Superpixel Hierarchy

Xing Wei, Qingxiong Yang, *Member, IEEE*, Yihong Gong, *Fellow, IEEE*, Narendra Ahuja, *Fellow, IEEE*, and Ming-Hsuan Yang, *Senior Member, IEEE*

***Abstract*— Superpixel segmentation has been one of the most important tasks in computer vision. In practice, an object can be represented by a number of segments at finer levels with consistent details or included in a surrounding region at coarser levels. Thus, a superpixel segmentation hierarchy is of great importance for applications that require different levels of image details. However, there is no method that can generate all scales of superpixels accurately in real time. In this paper, we propose the superhierarchy algorithm which is able to generate multi-scale superpixels as accurately as the state-of-the-art methods but with one to two orders of magnitude speed-up. The proposed algorithm can be directly integrated with recent efficient edge detectors to significantly outperform the state-of-the-art methods in terms of segmentation accuracy. Quantitative and qualitative evaluations on a number of applications demonstrate that the proposed algorithm is accurate and efficient in generating a hierarchy of superpixels.**

***Index Terms*— Superpixel, segmentation, Borůvka algorithm.**

I. Introduction

Superpixels are groupings of pixels with perceptual meanings that serve as primitives to facilitate further analysis. In recent years, superpixels have been the key building blocks for numerous algorithms as they significantly reduce the number of image primitives and contain important visual information when compared to raw pixels. In this paper, we propose a computationally efficient hierarchical superpixel segmentation algorithm that can be applied to a wide range of image analysis tasks. In order to achieve such versatile utilities, a superpixel method should have the following properties:

- **Adherent boundary**: Superpixels should adhere well to image boundaries such that each superpixel only overlaps with one object.
- **Computational efficiency**: The computational complexity for an efficient segmentation algorithm should be independent of the number of superpixels and linear/sublinear in the image size.
- **Hierarchal segmentation**: Superpixel segmentation results at different levels should be close to the human vision system. Numerous algorithms can benefit from multi-resolution representations of images and hierarchical superpixels can be applied to these tasks.
- **Preserved topology**: Superpixels should conform to a simple topology such that neighborhood relationships can be maintained.

While the recent years have seen considerable progress in superpixel segmentation [5], [6], [9]–[11], [13]–[17], the state-of-the-art methods possess only one to two of these properties which limit their application domains. For instance, Liu *et al.* [6] proposed a graph-based method that has high segmentation accuracy. However, it is computationally prohibitive for real-time applications. The SEEDS algorithm [10] achieves a compromise between accuracy and efficiency but its run-time depends on the number of superpixels. One class of approaches [13], [18] generate superpixels that conform to a grid topology which can be conveniently used by several vision tasks. However, the computational complexity of these methods is high and the segmentation accuracy is lower than the state-of-the-art methods [11]. In addition to grid topology, several algorithms [7], [19], [20] use tree structures of regions to represent images. Felzenszwalb and Huttenlocher [5] proposed a method that can accommodates such a structure by adding edges between segments. Nevertheless, its under-segmentation error is high as shown by the recent superpixel benchmark evaluation results [21]. Furthermore, while vision algorithms [22]–[24] benefit from hierarchical or multi-scale segmentation, most superpixel methods do not generate such results directly. Consequently, the same superpixel algorithm needs to be performed several times to generate superpixels at different scales, which increase the computational cost.

In this work, we present a super hierarchy (SH) algorithm that enjoys all the above-mentioned properties (see Table I). Our method efficiently constructs a superpixel hierarchy that can generate any number of superpixels (between one and the number of pixels) on the fly (see Figure 1) in a tree structure. Extensive experiments demonstrate that our algorithm performs favorably against the state-of-the-art methods in terms of accuracy and efficiency.

II. Related Work

There is a rich literature on image segmentation. In this section, we discuss the most relevant methods to this work in

Manuscript received June 27, 2017; revised February 2, 2018; accepted April 17, 2018. Date of publication May 16, 2018; date of current version June 29, 2018. This work was supported in part by the State Key Program of National Natural Science Foundation of China under Grant 61332018, in part by the National Basic Research Program of China under Grant 2015CB351705, in part by NSF CAREER under Grant 1149783, in part by Adobe and Nvidia and in part by the Office of Naval Research under Grant N00014-16-1-2314. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Paul Rodriguez. *(Corresponding author: Ming-Hsuan Yang.)*

X. Wei and Y. Gong are with Xi'an Jiaotong University, Xi'an, China.

Q. Yang is with Muyue Technology Company, Ltd., Shenzhen Software Industrial Base, Shenzhen, China.

N. Ahuja is with the University of Illinois at Urbana–Champaign, Urbana, IL USA.

M.-H. Yang is with the University of California at Merced, Merced, CA USA (e-mail: mhyang@ucmerced.edu).

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.2018.2836300

TABLE I

SUPER HIERARCHY COMPARED TO THE STATE-OF-THE-ART SUPERPIXEL ALGORITHMS. *Property 1*: SEGMENTATION ACCURACY IS MEASURED ACCORDING TO THREE STANDARD METRICS: SEGMENTATION ACCURACY, UNDER-SEGMENTATION ERROR AND BOUNDARY RECALL ON THE BSDS500 DATASET [1], AND THE SEGMENTATION ACCURACY ON THE MSRC-21 DATASET [2] USING THE METHOD PROPOSED IN [3]. *Property 2*: WE REPORT THE AVERAGE RUNTIME REQUIRED TO SEGMENT IMAGES ON A MACHINE WITH AN INTEL I7 3.4 GHZ CPU (SINGLE CORE WITHOUT SIMD INSTRUCTIONS). THE COMPUTATIONAL COMPLEXITY OF EACH METHOD IS ALSO PRESENTED. *Property 3*: THE ADVANTAGE OF MULTI-SCALE SEGMENTATION IS DEMONSTRATED BY THE SALIENCY DETECTION TASK [4]. *Property 4*: WE DEMONSTRATE THE EFFECTIVENESS OF TREE STRUCTURES GENERATED BY THE FH [5], ERS [6], AND SH METHODS USING THE NON-LOCAL COST AGGREGATION ALGORITHM [7] FOR STEREO MATCHING AND EVALUATING THEM ON THE MIDDLEBURY BENCHMARK DATASET [8]. THE TOP THREE ALGORITHMS ARE HIGHLIGHTED IN RED, GREEN AND BLUE, RESPECTIVELY

			FH [5]	SLIC [9]	ERS [6]	SEEDS [10]	LSC [11]	Our SH
Property 1: Segmentation accuracy	Achievable segmentation accuracy (average on BSDS500 [1])		93.5%	94.1%	94.9%	94.7%	95.0%	95.1%
	Under-segmentation error (average on BSDS500)		12.6%	11.5%	10.1%	10.4%	10.0%	9.7%
	Boundary recall (average on BSDS500)		78.0%	67.2%	75.5%	72.9%	78.7%	80.8%
	Semantic segmentation accuracy (using [3] on MSRC [2])		63.8%	65.4%	65.5%	65.8%	65.3%	66.6%
Property 2: Segmentation speed	321 × 481 image (average on BSDS500)		328 ms[i]	108 ms	689 ms	52 ms	302 ms	31 ms
	Complexity of obtaining m scales of superpixels		$O(m)$	$O(m)$	$O(m)$	$O(m)$	$O(m)$	$O(1)$
Property 3: Hierarchical segmentation	Saliency detection	Run-time of superpixels[ii]	1810 ms[i]	554 ms	4160 ms	282 ms	1720 ms	35 ms
	(using [4] on PASCAL-S[iii] [12])	mean absolute error[ii]	0.187	0.190	0.189	0.186	0.190	0.181
Property 4: Maintain topology	Stereo matching error (using [7] on Middlebury [8])		8.10%	-	8.69%	-	-	7.63%

[i] Reported time includes parameter search [ii] Using 5 scales of superpixels [iii] Average size: 500 × 356 pixels

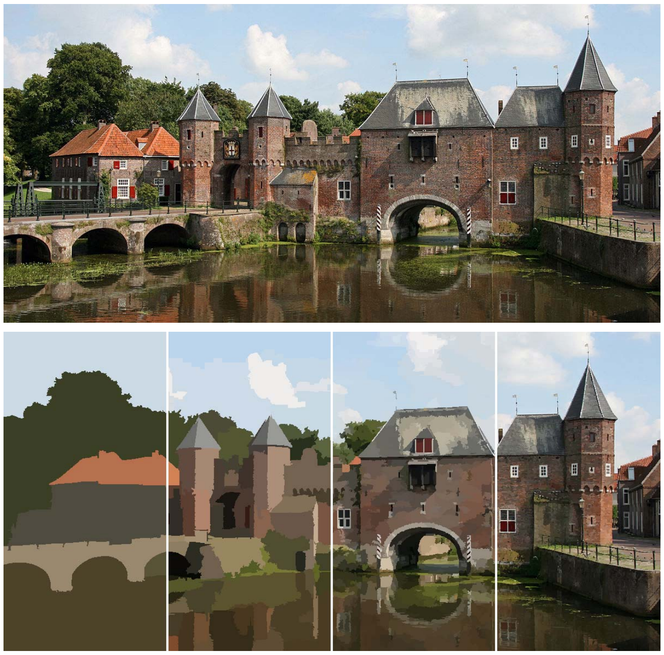


Fig. 1. **Super hierarchy.** Segmentations with 16, 256 ,4096, and 65536 superpixels on an image of 1200 × 582 pixels are shown. Superpixels of *all* scales are obtained *at once*.

the context of hierarchical image segmentation and efficient superpixel extraction.

