Rooted Spanning Superpixels

Dengfeng Chai¹

Received: 25 April 2018 / Accepted: 9 June 2020 / Published online: 20 July 2020
© The Author(s) 2020

Abstract

This paper proposes a new approach for superpixel segmentation. It is formulated as finding a rooted spanning forest of a graph with respect to some roots and a path-cost function. The underlying graph represents an image, the roots serve as seeds for segmentation, each pixel is connected to one seed via a path, the path-cost function measures both the color similarity and spatial closeness between two pixels via a path, and each tree in the spanning forest represents one superpixel. Originating from the evenly distributed seeds, the superpixels are guided by a path-cost function to grow uniformly and adaptively, the pixel-by-pixel growing continues until they cover the whole image. The number of superpixels is controlled by the number of seeds. The connectivity is maintained by region growing. Good performances are assured by connecting each pixel to the similar seed, which are dominated by the path-cost function. It is evaluated by both the superpixel benchmark and supervoxel benchmark. Its performance is ranked as the second among top performing state-of-the-art methods. Moreover, it is much faster than the other superpixel and supervoxel methods.

Keywords Superpixels · Segmentation · Spanning forest

1 Introduction

Superpixels have become effective alternative to pixels in the past decade. They result from image oversegmentation, which is dedicated to reducing image complexity while avoiding undersegmentation (Ren and Malik 2003). An image is oversegmented into many perceptually meaningful segments such that each segment covers a local region consisting of some connected similar pixels, and each segment is called as a *superpixel*. Superpixels have two prime advantages over pixels. One advantage is the perceptual meaning. In contrast with raw pixels generated by digital sampling, superpixels are formed by pixel grouping,

whose principles are based on the classical Gestalt theory (Wertheimer 1938) assuring superpixels enhanced perceptual meaning. This characteristic facilitates defining higher order potentials, high order conditional random fields and associative hierarchical random fields (Arnab et al. 2016). The other advantage is the complexity. Since many pixels are grouped into one superpixel, the number of superpixels is much smaller than that of pixels. When superpixels instead of pixels serve as atoms, the size of an image is reduced greatly. The size reduction can accelerate the processing in subsequent tasks, and in turn, it is possible to employ some advanced methods which might be computationally infeasible for the huge number of pixels. For example, compared with pixel-based convolutional neural network (CNN), superpixel-based CNN (SuperCNN) enables efficient analysis of large context (He et al. 2015). Moreover, superpixels can be further grouped to generate some object proposals (Uijlings et al. 2013). It can dramatically reduce the number of possible candidates to be checked in object detection. For example, both R-CNN (Girshick et al. 2014) and Fast R-CNN (Girshick 2015) benefit from such reduction of candidate numbers. Such advantages lead to successful applications in many vision problems covering image segmentation (Boix et al. 2012; Liu et al. 2018), video segmentation (Tsai et al. 2016), semantic segmentation (Farabet et al. 2012; Mostajabi

Communicated by Yuri Boykov.

This work was supported by the National Natural Science Foundation of China (No. 41571335).

The source code for RSS algorithm can be found at <https://github.com/dfchai/Rooted-Spanning-Superpixels>.

✉ Dengfeng Chai
chaidf@zju.edu.cn

¹ Key Laboratory of Geoscience Big Data and Deep Resource of Zhejiang Province, School of Earth Sciences, Zhejiang University, No. 38, Zheda Road, Hangzhou 310027, Zhejiang, China

et al. 2015; Gadde et al. 2016), stereo computation (Mičušík and Košecká 2010; Guney and Geiger 2015), object tracking (Wang et al. 2011), objectness measuring (Alexe et al. 2012), object proposal generation (Hosang et al. 2015), etc.

