^{(t)})$ to $l(R_j^{(t)})$ in frame $I^{(t)}$ is defined as

$$E(i,j,t) = [E_{\text{D}}(i,j,t) + \gamma E_{\text{L}}(i,j,t) + \eta E_{\text{I}}(i,j,t)] \times E_{\text{T}}(i,j,t) \quad (10)$$

where $E_{\text{T}}(i,j,t)$ is the temporal contour constraint. Note that the feature distance $E_{\text{D}}(i,j,t)$, the boundary length cost $E_{\text{L}}(i,j,t)$, and the inter-region color cost $E_{\text{I}}(i,j,t)$ are defined in the same way as (2), (3), and (5), respectively.

We adopt the temporal contour constraint $E_{\text{T}}(i,j,t)$ in (10) to make superpixels temporally consistent and also compatible with image contours. It is formulated as

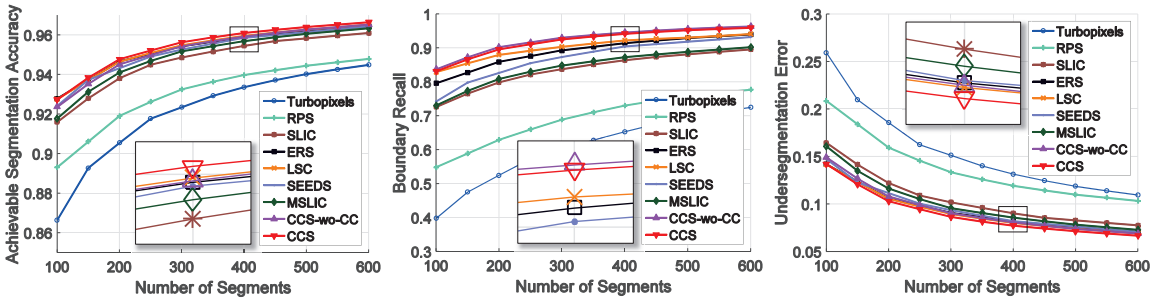
$$E_{\text{T}}(i,j,t) = E_{\text{C}}(i,j,t) \times \rho(i,j,t) \quad (11)$$

where $E_{\text{C}}(i,j,t)$ is computed in the same way as (8), *i.e.*

$$E_{\text{C}}(i,j,t) = \exp(\beta \times \psi(R_i^{(t)}, R_j^{(t)})). \quad (12)$$

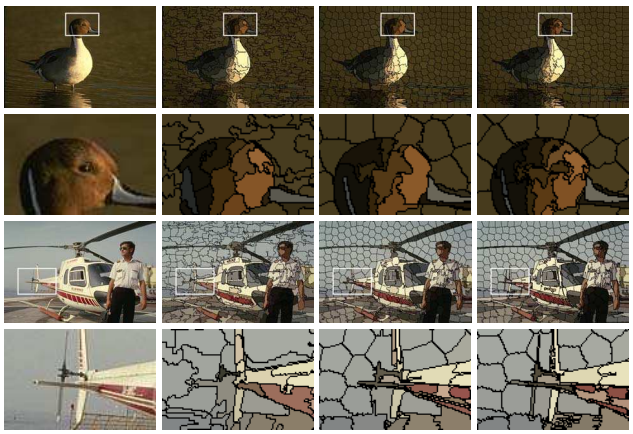
Also, $\rho(i,j,t)$ is a relaxation factor that diminishes the contour constraint adaptively to improve the temporal consistency of superpixel labels, which is defined as

$$\rho(i,j,t) = \begin{cases} \frac{1}{1+\exp\left(-\zeta \times h(R_i^{(t)})\right)} & \text{if } l(R_j^{(t)}) \in \mathcal{L}_i^{(t)}, \\ 1 & \text{otherwise,} \end{cases} \quad (13)$$



(a) ASA ↑ (b) BR ↑ (c) UE ↓

Figure 4. Quantitative evaluation of superpixel algorithms. The horizontal axis represents the number of segments (or superpixels) in each image. CCS and CCS-wo-CC denote the proposed CCS algorithm with and without the contour constraint, respectively.



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

Figure 5. Visual comparison of superpixel results. Each image consists of about 400 superpixels. The second and last row show the magnified parts of the images in the first and third rows, respectively. In (b)∼(d), each superpixel is represented by the average color.

where $\zeta = 2$, $\mathcal{L}_i^{(t)}$ is the set of superpixel labels that are mapped to $R_i^{(t)}$ from $I^{(t-1)}$, and $h(R_i^{(t)})$ is the HED contour response [28] for $R_i^{(t)}$. Thus, if the neighboring label $l(R_j^{(t)})$ belongs to $\mathcal{L}_i^{(t)}$, we relax the contour constraint in (11). However, the relaxation factor gets closer to 1 if the contour response $h(R_i^{(t)})$ increases.