A. Hierarchical Image Segmentation

Hierarchical image segmentation methods generate a set of segments with different details in which the ones at the coarser levels are composed of regions at the finer levels.

In [25], an image is partitioned into homogeneous regions of all shape, size, as well as degree of photometric homogeneity, and organized in a hierarchical tree. Nodes at the upper levels correspond to large segments while their children nodes capture finer details. A connected segmentation tree [26] includes additional edges to sibling nodes by introducing neighboring Voronoi regions. This segment tree structure has been applied to image classification, semantic image segmentation and object detection [27].

Superpixels can also be grouped baed on region contours. In [1] and [28] the output of a contour detector is transformed into a hierarchical region tree. To this end, it uses the oriented watershed transform to construct a set of initial regions followed by an agglomerative clustering procedure to construct a hierarchical representation. This hierarchical segmentation method has been widely used in recognition and detection problems [29], [30]. Jain *et al.* [31] proposed a method to learn a similarity function using reinforcement learning for agglomerate superpixels hierarchies. The method shows good performance on brain images from serial electron microscopy. Recently, a hierarchical edge-weighted Voronoi tessellation algorithm [32] is proposed for generating multi-sccale superpixels and supervoels.

B. Efficient Superpixel Extraction

Although image segmentation plays an important role in numerous tasks, accurate segmentation methods are often time-consuming and thereby limit the application domains. On the other hand, numerous methods have been developed with the aim to generate superpixels efficiently with low under-segmentation errors. Superpixel algorithms have received much attention since their naming in [33]. Previous methods such as normalized cuts [34], mean shift [35], turbopixels [36] and [37] tackle the problem as a graph partition, mode-seeking or level-set based optimization which have high computational cost. Superpixel using quick shift [38] and watershed [39] are faster but do not exhibit good boundary adherence. Significant progress has been made in recently years to improve the segmentation quality and speed. Here we review the state-of-the-art superpixel algorithms that are either based on image partition or region merging.

1) Image Partition: These algorithms start from an initial coarse partition of an image, typically with a regular grid and then refine the segments iteratively. The SLIC [9] method is an adaptation of the k-means clustering scheme for superpixel generation. It limits the search space of each cluster center and results in a significant speed-up over the conventional k-means clustering scheme. The objective function of POISE [15] is also similar to k-means clustering but is optimized in a coarse-to-fine framework. It also employs a priority queue structure for pixel assignment to avoid the post processing for connectivity check in SLIC. The LSC [11] method maps pixel values and coordinates into a high dimensional feature space and proves that optimizing the cost function of normalized cuts [34] can be by iteratively applying simple k-means clustering in the proposed feature space. The SEEDS [10] scheme is based on an objective function that can be maximized by a simple hill-climbing optimization process efficiently. Instead of computing distance from centers, this method directly exchanges pixels between neighboring superpixels. Recently, Shen *et al.* [16] proposed a fast superpixel method based on the DBSCAN [40] clustering. Methods in these category often start from a regular sampling and have compactness constraints that favor equal-sized or regular-shaped segments. However, the generated superpixels do not adhere to image boundaries well, especially for fine-structured objects as they are not well modeled by regular sampling (see Figure 6 for an example).

2) Region Merging: These algorithms operate on growing regions into segments. The ERS [6] method uses entropy rate of a random walk on a graph as a criteria to generate high-quality superpixels. Humayun *et al.* [14] proposed a superpixel merging algorithm which can generate a small set of seeds that reliably cover a large number of objects of all sizes. Though these methods have good segmentation accuracies, their computational costs are to high for real-time applications. Felzenszwalb and Huttenlocher (FH) [5] proposed a fast graph-based method in which pixels are vertices and edge weights measure the dissimilarity between vertices. Similar to other region merging methods [41], [42], it uses the Kruskal algorithm [43] to construct a minimum spanning forest in which each tree is a segment. Each vertex is initially placed in its own component, and the FH method merges regions by a criterion that the resulting segmentation is neither too coarse nor too fine. The FH method adaptively adjusts segmentation criterion based on the degree of variability in neighboring regions of the image, such that it obeys certain global properties even though greedy decisions are made. However, the under-segmentation error is high as shown by the recent studies [9], [10]. In contrast, the proposed algorithm dynamically adjusts the weights of the graph by aggregating the attributes of clusters during segmentation. We show that this feature aggregation scheme outperforms the state-of-the-art methods in all evaluation metrics.

III. Superpixel Hierarchy

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote an undirected graph consisted of n vertices $v \in \mathcal{V}$ and m edges $e \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ with cardinalities $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Each pixel is associated with a vertex and locally connected to its 4 neighbors. Each edge $e_{ij} = (v_i, v_j)$ is assigned a weight (typically non-negative real value) that measures the dissimilarity between the two vertices. In the superpixel segmentation task, let k denote the number of superpixels to be extracted, a segmentation $\mathcal{S}$ of a graph $\mathcal{G}$ is a partition of $\mathcal{V}$ into k disjoint components and each component $\mathcal{C} \in \mathcal{S}$ corresponds to a connected subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$.

Our algorithm belongs to a class of region merging methods [5], [42]. Different from methods based on the Kruskal algorithm [43] to merge regions, we grow regions based on the the Borůvka method [43]. The advantages are three-fold:

- The Borůvka algorithm has a linear time solution [44] and is parallelizable.
- Neighborhood information can be incorporated into one unified framework. After each iteration, the features are aggregated within the newly formed clusters. This scheme is more robust than methods such as SLIC and LSC that only use per-pixel features to determine its affiliation of the cluster.
- A hierarchy is constructed from which any amount of superpixels can be generated on the fly.

Figure 2 shows on example on superpixels at all scales are generated by the proposed SH algorithm. We first review the Borůvka algorithm, and then address the efficiency and accuracy issues by edge contraction and feature aggregation.

A. Extracting Superpixels via the Borůvka Algorithm

The Borůvka algorithm computes a minimum spanning tree (MST) in a bottom-up manner. Consider a graph as a forest with n trees, namely one vertex itself is a tree. For each tree, we find its nearest neighbor which is connected by the lightest edge and join them together.

Let $\mathcal{C}_2$ denote the nearest neighbor of $\mathcal{C}_1$ ($\mathcal{C}_1$ may not be the nearest neighbor of $\mathcal{C}_2$). We define the distance between two trees as

$$D(\mathcal{C}_1, \mathcal{C}_2) = \min_{v_i \in \mathcal{C}_1, v_j \in \mathcal{C}_2, (v_i, v_j) \in \mathcal{E}} w((v_i, v_j)). \tag{1}$$

After the nearest neighbor search, an auxiliary graph is built where each vertex represents a cluster and each edge corresponds to one chosen light edge. If there are mutual nearest neighbors, we use duplicated edges to represent them in the auxiliary graph; on the other hand, edges in the auxiliary graph are distinctive. Then we use deep-first search to find the connected components. The Borůvka algorithm repeats merging trees in this manner until only one tree is left. The major difference between Borůvka and Kruskal algorithms is that the former searches for edges locally and simultaneously while the later sorts the edges globally and