Although superpixel segmentation is application oriented in some way, some general characteristics are expected:

1. *locality* a superpixel covers a local region;
2. *coherency* a superpixel is composed of similar pixels;
3. *connectivity* a superpixel is composed of connected pixels;
4. *compactness* a superpixel is compact in absence of edges, it is expected to be square in an area of constant color;
5. *adherence* superpixel boundaries adhere well to object boundaries;
6. *uniformity* superpixels are homogeneous in sizes and shapes;
7. *adaptivity* compactness and adherence are maintained adaptively;
8. *efficiency* segmentation should be computationally and memory efficient.
9. *scalability* supervoxel segmentation can be achieved in the same way;

These characteristics follow principles of perceptual grouping and support general applications. They are the criteria for developing superpixel segmentation methods. Although there are many approaches available for use, none of them satisfy all the aforementioned characteristics. For example, Watersheds generate superpixels of irregular sizes and shapes (Vincent and Soille 1991), which conflict with the compactness and uniformity.

1.1 Existing Superpixel Methods

Three types of formulations can be distinguished in the existing literature: graph partitioning, boundary evolution and feature space analysis.

Graph partitioning is the most common formulation for superpixel boundary determination. A vertex denotes a pixel, an edge links two neighboring pixels, and all the vertices and edges constitute a graph representing an image. Superpixel segmentation is achieved by partitioning the graph into a set of connected subgraphs, each of which denotes one superpixel. The pioneering work is based on Normalized Cut (NC) (Ren and Malik 2003). It measures both the total dissimilarity between the different subgraphs as well as the total similarity within the subgraphs by a criterion based on normalized cut, and solves a generalized eigenvalue system to find the optimal cut partitioning the graph (Shi and Malik 2000). It produces uniform, compact and coherent superpixels. However, its computational and memory requirements are high and its boundary adherence is relatively poor. Some

methods such as Superpixel Lattice (SL) (Moore et al. 2008), Lattice Cut (LC) (Moore et al. 2010) Superpixels via Pseudo-Boolean Optimization (SPBO) and Superpixels via Quaternary Labeling (SQL) (Chai 2019) find both horizontal and vertical boundaries to produce a regular grid of superpixels. However, the segmentation performance is sacrificed by these constraints. Compact Superpixels (CS), Variable Patch Superpixels (VPS), and Constant Intensity Superpixels (CIS) work similarly but generate superpixels without lattice structure (Veksler et al. 2010). These approaches optimize an objective function to find the graph cuts. The objective function consists of a data term and a smooth term. The data term favors coherent superpixels and the smooth term encourages the boundaries to align with intensity edges. Their computation costs are high and their performances are not good. Entropy Rate Superpixels (ERS) result from greedy optimization of an objective function consisting of entropy rate of random walk on graph and a balancing term (Liu et al. 2011). The entropy rate favors compact and homogeneous superpixels, while the balancing function encourages uniform superpixels. It achieves good segmentation performance with a relatively high computational cost. Minimum spanning tree is an alternative to graph cut. Graph Based Superpixels (GBS) (Felzenszwalb and Huttenlocher 2004) performs an agglomerative clustering such that each cluster is the minimum spanning tree of the constituent nodes. The generated superpixels adhere well to image boundaries. The clustering is very fast. However, the size and number of superpixels cannot be controlled explicitly, and the shapes of superpixels are arbitrary. The locality, compactness and uniformity are poor.

Boundary evolution is an alternative to graph partitioning for boundary determination. Two ways of evolution have been developed for superpixel segmentation. One is growing superpixels from some given centers, the other is adjusting some given boundaries. Turbopixel (TP) is a representative of the first category (Levinshtein et al. 2009). Originating from the given centers, Turbopixels grow with their boundaries evolving step by step. Boundary evolution is driven by geometric flow calculated from grown regions and the rest regions. In this framework, geodesic distance is introduced as a measure of structure and layout of superpixels, and centers are relocated to generate structure-sensitive superpixels (Wang et al. 2013). The main drawback of these methods is very high computational cost, restricting practical applications. Superpixel Extracted via Energy-Driven Sampling (SEEDS) falls into the second category (Van den Bergh et al. 2015). Starting from an initial partitioning, it adjusts the boundaries by exchanging pixels or blocks of neighboring superpixels. An objective function based on superpixel colors and boundary shapes is defined to favor coherent superpixels. A simple hill-climbing approach is employed to optimize the objective function efficiently. However, it may not converge