4.3. Merging, Splitting, and Relabeling

As the superpixel labeling is performed frame by frame, a superpixel can grow or shrink. To prevent irregular superpixel sizes, we carry out superpixel merging and splitting. Also, superpixels can be labeled incorrectly because of occlusion or illumination variation. To avoid this mislabeling, we perform relabeling as postprocessing.

Let $\bar{A} = N/K$ denote the average superpixel size, where N is the number of pixels in a frame. By comparing the size $A_k^{(t)}$ of each superpixel S_k at frame $I^{(t)}$ with $\bar{A}$, we decide whether to merge or split S_k. When $A_k^{(t)}/\bar{A}$ is larger than τ_{spl}, we divide superpixel S_k in $I^{(t)}$ across the main axis, corresponding to the biggest eigenvector of the spatial distribution, as done in [35]. Also, when $A_k^{(t)}/\bar{A}$ is smaller than τ_{mer}, we merge S_k with the nearest superpixel. We find the nearest superpixel by comparing the centroid of S_k with those of adjacent superpixels. We set $\tau_{\text{spl}} = 3$ and $\tau_{\text{mer}} = 1/16$ in all experiments.

We also perform superpixel relabeling, by measuring color consistency. We define the color consistency C_k of superpixel S_k, by comparing the average color $\mathbf{c}_{1:t-1}(S_k)$ of S_k from frame $I^{(1)}$ to $I^{(t-1)}$ and the average color $\mathbf{c}_t(S_k)$ of superpixel S_k in frame $I^{(t)}$,

$$C_k = \|\mathbf{c}_{1:t-1}(S_k) - \mathbf{c}_t(S_k)\|^2 . \tag{14}$$

If C_k is larger than a threshold $\tau_{\text{rel}} = 120$, we relabel S_k with a new label. **Algorithm 2** summarizes the proposed temporal superpixel algorithm.

Table 1. Run-times of the superpixel algorithms.

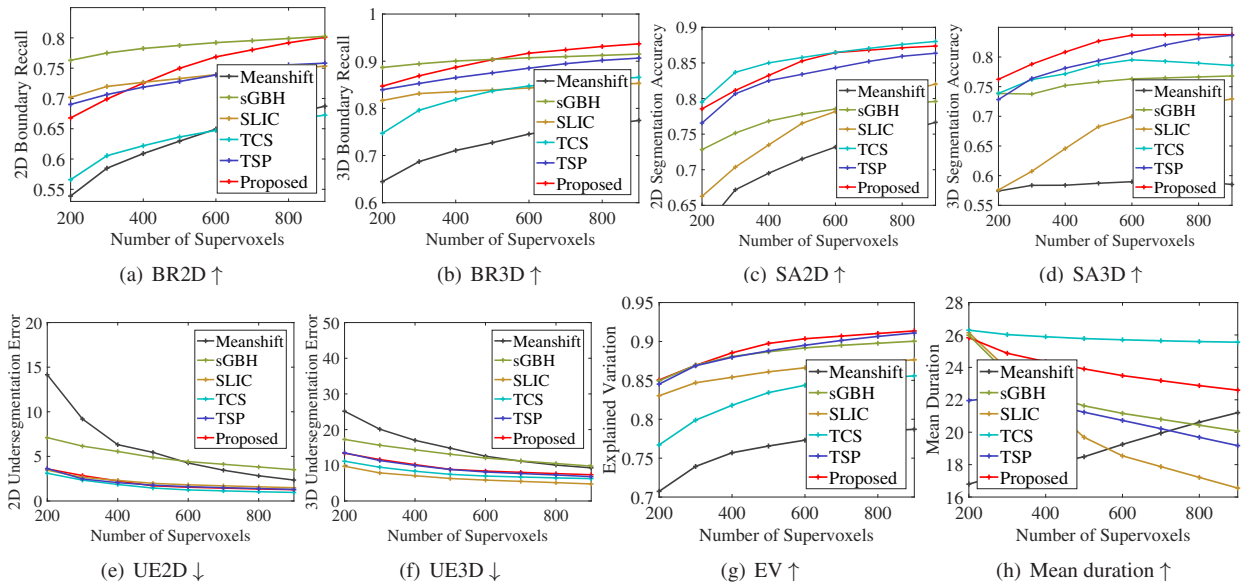
	[10]	[8]	[1]	[13]	[11]	[24]	[16]	Proposed
Time (s)	8.09	12.78	0.26	1.52	0.34	0.06	0.36	0.97

5. Experimental Results

5.1. Superpixel Algorithm

We assess the proposed CCS algorithm on the 200 test images in the BSDS500 dataset [17]. All parameters are fixed in all experiments. We compare the proposed algorithm with seven conventional superpixel algorithms: Turbopixels [10], regularity preserved superpixels (RPS) [8], SLIC [1], entropy rate superpixel segmentation (ERS) [13], LSC [11], SEEDS [24], and manifold SLIC (MSLIC) [16].