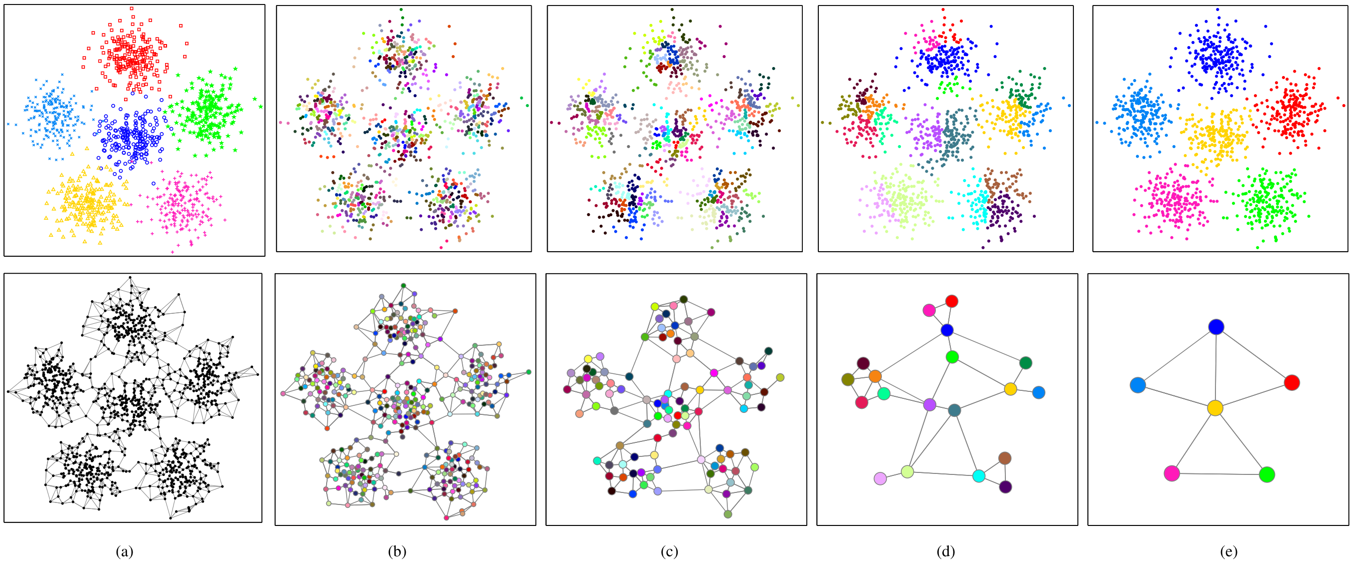


Fig. 2. **Towards concurrently generating superpixels at all scales**. (a) A data set consisting of 6 Gaussian clouds and its 4 nearest neighbor graph. (b)-(e) Results of the first 4 iterations by the SH algorithm. Superpixels of all scales are obtained by region merging. Unlike other region merging methods [5], [41], [42] that use the Kruskal algorithm [43], we adapt the Borůvka algorithm [43] to grow a spanning tree. The advantages are three-fold. First, the Borůvka algorithm has linear time solution [44] and is parallelizable. As shown in (b)-(e), the numbers of nodes and edges are decreasing geometrically after each iteration which enables the SH algorithm to have linear time complexity. Second, neighborhood information can be incorporated in a unified framework. After each iteration, the features are aggregated within the newly formed clusters. Third, a natural hierarchy is constructed during merging from which any amount of superpixels can be generated on the fly.

executes sequentially. As such, the Borůvka algorithm can be processed in parallel. In addition, the Borůvka algorithm assumes that clusters are uniformly distributed and alleviates the drawback of the Kruskal algorithm that tends to generate heavily unbalanced clusters [43]. In the proposed superpixel hierarchy method, we adapt the Borůvka algorithm to construct a MST. Meanwhile, the order that each edge is added to the MST is recorded. Once an edge is added to the MST, the number of trees in the forest is reduced by one. Suppose that k superpixels need to be extracted, we connect vertices by the first $n-k$ edges and have k connected components which are the superpixels exactly.

B. Linear Time Algorithm via Edge Contraction

In this section, we re-formulate the Borůvka algorithm with edge contraction. Instead of maintaining a forest of trees, we contract each tree to a single vertex. This reduces the number of vertices and edges substantially and facilitate achieving significant speed-up.

An edge contraction is illustrated in Figure 3. At each vertex of Figure 3(a), a number denotes the attributes (e.g., pixel intensity). Each edge weight is computed by the absolute distance of the attributes at two ends. An edge contraction is performed between vertex 4 and 2. After contracting the edge, vertex 4 and 2 become a supervertex, resulting in a self-loop and two parallel edges (shown in red edges in Figure 3(b)). A flattening operation is followed (Figure 3(c)) by removing the self-loop and replacing parallel edges with the lightest one.

In the following, we explain the details of our implementation with complexity analysis. We denote the graph at the beginning of the i-th iteration by $\mathcal{G}_i$ and the number of vertices and edges of this graph by n_i and m_i, respectively.

Lemma 1: The SH algorithm stops in $O(\log n)$ merging iterations.

Proof: Each tree is merged with at least one of its neighbors, and the number of trees in $\mathcal{G}_i$ decreases by at least a factor of two. Thus, the SH algorithm stops in $O(\log n)$ iterations. □

Lemma 2: Each merging iteration of SH algorithm runs in $O(m_i)$ time.

Proof: First, the nearest neighbor search for each vertex loops through all edges to determine the lightest edge for the vertex on either endpoint, which takes $O(m_i)$ time. Next, the histogram sorting [45] process of the n_i chosen edges (one for each vertex) takes $O(n_i)$ time. In addition, tree growing uses an auxiliary graph whose vertices are the labels of the original trees and edges correspond to the chosen lightest edges. The auxiliary graph has n_i vertices and n_i edges (where the edges may be duplicated). We find the connected components of this graph using depth-first search, which takes $O(n_i)$ time [45].

The edge contraction process is carried out by histogram sorting edges lexicographically and then removing loops and parallel ones, which takes $O(m_i)$ time. Thus, each iteration of the SH algorithm takes $O(m_i + n_i) = O(m_i)$ time. □

Theorem 1: The SH algorithm takes $O(n)$ to operate on a planar graph.

Proof: When the input is a planar graph, every $\mathcal{G}_i$ is planar as the class of planar graphs is closed under edge contraction [43]. Furthermore, $\mathcal{G}_i$ is also simple (loops and parallel edges have already been removed) such that we can use Euler's formula on the number of edges of a planar simple graph to obtain $m_i \leq 3n_i$. From Lemma 1, we know that

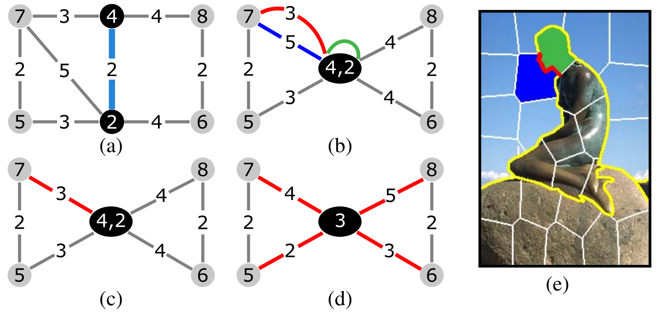


Fig. 3. **Illustration of edge contraction and feature aggregation.** The number in each vertex representing its features. Edge weights are computed by the absolute distance of their two ends. (a)-(c) An edge contraction is performed between vertex 4 and 2. After contracting an edge, the graph becomes a multi-graph with a self-loop (green line) and parallel edges (blue and red line). A flattening operation is followed by removing the self-loop and replacing parallel edges by the lightest one (red line). (d) Feature aggregation is carried out after each iteration by gathering features from newly formed clusters and then updating edge weights (red lines). (e) Our method can combine both color and edge features (e.g., the segments computed via an edge detector, represented by the yellow curves) to improve the segmentation accuracy. Our method explicitly maintains the connectivity of clusters so that the edge confidence between two regions (represented by the red curve in the figure) can be determined directly. This is different from the procedures such as SLIC and LSC that connectivity is enforced at the end.

$n_i \leq n/2^i$, and therefore the total time complexity of the SH algorithm is $O(\sum_i m_i) = O(\sum_i n/2^i) = O(n)$. □

C. Improving Robustness via Feature Aggregation

The Borůvka algorithm can be applied to superpixel segmentation directly. However, a straightforward application of this algorithm does not generate satisfactory results (see Section IV). The issues stem from the greedy and local algorithmic design. Recall that (1) measures the distance between two trees as the minimum edge weight. This measurement is sensitive to outliers as for each tree only the attribute of one vertex is used. In addition to linear time complexity, another advantage of the Borůvka algorithm is that it can incorporate neighborhood information within one unified framework. Since we obtain new clusters after each iteration, it is natural to aggregate the attributes of each cluster and update the weights connected to other clusters. Figure 3(d) illustrates this procedure. After merging vertex 4 and 2, a supervertex is formed with an average value of 3. The self-loop is removed and parallel edges are replaced by one edge. At the same time, weights of all edges connected to the supervertex are updated according to the distance of aggregated attributes (red lines). Feature aggregation takes advantages of "the wisdom of crowds" rather than only two vertices such that better performance can be achieved.

This feature aggregation procedure is in spirit similar to the SLIC method in which centroids are updated by computing the means after each iteration. However, the proposed SH algorithm is more robust and efficient than the SLIC method. First, the SH algorithm performs more robustly as it naturally incorporates neighborhood information during segmentation. In contrast, the SLIC method operates on the pixel level which is sensitive to outliers. Second, both SH and SLIC methods are efficient because they search for limited regions (SH operates on a planar graph and SLIC searches around predefined centers) for cluster assignment in one iteration, our approach is faster as the number of nodes decreases geometrically after each iteration while the SLIC method remains the same. In practice, the SLIC method can be executed with fewer iterations for efficiency at the expense of accuracy. Third, the features used for clustering depend on the task and may not lie in the Euclidean space (e.g., edge confidence) such that centroids can not be computed simply by means. Our experimental results in Section IV show that edge information is useful to superpixels. The proposed SH algorithm explicitly maintains the connectivity of clusters such that the edge confidence between two regions can be determined directly as shown in Figure 3(e). Thus our method can directly combine an edge detector [46] and absorb rapid progress [47], [48] in this area to improve segmentation accuracy. This is our advantage over methods such as SLIC and LSC that connectivity must be enforced at the end such that it is unclear how to integrate edge information into such procedures efficiently.