Fig. 1 Rooted spanning superpixels. Starting from the seed pixels, the superpixels grow up pixel by pixel until they cover the whole image. The initial state, intermediate states, and final state are shown respectively

from left to right. Their boundaries and constituting pixels are depicted as green lines and colored regions in the top and bottom images respectively (Color figure online)

to optimal segmentation in some cases, and it lacks adaptivity between compactness and adherence. By constructing an objective function based on boundary and topology preserving Markov Random Fields, Efficient Topology Preserving Segmentation (ETPS) achieves excellent performance (Yao et al. 2015). But it is not fast enough to support realtime applications.

Feature space analysis is another formulation for superpixel segmentation. Working in feature space, it determines a superpixel by finding its pixels instead of its boundary. There are two ways to achieve this goal. One is mode seeking, and the other is k-means clustering. Mean Shift (MS) and Quick Shift (QS) are two techniques for searching modes of a density function (Comaniciu and Meer 2002; Vedaldi and Soatto 2008). Once modes in the image feature space are found, pixels converging to same mode constitute one superpixel. They are slow. The size and number of superpixels cannot be controlled explicitly and the shapes of superpixels are usually irregular. Simple Linear Iterative Clustering (SLIC) is a constrained k-means clustering (Achanta et al. 2012), which searches pixels in a limited region instead of the whole image to generate compact superpixels efficiently. VCell is also built upon k-means clustering (Wang and Wang 2012). Edge-Weighted Centroidal Voronoi Tessellations is developed to constrain superpixel boundary. The weak point of k-means clustering is no guarantee of connectivity. SLIC relies on post-processing to repair connectivity whereas VCell relies on two special mechanisms to maintain connectivity. As an improved SLIC, Simple Non-Iterative Clustering (SNIC) updates clusters by incorporating pixels connected to the clusters (Achanta and Süsstrunk 2017). It is faster than SLIC since no iteration is involved. The k-means clustering is based on the color difference and spatial distance between pixels and cluster centers, which facilitates shape regularization. However, only mean value of the superpixels are referred in clustering, more cues can be utilized to improve performance.

Superpixel has been extended to supervoxel, which is a three-dimensional version and has many applications in volumetric image segmentation (Lucchi et al. 2012) and video preprocessing (Xu and Corso 2012). Some methods treat video as a series of images, and segment them frame by frame. The others treat video as volumetric image. An evaluation of supervoxel methods for video processing is presented in Xu and Corso (2012), where, Mean Shift (MS) (Comaniciu and Meer 2002), Graph Based (GB) (Felzenszwalb and Huttenlocher 2004), Hierarchical Graph Based (GBH) (Grundmann et al. 2010), and NC (Shi and Malik 2000), Segmentation by Weighted Aggregation (SWA) (Sharon et al. 2000) and Temporal Superpixels (TSP) (Chang et al. 2013) have been evaluated and compared.

1.2 Motivation and Contribution

The above formulations have both positive and negatives. For examples, GBS is fast but does not allow number and shape controlling, k-means clustering based methods allow such controlling but needs advanced measures for the differences between pixels and superpixel centers.

This paper adapts minimum spanning tree and proposes a formulation similar to image foresting transform (Falcão et al. 2004) to integrate the positives of different formulations. First, it adapts the underlying graph to represent vertex values instead of edge weights. Second, it selects some vertices to serve as roots of trees in spanning forest. Third, it introduces a path-cost function to measure both color similarity and spatial closeness between the seeds and the remaining pixels. Fourth, superpixel segmentation is formulated as searching a rooted spanning forest of the underlying graph with respect to the roots and path-cost function. Finally, a set of first-in-first-out (FIFO) queues are introduced to maintain the candidates to search the forest efficiently.