We quantify the superpixel partitioning performance using three evaluation metrics, as in [13]: achievable segmentation accuracy (ASA), boundary recall (BR), and undersegmentation error (UE). ASA is the highest achievable object segmentation accuracy when the resultant superpixels are employed as units. BR is the proportion of the ground-truth boundaries that match the superpixel boundaries. UE measures the proportion of the pixels that leak across the



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

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

Figure 6. Quantitative evaluation of temporal superpixel algorithms on the SegTrack dataset [23].

ground-truth boundaries. Note that higher ASA and BR corresponds to better performance, while a lower UE is better. Figure 4 compares the algorithms, in which CCS and CCS-wo-CC denote the proposed CCS algorithm with and without the contour constraint, respectively. We see that CCS performs better than CCS-wo-CC in terms of ASA and UE while providing comparable BR performance. This indicates that the contour constraint plays an essential role in the proposed algorithm. Moreover, notice that the proposed CCS outperforms all the conventional algorithms by considerable margins in terms of all three metrics. Especially, the proposed algorithm yields 2.1% higher BR and 3.7% lower UE values than LSC, which is the state-of-the-art conventional algorithm, when the number of segments K is 400.

Figure 5 compares superpixel results qualitatively. We see that the proposed algorithm successfully separates the objects from the background regions, even though the objects and the background regions have similar colors. Especially, the proposed algorithm successfully delineates the duck head and the tail rotor of the helicopter, whereas the three conventional algorithms fail.

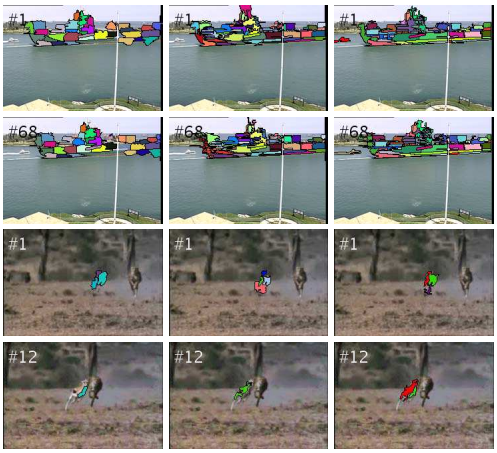
We have measured the run-times of the proposed algorithms using a PC with a 2.2 GHz CPU. Table 1 compares the run-times of the superpixel algorithms for dividing a 481×321 image into about 200 superpixels. The run-time of the proposed algorithm is comparable to those of the conventional algorithms.

5.2. Temporal Superpixel Algorithm

Next, we evaluate the proposed temporal superpixel algorithm using the LIBSVX 3.0 benchmark [29]. Five conventional algorithms are compared: Meanshift [19], streaming hierarchical video segmentation (sGBH) [30], SLIC [1], TCS [20], and TSP [5]. Note that Meanshift, sGBH, and SLIC are video segmentation algorithms without topology constraint, rather than temporal superpixel algorithms. Thus, they generate segments that have multiple connected components or are shaped irregularly. Therefore, we compare the proposed algorithm mainly with TCS and TSP. We use the eight evaluation metrics in [29]: 2D boundary recall (BR2D), 3D boundary recall (BR3D), 2D segmentation accuracy (SA2D), 3D segmentation accuracy (SA3D), 2D undersegmentation error (UE2D), 3D undersegmentation error (UE3D), explained variation (EV), and mean duration. BR2D, SA2D, and UE2D are obtained by calculating BR, ASA, and UE for each frame and averaging them over all frames. BR3D, SA3D, and UE3D are obtained by considering a video sequence as a 3D volume and then computing BR, ASA, and UE. Also, EV quantifies how well the original information can be represented with the average colors of superpixels, and mean duration measures how long superpixels last in terms of the number of frames.

Figure 6 compares the quantitative results on the SegTrack dataset [23]. The proposed algorithm yields the highest SA3D and EV curves. Moreover, although sGBH has no topology and regularity constraints and thus has advantages when calculating BR2D and BR3D, the proposed algorithm provides comparable BR2D and BR3D results. Also, when we compare with the temporal superpixel algorithms TCS and TSP only, the proposed CCS provides the best BR2D, BR3D, SA3D, and EV values, while providing comparable results in terms of the other metrics. The results on the Chen dataset [6] are available in the supplemental materials, which show similar tendencies to Figure 6.

Figure 7 shows temporal superpixel results. We see that the proposed algorithm detects and tracks objects faithfully.