By incorporating edge confidence, our distance measure becomes

$$D(\mathcal{C}_1, \mathcal{C}_2) = d_c \times d_e, \tag{2}$$

where d_c and d_e are color and edge distance, respectively. The color distance is measured by the absolute difference of mean color. However, the mean color is not sufficient to represent superpixels as they become larger at different scales. For better performance, we measure color difference by the χ^2 distance of color histograms after j iterations (see Section IV for parameter settings). We note that the SEEDS method also adopts a color histogram based distance metric. The difference is that SEEDS uses intersection distance while our method adopts the χ^2 distance. The edge distance d_e is measured by the average edge confidence between the two regions (i.e., the average edge confidence in the red curve in Figure 3(e)).

IV. Experiments

We present the experimental results of the proposed algorithm against the state-of-the-art methods in terms of segmentation accuracy and run-time. The source code will be made available to the public for accessible reproducible research.

A. Experimental Setup

1) Datasets and Evaluated Methods: We use the Berkeley Segmentation Dataset (BSDS500) [1], segmentation challenge of Pascal 2012 Visual Object Classes (SegVOC12) [49], Berkeley Semantic Boundaries Dataset (SBD) [50], and Microsoft Common Objects in Context (COCO14) [51] for performance evaluation. We use the BSDS500 dataset for thorough segmentation evaluation as the images contain accurate annotated segments and boundaries. On the other hand, we use the other datasets for object segmentation evaluation. Although these images do not contain accurate boundaries, the adopted achievable segmentation accuracy (ASA) metric provides the upper bound performance when using superpixels as units for object segmentation.

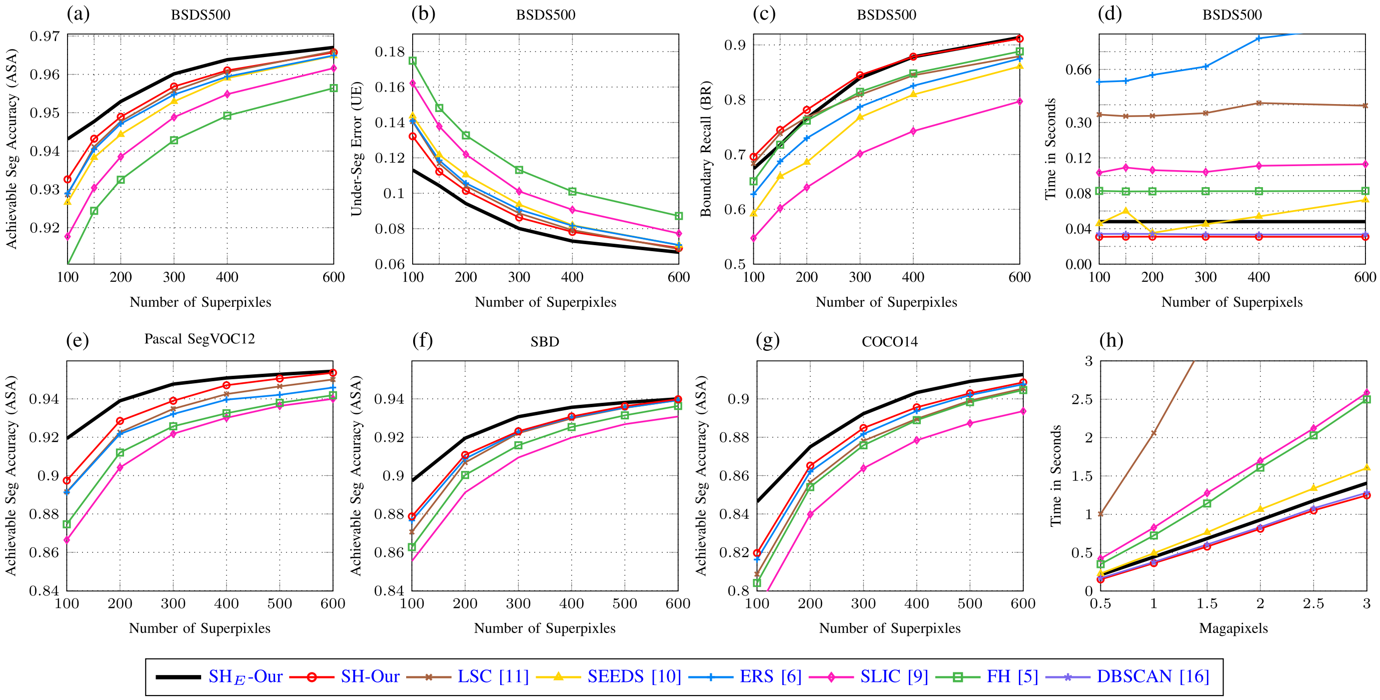


Fig. 4. **Segmentation accuracy and efficiency evaluation on the BSDS500, Pascal SegVOC12, SBD, and COCO14 datasets.** We evaluate the proposed algorithm against the state-of-the-art superpixel segmentation methods.

We evaluate the proposed SH algorithm against the FH [5],[1] SLIC [9],[2] ERS [6],[3] SEEDS [10],[4] and LSC [11][5] methods using the original implementations. The FH and SLIC methods are widely used in the literature due to simplicity and efficiency. The ERS and LSC schemes are considered as the state-of-the-art in terms of accuracy but with high computational load, and the SEEDS approach is the most efficient one among these methods. Evaluation against other methods such as normalized cuts [34] and turbopixels [36] are not presented here as the accuracy or efficiency are far from state-of-the-art methods [21]. We evaluate two versions of proposed scheme: **SH** with color features only, and **SH**$_E$ with both color and edge features.

2) Parameter Settings: In this work, the color difference is measured by the χ^2 distance of histograms (equally divided into k bins) after j iterations. We use the structured forest edges (SFE) [46] to compute the edge feature. These parameters are set based on a training database and fixed as $\{j, k\} = \{4, 20\}$ in our experiments. For other methods, the default parameters are used for fair comparisons.

B. Evaluation Metrics

We use the widely used metrics [21] to evaluate superpixel segmentation methods including achievable segmentation accuracy, under-segmentation error and boundary recall.

The achievable segmentation accuracy (ASA) measures the fraction of ground truth segment that is correctly labeled by superpixels,

$$ASA(\mathcal{S}) = \frac{\sum_k \max_i |s_k \cap g_i|}{\sum_i |g_i|}, \tag{3}$$

where g_i is a ground truth segment, s_k is a superpixel and $|\cdot|$ indicates the size of the segment.

The under-segmentation error (UE) measures the extent superpixels cover the ground truth segment border

$$UE(\mathcal{S}) = \frac{\sum_i \sum_k \min(|s_k \cap g_i|, |s_k - g_i|)}{\sum_i |g_i|}. \tag{4}$$

The boundary recall (BR) measures the percentage of ground truth edges that fall within superpixel boundaries within a margin $\varepsilon = 2$ pixels. Given a ground truth boundary union sets $\mathcal{B}(g)$ and the superpixel boundary sets $\mathcal{B}(s)$, the boundary recall of a segmentation $\mathcal{S}$ is defined by

$$BR(\mathcal{S}) = \frac{TP(\mathcal{S})}{TP(\mathcal{S}) + FN(\mathcal{S})}, \tag{5}$$

where $TP(\mathcal{S})$ is the number of boundary pixels in $\mathcal{B}(g)$ that fall within a boundary pixel $\mathcal{B}(s)$ in the range ε, and $FN(\mathcal{S})$ is the opposite case.

C. Segmentation Accuracy

Figure 4(a)-(c) show the quantitative evaluation results on the BSDS500 dataset using the three metrics. The proposed SH and SH$_E$ algorithms (—○— and ——) perform well in all three metrics. The SH algorithm outperforms the LSC (—×—) and ERS (—+—) methods while it generates superpixels of all scales simultaneously and efficiently. With the assistance of the SFE method [46], the SH$_E$ algorithm performs significantly well. Figure 5 shows sample segmentation results on the BSDS500 test set by the proposed SH algorithm.

[1] http://cs.brown.edu/ pff/segment/, parameters: $\sigma = 0.8$.
[2] http://ivrl.epfl.ch/research/superpixels/, parameters: $m = 10$.
[3] http://mingyuliu.net/, parameters: $\lambda' = 0.5, \sigma = 5.0$.
[4] http://www.mvdblive.org/seeds/, parameters: $\gamma = 1, N = 3, K = 5$.
[5] http://jschenthu.weebly.com/, parameters: $r_c = 0.075$.