Based on this formulation, superpixel segmentation is achieved via region growing as depicted in Fig. 1. Starting

Fig. 2 Minimum spanning forest and rooted spanning forest. Vertices and edges are illustrated as circles and lines. Edge weights are indicated by their thickness, and vertex values are indicated by their colors. **b** Minimum spanning forest of the disconnected graph in **a**, **d** is a mini-

mum spanning forest of the connected graph in **c**, **f** is a rooted spanning tree of the connected subgraph (with green vertices) in **e**, **h** is a rooted spanning forest of the connected graph in **g**, the roots are illustrated as filled circles (Color figure online)

from the evenly distributed seeds, the superpixels are guided by a path-cost function to grow uniformly in homogeneous regions and adaptively when they touch object boundaries, the superpixels grow pixel by pixel until they cover the whole image. The number of superpixels is controlled by the number of seeds. Good performances are assured by connecting each pixel to the similar seed, which are dominated by the path-cost function. Benefiting from the FIFO queues, superpixels grow efficiently and the proposed algorithm is much faster than the other superpixel and supervoxel methods.

The rest of this paper is organized as follows: background and adaptation of rooted spanning forest are presented in Sect. 2, rooted spanning superpixels are formulated in Sect. 3, experiments with evaluations and comparisons are presented in Sect. 4, and conclusions are drawn in Sect. 5.

2 Rooted Spanning Forest

2.1 Minimum Spanning Forest

Our formulation is based on an *undirected graph*, which is called as a graph for simplicity. A *graph* is an ordered pair $G = (V_G, E_G)$, where V_G is a nonempty set and E is a set of unordered pairs of elements of V_G , i.e. $E_G \subseteq V_G \times V_G$. Elements of V_G and E_G are called the *vertices* and the *edges* respectively. Each edge $e = (s, t) \in E_G$ links two vertices $s \in V_G$ and $t \in V_G$. If two or more edges link the same two vertices, the edges are called *multiple edges*. If an edge links a vertex with itself, the edge is called a *loop*. A graph is *simple* if it has no loops and no multiple edges. A *weighted graph* associates a weight with every edge in the graph.

A *simple path* in a graph $G(V_G, E_G)$ is an alternating sequence of distinct vertices and edges $\pi = v_1 e_1 v_2 e_2 \dots v_{p-1} e_{p-1} v_p$, where $e_i = (v_i, v_{i+1}) \in E_G, i = 1, \dots, p-1$. v_1 and v_p are the *origin* and the *destination* of the path, and they are denoted as $v_1 = \text{org}(\pi)$ and $v_p = \text{dst}(\pi)$ respectively. When $p = 1$, the path is called a trivial path. The weight of a path is the sum of the weights of the traversed edges. A graph is *connected* if any two vertices are connected by one or more paths. If any two vertices are connected by

exactly one path, the graph is called a *tree*. The weight of a tree is the sum of the weights of all its edges. A *forest* is a disjoint union of trees.

A *spanning tree* of a connected graph is a tree that connects all the vertices and each edge of the tree is an edge of the underlying graph. One graph usually has many different spanning trees. A *minimum spanning tree* of a weighted connected graph is a spanning tree whose weight is no more than the weight of every other spanning tree. A *minimum spanning forest* of a weighted disconnected graph is a union of minimum spanning trees for its connected components. As demonstrated in Fig. 2b, the minimum spanning forest consists of three trees; each is a minimum spanning tree of one connected component of the disconnected graph in Fig. 2a.

Minimum spanning forest has been applied to image segmentation (Felzenszwalb and Huttenlocher 2004). However, the graph representing an image is a connected graph as in Fig. 2c rather than a disconnected one as in Fig. 2a. It is partitioned into some components by removing all edges whose weights are above a weight threshold. Fig. 2d is a minimum spanning forest of the graph in Fig. 2c based on a weight threshold indicated by the thick edges. This approach deals with the edge weights derived from the pixel values. These derived values may introduce some extra errors in segmentation. Moreover, the number of segments and their coverages are not controlled explicitly but depend on the thresholds implicitly.