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

Figure 7. Comparison of temporal superpixels on the 'Container' and 'Cheetah' sequences. Each frame consists of about 200 superpixels. Regions surrounded by black boundaries in the first and third rows depict the labels of superpixels containing objects in the first frames. The second and last rows show the superpixels that still contain the objects in the later frames.

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

	[30]	[1]	[20]	[5]	Proposed
Time (s)	5.71	0.08	7.83	2.39	1.70

For instance, the proposed algorithm successfully delineates the small boat in the 'Container' sequence, while TCS and TSP yield superpixels whose boundaries do not match the contour of the boat. Also, notice that, as time goes on, the proposed algorithm maintains the superpixel labels of objects more effectively than TCS and TSP do.

Table 2 lists the run-times of the temporal superpixel algorithms to segment a 240×160 frame into about 200 superpixels. The proposed algorithm is faster than the conventional algorithms, except for SLIC [1].

5.3. Applications

The proposed CCS algorithm can be applied to various image and video processing tasks. We demonstrate the efficacy of the proposed algorithm on two exemplar tasks.

First, we improve the video object segmentation technique based on multiple random walkers (MRW) [9]. We modify it to use the proposed CCS, instead of SLIC. Then, we compare the two segmentation techniques, *i.e.* MRW-SLIC and MRW-CCS, on the SegTrack dataset [23]. Each segmentation technique uses about 400 superpixels per frame. We measure the intersection over union (IoU) scores [27]. The overall IoU score is increased from 0.532 to 0.571 by replacing SLIC with CCS.

Second, we use contour-constrained temporal superpixels to postprocess video saliency detection results. If we apply an image saliency detection technique to each frame in a video sequence independently, the resultant saliency maps may be temporally inconsistent. Therefore, we use the proposed contour-constrained temporal superpixels for the postprocessing. Specifically, we average the saliency values of the pixels in all frames, constituting each superpixel, and then replace those saliency values with the average value. This simple processing improves the saliency detection performance, as shown by the precision-recall curves in Figure 8. We test two saliency detection techniques, hierarchical saliency detection (HS) [32] and deep hierarchical saliency network (DHSNet) [14] on the NTT dataset [2]. HS-P and DHSNet-P denote the postprocessing results of HS and DHSNet. The postprocessing improves the performance of HS significantly. Furthermore, although the amount of the improvement is relatively small, the postprocessing is still effective for the state-of-the-art saliency technique DHSNet. The precision of the original DHSNet saturates at 0.981, while that of DHSNet-P at 0.994.

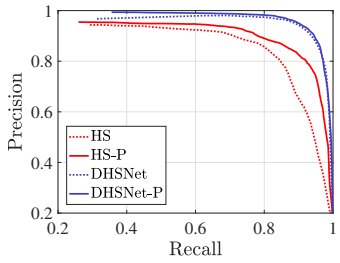


Figure 8. Precision-recall curves of saliency detection techniques.

6. Conclusions

We proposed the CCS algorithm. We initialized superpixels in a regular grid and performed the hierarchical refinement from block to pixel levels. We adopted the contour constraint to make superpixels adhere to object contours. We also extended to the CCS algorithm for video processing. We transferred superpixel labels using optical flows and performed the temporal superpixel labeling to yield temporally consistent superpixels. Experimental results showed that the proposed algorithm outperforms the state-of-the-art superpixel methods and can be applied to object segmentation and saliency detection effectively.

Acknowledgements

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

References

[1] 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, 7, 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] G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recogn. Lett.*, 15(10):1003–1011, 1994. 2

[4] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. 4

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

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

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vis.*, 59(2):167–181, 2004. 3

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

[10] 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, 2, 6

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

[12] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan. Semantic object parsing with graph LSTM. In *ECCV*, pages 125–143, 2016. 1

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

[14] N. Liu and J. Han. DHSNet: Deep hierarchical saliency network for salient object detection. In *CVPR*, pages 678–686, 2016. 1, 8

[15] 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

[16] 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

[17] 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. 4, 6

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

[19] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *CVPR*, pages 1–8, 2007. 7

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

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

[22] S. Theodoridis and K. Koutroumbas. Pattern recognition, fourth edition. chapter 13, pages 653–700. Academic Press, 2008. 4

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

[24] 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, 5, 6

[25] 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

[26] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 5

[27] L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. JOTS: Joint online tracking and segmentation. In *CVPR*, pages 2226–2234, 2015. 8

[28] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, 2015. 1, 3, 6

[29] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, pages 1202–1209, 2012. 7

[30] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, pages 626–639, 2012. 1, 7, 8

[31] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun. Continuous markov random fields for robust stereo estimation. In *ECCV*, pages 45–58, 2012. 1

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

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

[34] 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, 5

[35] 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, 6