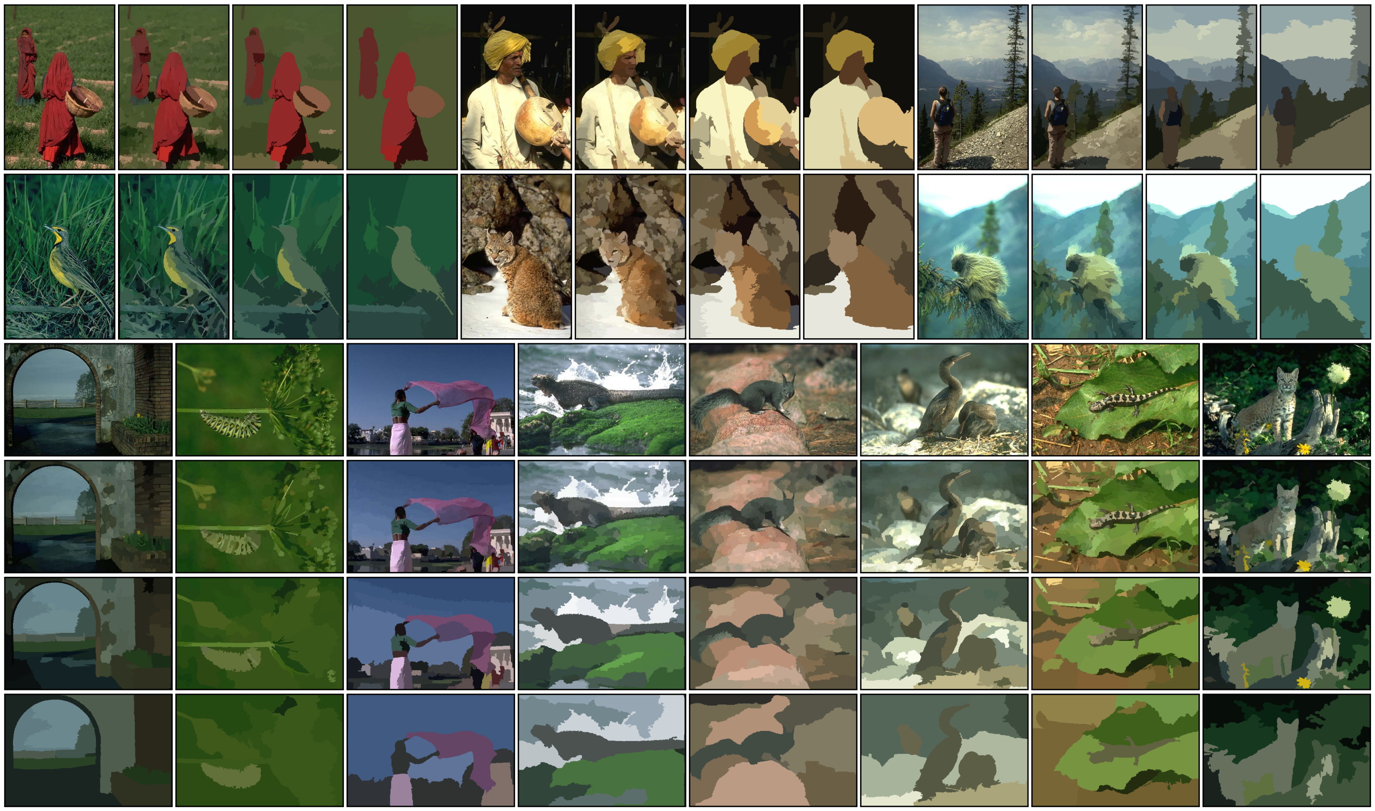


Fig. 5. **Super hierarchy on the BSDS500 test set**. For each image, hierarchical segmentation with 600, 50 and 10 superpixels are shown.

The results are similar on other three object datasets, as shown in Figure 4(e)-(g). On the SegVOC12 dataset, the SH method performs well against the LSC and ERS schemes in terms of ASA. The SLIC approach (—◆—) does not perform well due to its regular sampling strategy (see Figure 6). The LSC scheme starts with regular sampling but maps pixels into a high dimensional feature space. This helps LSC to capture the global image structure, but the segmentation results also depend on image quality (see Figure 7).

Table II shows the results of the SH algorithm with several edge detection methods. The segmentation accuracy of the SH algorithm can be further improved when more effective edge detectors, e.g., [47], are used. In this work, we choose the SFE scheme [46] for computational efficiency. We note that the POISE [14] [6] method also incorporates edge features thus we make a comparison with it. It can be seen from Table II that our method outperforms POISE on several edge detectors.

TABLE II

SH AND POISE [14] WITH SEVERAL EDGE DETECTORS ON THE BSDS500 DATASET

	mean ASA ↑	mean UE ↓	mean BR ↑
SH+SCG [51]	95.4%	9.1%	78.4%
SH+SFE [45]	95.5%	8.9%	79.8%
SH+HED [46]	**95.7%**	**8.6%**	**80.2%**
POISE+SCG [51]	94.3%	10.7%	72.6%
POISE+SFE [45]	94.8%	10.3%	73.5%
POISE+HED [46]	95.0%	10.1%	74.1%

D. Computational Complexity and Run-Time

We analyze the computational complexity and run-time performance of superpixel algorithm based on the number of superpixels and image size on a machine with one 3.4 GHz i7 CPU. We do not use any parallelization (multi-core, SIMD instructions or GPU).

1) Computational Complexity: The FH method uses the Kruskal algorithm [43] to grow region that runs in $O(n \log n)$ worst time with low constant factors. The SLIC scheme uses the k-means clustering procedure with constrained search region which runs in $O(n)$ time of each iteration but needs several iterations to convergence. The SEEDS algorithm maximizes its energy function via hill-climbing optimization at pixel and block levels. The run-time of block-level optimization depends on the number of superpixels. The ERS approach constructs a submodular and monotonic objective function that can be optimized by a lazy greedy algorithm. The worst case complexity of lazy greedy algorithms is $O(n^2 \log n)$ while in [6] it is shown that on average the complexity is $O(n \log n)$. The LSC procedure shares a similar framework with the SLIC method and the complexity is also $O(n)$. However, it is apparently slower than SLIC because it operates in high dimensional feature space and requires more iterations to achieve higher segmentation accuracy. The recent DBSCAN [16] [7] method uses a similar framework to SLIC and also has a $O(n)$

[6] http://rehg.org/poise/, parameters: $\sigma = 5.0$.

[7] https://github.com/shenjianbing, parameters: $\alpha_1 = 0.6, \alpha_2 = 0.4, \alpha_3 = 1, \Psi = 30$.

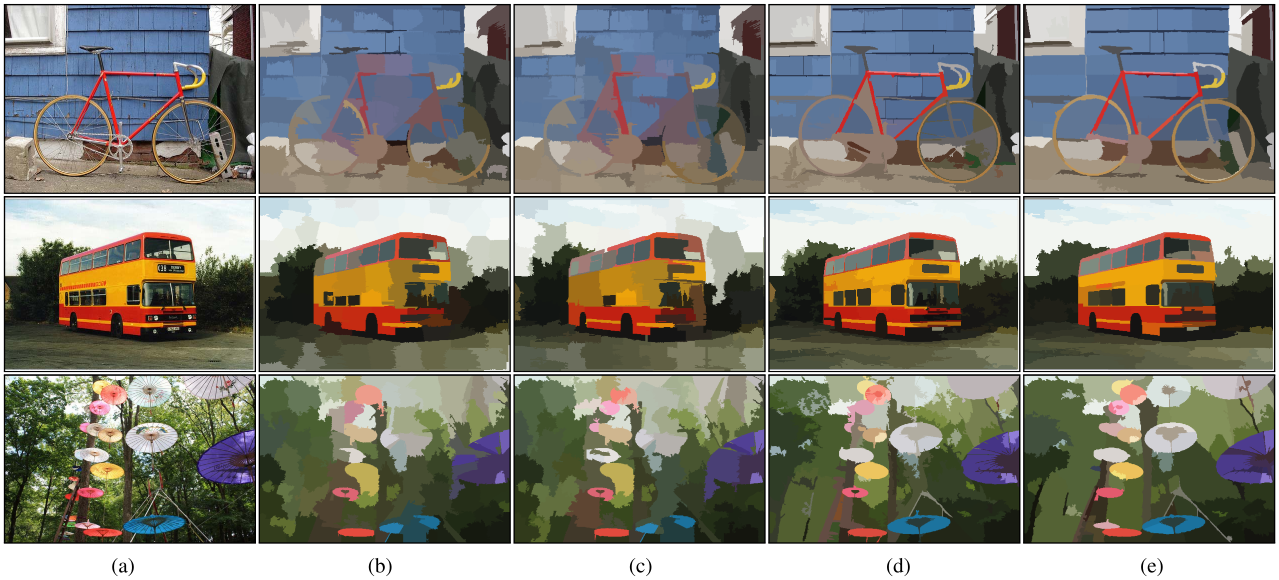


Fig. 6. **SH compared to SLIC with 100 superpixels on the Pascal segVOC12, SBD, and COCO14 datasets.** The SLIC method does not perform well on fine-structured objects even when the compact factor m is reduced. The SH$_E$ method improves the SH algorithm by using edge features. (a) input. (b) SLIC ($m = 10$). (c) SLIC ($m = 2$). (d) SH. (e) SH$_E$.