2.2 Rooted Spanning Forest

This paper extends minimum spanning tree and forest to *rooted spanning tree and forest*. The underlying graph is adapted to represent the pixel colors instead of image edges. Some roots are introduced to control the number and shape of superpixels.

An intuitive picture of the above concepts is depicted in Fig. 2, where Fig. 2f, h are a rooted spanning tree and forest of the underlying graph Fig. 2e, g respectively. As indicated by the same thickness, no weights are assigned to the edges. Graph partitioning is based on the vertex values (colors).

2.2.1 Rooted Spanning Tree

Let $r \in V_G$ be a root, $f(*)$ be a path-cost function, i.e., $f(\pi)$ is the cost of a path π . A *rooted spanning tree* of a graph G with respect to r , $f(*)$ is a tree T such that:

- $V(T) = V_G$, $E(T) \subseteq E_G$;
- r is the root of T ;
- any root-originated path π in T meets one of the two conditions:
 1. $\pi = r$;
 2. $\pi = \tau ev$ subject to $f(\tau ev) \leq f(\tau' e'v)$ for any $\tau' \in \Pi$.

Where τev and $\tau' e'v$ are one-edge-extensions of path τ and path τ' respectively, Π is the set of existing root-originated paths.

2.2.2 Rooted Spanning Forest

Let $R = \{r_i | r_i \in V_G, i = 1, 2, \dots, K\}$ be a set of roots, $f(*)$ be a path-cost function. A *rooted spanning forest* F of a graph G with respect to R and $f(*)$ is an union of trees $F = T_1 \cup T_2 \cup \dots \cup T_K$ such that:

- $V(T_i) \subseteq V_G$, $E(T_i) \subseteq E_G$ for $i = 1, \dots, K$;
- $V(T_1) \cup \dots \cup V(T_K) = V_G$, $T_i \cap T_j = \emptyset$ for $i \neq j$;
- r_i is the root of T_i for $i = 1, \dots, K$;
- any root-originated path π in F meets one of the two conditions:
 1. $\pi = r$;
 2. $\pi = \tau ev$ subject to $f(\tau ev) \leq f(\tau' e'v)$ for any $\tau' \in \Pi$.

where τev and $\tau' e'v$ are one-edge-extensions of path τ and path τ' respectively, Π is the set of existing root-originated paths.

Rooted spanning tree is extended to rooted spanning forest by injecting multiple roots instead of a single root.

3 Rooted Spanning Superpixels

This paper formulates superpixel segmentation as finding a rooted spanning forest F of a graph G representing an image I with respect to a set of roots R and a path-cost function $f(*)$. Each tree $T_i \subset F$ represents one *rooted spanning superpixel (RSS)*. G can represent a volumetric image or a video, then each tree represents one *rooted spanning super-voxel*.

3.1 Implicit Graph

A graph $G(V, E)$ is employed to represent an image $I(p)$. Each vertex $v_p \in V$ represents a pixel p , and each edge $e_{p,q} = (v_p, v_q) \in E$ links a pair of neighboring pixels p and q . It is not necessary to explicitly store the edges since they are recoverable based on a given neighborhood system. Therefore, no explicit graph is constructed and the image is dealt with directly. Each pixel is treated as a vertex and it is linked to its 8 nearest neighbors based on the second-order neighborhood system employed in this paper. When the underlying graph represents a volumetric data, each vertex represents one voxel and it has 26 neighbors.

3.2 Roots Selection

A set of seed pixels are selected to serve as the roots of spanning forest to control the number and locality of superpixels. They are selected regularly and evenly on the image plane as shown by the initial state in Fig. 1. Given the expected number of superpixels K , the expected width of a superpixel is $w = \sqrt{N/K}$, where N is the total number of pixels. Seeds are selected by sampling the rows and columns with an interval of w . A schema similar to that of SLIC can be adopted to adjust the seeds within their 3×3 windows, however, no performance gains are found in our experiments.