TABLE III

SEMANTIC SEGMENTATION ACCURACY. FOLLOWING [9], WE USE THE METHOD OF [3] ON THE MSRC-21 DATASET [2] FOR EVALUATION. THE GLOBAL SCORE GIVES THE PERCENTAGE OF CORRECTLY CLASSIFIED PIXELS AND THE AVERAGE SCORE PROVIDES THE PER-CLASS AVERAGE [53]. THE GLOBAL SCORES OF THE SEEDS, LSC, AND SH METHODS ARE SIMILAR WHILE THE SH ALGORITHM IMPROVES THE PER-CLASS ACCURACY SIGNIFICANTLY

	Build.	Grass	Tree	Cow	Sheep	Sky	Air.	Water	Face	Car	Bicycle	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat	**Global**	**Average**
FH [5]	77.9	92.4	88.1	75.8	75.4	89.3	48.5	58.5	82.6	60.0	82.2	45.0	72.6	15.2	88.0	45.0	85.0	46.3	36.6	70.3	04.7	78.1%	63.8%
SLIC [9]	78.0	93.6	83.8	87.9	74.4	92.6	46.8	68.6	84.6	57.7	76.5	67.3	57.9	19.3	92.5	40.6	82.9	60.8	42.9	57.6	07.5	79.5%	65.4%
ERS [6]	78.9	92.5	85.5	83.6	59.1	95.5	68.6	67.9	86.4	54.7	74.1	53.1	69.8	25.0	93.2	37.0	84.1	51.3	48.6	52.7	13.8	79.6%	65.5%
SEEDS [10]	77.1	92.7	88.3	81.8	71.8	95.3	57.2	70.2	82.5	53.7	76.6	63.5	67.0	22.8	94.3	38.4	85.5	50.0	48.0	55.8	10.4	80.3%	65.8%
LSC [11]	81.5	93.4	84.4	84.7	78.0	93.4	46.4	73.2	83.7	52.7	77.4	73.8	60.0	16.3	92.5	25.2	85.9	51.5	48.1	57.8	11.4	**80.5%**	65.3%
SH	80.6	92.9	84.9	86.8	70.6	92.1	58.0	69.0	83.2	59.3	80.0	65.5	79.3	14.2	86.4	42.0	85.5	47.3	52.1	58.3	10.1	80.4%	**66.6%**

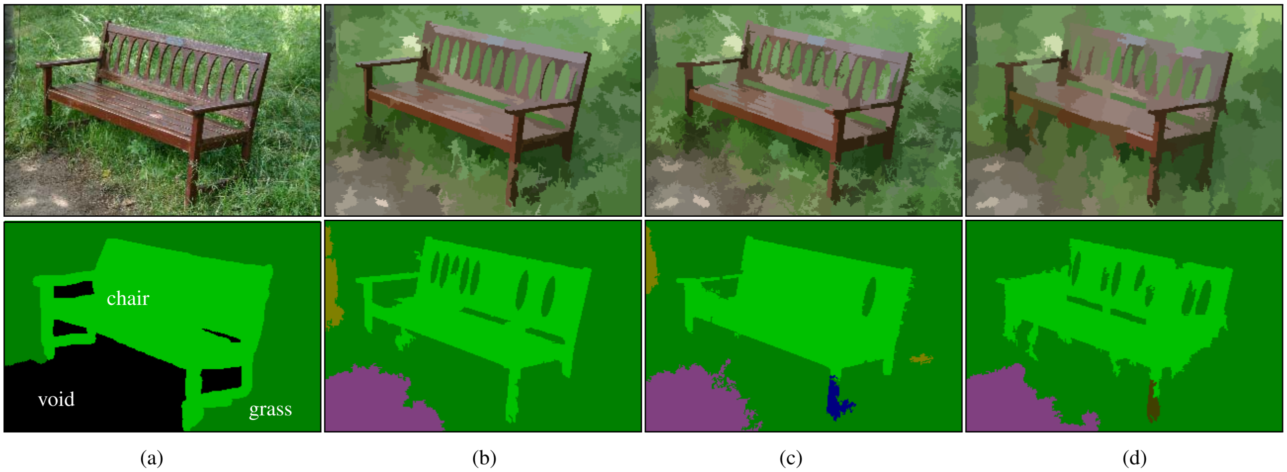


Fig. 7. **Semantic segmentation examples.** Top: input image and segmentation with 200 superpixels. Bottom: ground truth (GT) and classification results for the method of [3] with different superpixel algorithms. (a) input/GT. (b) SH. (c) LSC [11]. (d) SLIC [9].

complexity. As analyzed in Section III-B, the computational complexity of the proposed SH algorithm is $O(n)$. Compared to other linear time methods, the proposed algorithm has $O(1)$ complexity to generate m scales of superpixels while that of other methods is $O(m)$. To reduce computational cost for other methods (e.g., FH) for generating multi-scale representations, it is possible to first generate a large number of superpixels and then merge the neighbors using a similar scheme to ours. However, this approach is more ad-hoc than our unified framework and does not facilitate parallel computation.

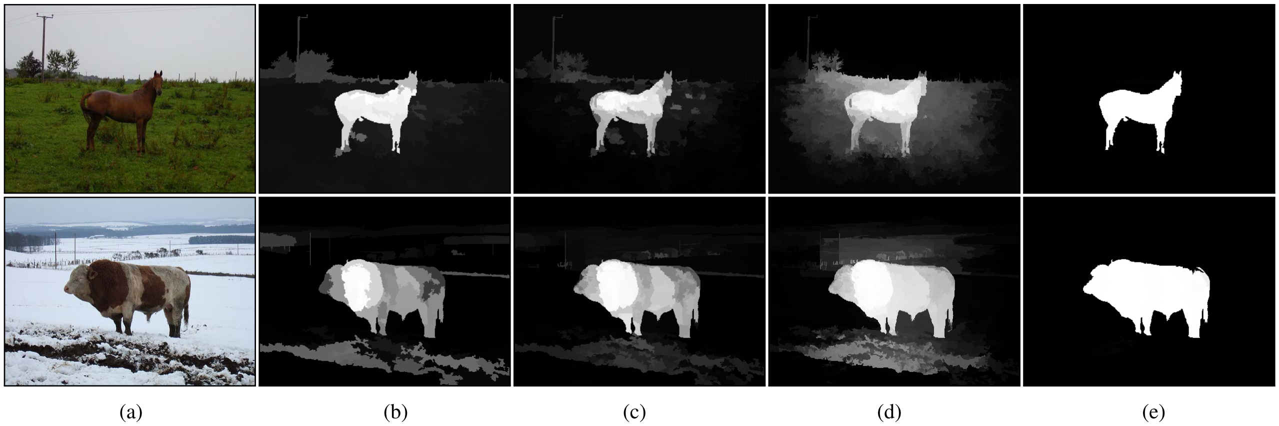


Fig. 8. **Multi-scale saliency detection using [4] with the SH algorithm.** Results for the multi-scale saliency detection algorithm [4] with 5 scales of superpixels which can be generated by SH at once. (a) input. (b) 100 superpixels. (c) 300 superpixels. (d) 800 superpixels. (e) integrated.

TABLE IV

MULTI-SCALE SEGMENTATION FOR SALIENCY DETECTION USING [4] WITH DIFFERENT SUPERPIXEL ALGORITHMS. THE MULTI-SCALE METHOD INTEGRATES 5 SCALES OF SUPERPIXELS. NOTE THAT MULTI-SCALE SUPERPIXELS CLEARLY AND CONSISTENTLY IMPROVE SALIENCY DETECTION FOR ALL METHODS. THE SH ALGORITHM PERFORMS WELL BOTH IN SINGLE-SCALE AND MULTI-SCALE CASES. IN ADDITION, SH IS MORE EFFICIENT THAN OTHERS SINCE ALL SCALES OF SUPERPIXELS ARE GENERATED AT ONE TIME (8 TIMES FASTER THAN THE SECOND BEST, SEEDS, ON THE PASCAL-S DATASET)

	Mean Absolute Error						
	PASCAL-S[12]			ECSSD[23]		DUT-OMRON[53]	
	Single	Multi	Time	Single	Multi	Single	Multi
FH [5]	0.227	0.187	1810 ms	0.186	0.141	0.177	0.154
SLIC [9]	0.225	0.190	554 ms	0.183	0.139	0.191	0.169
ERS [6]	0.224	0.189	4160 ms	0.182	0.138	0.189	0.166
SEEDS [10]	0.223	0.186	282 ms	0.179	0.137	0.178	0.155
LSC [11]	0.226	0.190	1720 ms	0.185	0.139	0.195	0.167
SH	**0.217**	**0.181**	**35 ms**	**0.178**	**0.133**	**0.172**	**0.146**

2) Run-Time: Figure 4(d) shows the run-time with respect to the number of superpixels on the BSDS500 dataset. The run-time of the SH (—○—) and FH (—□—) methods is independent of superpixel numbers. The run-time of the SLIC (—◆—) and LSC (—×—) schemes fluctuates but remains constant in general. The LSC method is 10 times slower than the SH algorithm. The run-time of the SEEDS scheme (—▲—) varies significantly and the worst case here is twice slower. The ERS (—+—) approach is 20 times slower than the SH algorithm and the run-time increases with respect to the number of superpixels. The DBSCAN (—★—) method has a similar speed to the proposed SH method.

Figure 4(h) shows the run-time with respect to image size. Each set has 10 images and we report the average time. The average size of superpixels is 1024 for all image sets and algorithms. The result with the ERS algorithm is not plotted due to its high computational cost. The FH, SLIC, SEEDS, LSC, DBSCAN, and SH methods all run in time nearly linear in image size in practice.

V. APPLICATIONS

We show how the properties of the SH algorithm as summarized in Table I facilitate three image analysis tasks: semantic segmentation, saliency detection and stereo matching.

A. Semantic Segmentation

Semantic segmentation aims to assign pre-defined class labels to every pixel in an image. One effective approach for this task is to formulate this problem as an energy minimization task on a conditional random field (CRF) [3], [53]. By operating directly on the superpixels rather than pixels, the number of nodes in the CRF is significantly reduced (typically from 10^5 to 10^2 per image [53]). Thus, it requires much less time for inference [53].

Similar to [9], we use the method of [3] to evaluate superpixel algorithms on the MSRC-21 dataset [2]. As the original annotations are imprecise, we use labelings by [55] for performance evaluation. All settings of [3] are fixed for all superpixel methods. Table III shows that the SEEDS, LSC and SH methods perform well on per-class evaluation while the SH algorithm improves the per-class accuracy significantly. Figure 7 shows sample semantic segmentation results.

B. Saliency Detection

The goal of saliency detection is to determine whether a pixel belongs to the most salient object in an image. For this task, we show that a multi-scale image representation is effective for saliency detection. We use the method in [4] to detect salient regions at different scales of an image and apply the fusion method proposed in [4] to combine multiple saliency maps. We use five scales of superpixels (from 100 to 1000) for experiments on the PASCAL-S [12], ECSSD [23] and DTU-OMERON [54] datasets. As shown in Table IV and Figure 8, multi-scale segmentations can be used to effectively improve saliency detection accuracy.

C. Stereo Matching

To demonstrate the usefulness of tree structure provided by the SH algorithm, we integrate it with the non-local

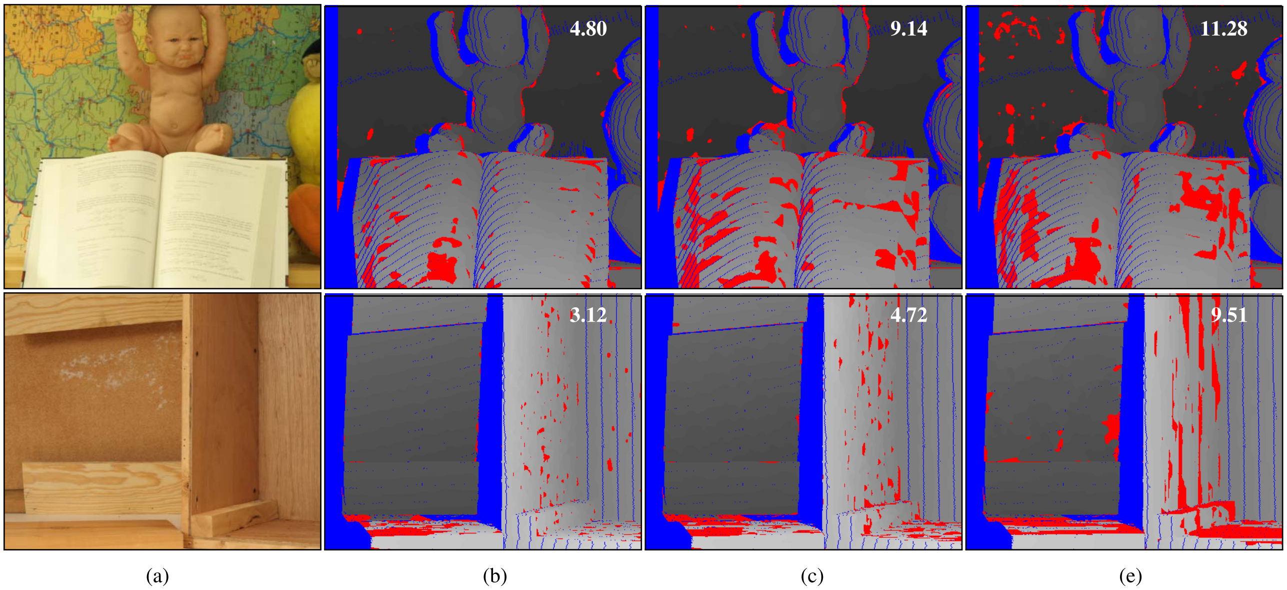


Fig. 9. **Stereo matching using different tree structures.** Similar to [20], we use the method of [7] with different tree structures on the Middlebury dataset for evaluation. The occlusion regions are marked in blue and the erroneous pixels are marked in red. Numbers in the upper-right corner indicate percentages of bad pixels. (a) left image. (b) SH. (c) ERS [6]. (e) FH [5], [20].

TABLE V

STEREO MATCHING EVALUATION ON THE MIDDLEBURY DATASET [8] USING DIFFERENT *TREE STRUCTURES*. RESULTS FOR THE METHOD OF [7] WITH 4 TREE STRUCTURES: MST [7], FH [5], ERS [6] AND SH. PERCENTAGES OF THE ERRONEOUS PIXELS IN NON-OCCLUSION REGIONS WITH THRESHOLD 1 ARE USED TO EVALUATE THE AGGREGATION ACCURACY OF THE STRUCTURES. THE SUBSCRIPTS REPRESENT THE RELATIVE RANK OF THE METHODS ON EACH DATA SET. THE SH ALGORITHM GENERATES THE MOST ACCURATE DISPARITY MAP ON 13 DATA SETS

	Tsukuba	Venus	Teddy	Cones	Aloe	Art	Baby1	Baby2	Baby3	Books	Cloth2	Cloth3	Dolls	Flower.	Lamp.	Laundry	Midd1	Moebius	Wood1	Avg.Error	Avg.Rank
MST [7]	$\mathbf{1.71}_1$	$\mathbf{0.64}_1$	7.14_2	3.89_3	4.46_3	10.54_4	8.89_4	13.53_4	6.37_4	10.10_4	3.61_3	1.95_4	5.70_3	19.21_4	11.41_4	12.92_2	30.99_2	$\mathbf{7.92}_1$	10.13_4	9.01_4	3.00_3
FH [5]	1.89_2	0.76_2	7.55_3	3.64_2	4.15_2	10.51_3	7.37_3	11.28_3	5.36_3	09.05_2	3.15_2	1.58_2	5.39_2	15.73_3	11.14_2	$\mathbf{12.70}_1$	$\mathbf{24.92}_1$	8.16_4	09.51_3	8.10_2	2.37_2
ERS [6]	2.65_4	1.45_4	8.87_4	3.94_4	4.57_4	10.00_2	5.78_2	09.14_2	5.02_2	09.77_3	4.03_4	1.82_3	5.86_4	14.81_2	$\mathbf{10.78}_1$	14.97_4	38.89_4	8.08_2	04.72_2	8.69_3	3.00_3
SH	2.22_3	1.15_3	$\mathbf{7.05}_1$	$\mathbf{3.50}_1$	$\mathbf{3.27}_1$	$\mathbf{08.12}_1$	$\mathbf{4.93}_1$	$\mathbf{04.80}_1$	$\mathbf{4.69}_1$	$\mathbf{08.77}_1$	$\mathbf{2.18}_1$	$\mathbf{1.17}_1$	$\mathbf{4.77}_1$	$\mathbf{12.61}_1$	11.20_3	14.63_3	38.72_3	8.08_2	$\mathbf{03.12}_1$	$\mathbf{7.63}_1$	$\mathbf{1.58}_1$

aggregation method [7], [56] for stereo matching. Different from previous local stereo methods, the approach [7], [56] performs cost aggregation over the entire image with a MST in a non-local manner. The method is computationally efficient, with the complexity comparable to uniform box filtering but has edge-preserving performance. In addition, this method has also been applied to depth upsampling [56], image filtering [57], background subtraction [58] and saliency detection [59].

Similar to [20], we quantitatively evaluate the aggregation accuracy with the MST, FH, ERS methods and the SH algorithm on the Middlebury dataset. We follow the steps in [20] to build a tree structure for FH and ERS. First, image pixels are grouped into a set of subtrees (around 200 superpixels). Second, all the subtrees are linked by the rest of light edges to produce the final tree. The MST and SH already produce a whole tree structure which is used directly. All the methods use the same cost volume and do not use any post-processing. The disparity error rates in non-occlusion regions are used for evaluation. Table V shows the experimental results where the subscripts represent relative rank of the methods on each dataset. All segmentation-based structures improve the performance of the basic MST approach. The performance of the SH algorithm is higher than that of the other tree methods. It achieves the lowest average error rate on 13 (out of 19) stereo image pairs.

VI. CONCLUSIONS

In this work, we propose an effective hierarchical superpixel segmentation algorithm that can be used in a wide range of computer vision tasks. Extensive experimental results demonstrate that the proposed algorithm is accurate and efficient in generating a hierarchy of superpixels that can be applied to numerous tasks. Our future work includes speeding up the proposed method on GPUs and applying it to point cloud and video segmentation.

REFERENCES

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[2] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proc. ECCV*, 2006, pp. 1–15.

[3] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," in *Proc. IJCV*, 2008, pp. 300–316.

[4] Y. Qin, H. Lu, Y. Xu, and H. Wang, "Saliency detection via cellular automata," in *Proc. CVPR*, Jun. 2015, pp. 110–119.

[5] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

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

[7] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proc. CVPR*, Jun. 2012, pp. 1402–1409.

[8] D. Scharstein and R. Szeliski, "Middlebury stereo datasets." Accessed: Apr. 25, 2016. [Online]. Available: http://vision.middlebury.edu/stereo/data/

[9] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[10] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "SEEDS: Superpixels extracted via energy-driven sampling," in *Proc. ECCV*, 2012, pp. 13–26.

[11] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *Proc. CVPR*, Jun. 2015, pp. 1356–1363.

[12] Y. Li, X. Hou, C. Koch, J. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proc. CVPR*, 2014, pp. 1–15.

[13] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *Proc. CVPR*, Jun. 2008, pp. 1–8.

[14] A. Humayun, F. Li, and J. M. Rehg, "The middle child problem: Revisiting parametric min-cut and seeds for object proposals," in *Proc. ICCV*, Dec. 2015, pp. 1600–1608.

[15] J. Yao, M. Boben, S. Fidler, and R. Urtasun, "Real-time coarse-to-fine topologically preserving segmentation," in *Proc. CVPR*, Jun. 2015, pp. 2947–2955.

[16] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by DBSCAN clustering algorithm," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5933–5942, Dec. 2016.

[17] Y. Zhang, X. Li, X. Gao, and C. Zhang, "A simple algorithm of superpixel segmentation with boundary constraint," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 7, pp. 1502–1514, Jul. 2017.

[18] A. P. Moore, S. J. Prince, and J. Warrell, "'Lattice cut'—Constructing superpixels using layer constraints," in *Proc. CVPR*, Jun. 2010, pp. 2117–2124.

[19] S. Todorovic and N. Ahuja, "Unsupervised category modeling, recognition, and segmentation in images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 12, pp. 2158–2174, Dec. 2008.

[20] X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang, "Segment-tree based cost aggregation for stereo matching," in *Proc. CVPR*, Jun. 2013, pp. 313–320.

[21] P. Neubert and P. Protzel, "Superpixel benchmark and comparison," in *Proc. Forum Bildverarbeitung*, 2012, pp. 1–12.

[22] P. Kohli, L. Ladický, and P. H. S. Torr *et al.*, "Robust higher order potentials for enforcing label consistency," *Int. J. Comput. Vis.*, vol. 82, no. 3, pp. 302–324, May 2009.

[23] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *Proc. CVPR*, Jun. 2013, pp. 1155–1162.

[24] 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.

[25] E. Akbas and N. Ahuja, "From ramp discontinuities to segmentation tree," in *Proc. ACCV*, 2010, pp. 123–134.

[26] N. Ahuja and S. Todorovic, "Connected segmentation tree—A joint representation of region layout and hierarchy," in *Proc. CVPR*, Jun. 2008, pp. 1–8.

[27] E. Akbas and N. Ahuja, "Low-level hierarchical multiscale segmentation statistics of natural images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 9, pp. 1900–1906, Sep. 2014.

[28] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multi-scale combinatorial grouping," in *Proc. CVPR*, Jun. 2014, pp. 328–335.

[29] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik, "Recognition using regions," in *Proc. CVPR*, Jun. 2009, pp. 1030–1037.

[30] 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.

[31] V. Jain, S. C. Turaga, K. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung, "Learning to agglomerate superpixel hierarchies," in *Proc. NIPS*, 2011, pp. 648–656.

[32] Y. Zhou, L. Ju, and S. Wang, "Multiscale superpixels and supervoxels based on hierarchical edge-weighted centroidal Voronoi tessellation," in *Proc. WACV*, Jan. 2015, pp. 1076–1083.

[33] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. ICCP*, 2003, p. 10.

[34] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[35] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[36] 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.*, vol. 31, no. 12, pp. 2290–2297, Dec. 2009.

[37] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *Proc. ECCV*, 2010, pp. 211–224.

[38] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Proc. ECCV*, 2008, pp. 705–718.

[39] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, Jun. 1991.

[40] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.

[41] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Trans. Image Process.*, vol. 9, no. 4, pp. 561–576, Apr. 2000.

[42] F. Calderero and F. Marques, "Region merging techniques using information theory statistical measures," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1567–1586, Jun. 2010.

[43] D. B. West *et al.*, *Introduction to Graph Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[44] M. Mareš, "Two linear time algorithms for MST on minor closed graph classes," *Arch. Math.*, vol. 40, no. 3, pp. 315–320, 2004.

[45] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[46] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proc. ICCV*, 2013, pp. 1841–1848.

[47] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. ICCV*, 2015, pp. 1395–1403.

[48] J. Yang, B. L. Price, S. Cohen, H. Lee, and M. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proc. CVPR*, 2016, pp. 193–202.

[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[50] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. ICCV*, 2011, pp. 991–998.

[51] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.

[52] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *Proc. NIPS*, 2012, pp. 584–592.

[53] J. M. Gonfaus, X. Boix, J. Van de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonzàlez, "Harmony potentials for joint classification and segmentation," in *Proc. CVPR*, Jun. 2010, pp. 3280–3287.

[54] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proc. CVPR*, Jun. 2013, pp. 3166–3173.

[55] T. Malisiewicz and A. A. Efros, "Improving spatial support for objects via multiple segmentations," in *Proc. BMVC*, 2007, pp. 1–10.

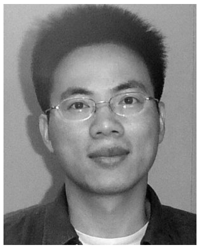
[56] Q. Yang, "Stereo matching using tree filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 4, pp. 834–846, Apr. 2015.

[57] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang, "Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 555–569, Feb. 2014.

[58] M. Chen, Q. Yang, Q. Li, G. Wang, and M.-H. Yang, "Spatiotemporal background subtraction using minimum spanning tree and optical flow," in *Proc. ECCV*, 2014, pp. 521–534.

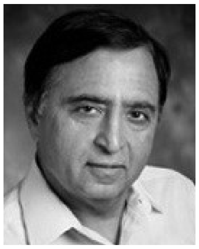
[59] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien, "Real-time salient object detection with a minimum spanning tree," in *Proc. CVPR*, Jun. 2016, pp. 2334–2342.

Xing Wei received the B.S. degree in control science and engineering from Xi'an Jiaotong University, Shannxi, China, in 2013, where he is currently pursuing the Ph.D. degree with the Institute of Artificial Intelligence and Robotics. He is interested in computer vision, image processing, and machine learning. Specifically, his research interests include 3D scene understanding, large-scale visual recognition, object detection, and segmentation.



Qingxiong Yang (M'11) received the B.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2010. From 2011 to 2016, he was an Assistant Professor with the Computer Science Department, City University of Hong Kong, Hong Kong. His current research interests include computer vision and computer graphics. He is a recipient of the Best Student Paper Award at MMSP in 2010 and the Best Demo Award at CVPR in 2007.

Yihong Gong (F'18) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Tokyo, Japan, in 1987, 1989, and 1992, respectively. In 1992, he joined Nanyang Technological University, Singapore, as an Assistant Professor with the School of Electrical and Electronic Engineering. From 1996 to 1998, he was a Project Scientist with the Robotics Institute, Carnegie Mellon University, USA. Since 1999, he has been with the Silicon Valley Branch, NEC Labs America, as a Group Leader, the Department Head, and the Branch Manager. In 2012, he joined Xian Jiaotong University, China, as a Distinguished Professor. His research interests include image and video analysis, multimedia database systems, and machine learning.



Narendra Ahuja (F'92) received the B.E. degree (Hons.) in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1972, the M.E. degree (Hons.) in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1974, and the Ph.D. degree in computer science from the University of Maryland, College Park, MD, USA, in 1979. From 1974 to 1975, he was a Scientific Officer with the Department of Electronics, Government of India, New Delhi, India. From 1975 to 1979, he was with the Computer Vision Laboratory, University of Maryland. Since 1979, he has been with the University of Illinois at Urbana–Champaign, Urbana, IL, USA, where he is currently a Donald Biggar Willet Professor with the Department of Electrical and Computer Engineering, Beckman Institute, Urbana, and the Coordinated Science Laboratory. His current research interests include extraction and representation of spatial structure in images and video, integrated use of multiple image-based sources for scene representation and recognition, versatile sensors for computer vision, and applications including visual communication, image manipulation, and information retrieval. He is a fellow of the ACM, AAAI, AAAS, IAPR, and SPIE.

Ming-Hsuan Yang (SM'06) was a Senior Research Scientist with the Honda Research Institute, where he was involved in vision problems of humanoid robots. He is currently a Professor in electrical engineering and computer science with the University of California at Merced (UC Merced), Merced. He is a Senior Member of the ACM. He received the U.S. National Science Foundation CAREER Award in 2012, the Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He served as an Associate Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE from 2007 to 2011. He is an Associate Editor of the *International Journal of Computer Vision, Image and Vision Computing*, and the *Journal of Artificial Intelligence Research.*