3.3 Cost Function

A path-cost function $f(*)$ is developed to measure the color similarity and spatial closeness between a seed pixel and another pixel through a path. A general function for a path $\pi = v_1 e_1 v_2 \dots v_{p-1} e_{p-1} v_p$ is

$$f(\pi) = f(v_1, \dots, v_p), \quad (1)$$

where the edges are excluded from the variable list since they have no weights as declared in Sect. 2.2.

An existing path-cost function is the geodesic distance

$$f^g(\pi) = \sum_{i=2}^p \|I(v_i) - I(v_{i-1})\|_2 + \lambda \sum_{i=2}^p \|v_{i-1} v_i\|_2, \quad (2)$$

where $\|I(v_i) - I(v_{i-1})\|_2$ is the L_2 norm of color difference between two successive pixels, and $\|v_{i-1} v_i\|_2$ is their Euclidean distance. However, this color term fails to measure similarity between two ends of a path shown in Fig. 3a. The alternating values amount to a large geodesic distance for two ends of the same value.

As illustrated in Fig. 3b, this paper utilizes global characteristics of a path instead of sum of local ones to measure color similarity of path ends, and proposes two novel path-cost functions.

Without loss of generality, assume that the image has a single channel. Let $I(v)$ be the value of v , the maximal difference between the origin and rest pixels on path is calculated as

$$f^d(\pi) = \max_{1 \leq i \leq p} |I(v_1) - I(v_i)|, \quad (3)$$

and the range of values of all pixels on path is calculated as

$$f^r(\pi) = \max_{1 \leq i \leq p} I(v_i) - \min_{1 \leq i \leq p} I(v_i). \quad (4)$$

The maximal difference acts as a barrier between the origin and destination. It reflects the cost of going from origin to destination along the path. Similarly, the range of values also reflects such cost. They are more robust than geodesic distance since they avoid summing local derivatives. Moreover, they outperform a simple difference between origin and destination, which does not take the intermediate pixels into account.

For images of multiple channels, the pixel value is a vector, the differences in Eqs. 3 and 4 are replaced by their L_∞ norms. Since only comparisons are involved, it is quite efficient. Moreover, by experiments, it is found to outperform alternative norms such as L_2 norm, which need floating-point operations.

Spatial closeness is measured in the same way by treating pixel coordinates as extra channels:

$$\begin{cases} I_{M+1}(v) = \lambda \cdot x(v) \\ I_{M+2}(v) = \lambda \cdot y(v) \\ I_{M+3}(v) = \lambda' \cdot z(v) \end{cases}, \quad (5)$$

where $x(v)$, $y(v)$, $z(v)$ are three coordinates of v , I_{M+1} , I_{M+2} , I_{M+3} are three additional channels, λ and λ' are two scaling factors. $I_{M+3}(v)$ is employed only in supervoxel segmentation. Usually, the third dimension has different meaning (e.g. time in video), and it needs a different scaling factor λ' .

For each pixel, its color value is in $\{0, 1, \dots, 255\}$, but its coordinates can be large when the size of an image is very large. By normalizing pixel coordinates using the expected width of a superpixel $w = \sqrt{N/K}$, Eq. 5 is written as:

$$\begin{cases} I_{M+1}(v) = \lambda \cdot x(v) = \lambda w \cdot x(v)/w = \hat{\lambda} \cdot \hat{x}(v) \\ I_{M+2}(v) = \lambda \cdot y(v) = \lambda w \cdot y(v)/w = \hat{\lambda} \cdot \hat{y}(v) \\ I_{M+3}(v) = \lambda' \cdot z(v) = \lambda' w' \cdot z(v)/w' = \hat{\lambda}' \cdot \hat{z}(v) \end{cases}, \quad (6)$$

where $\hat{\lambda}$, $\hat{\lambda}'$ are two normalized scaling factors, $\hat{x}(v)$, $\hat{y}(v)$, $\hat{z}(v)$ are three normalized coordinates of v .

The unified cost functions for maximal difference and range of values are

$$f_\infty^d(\pi) = \max_{1 \leq i \leq p} \|I(v_1) - I(v_i)\|_\infty; \quad (7)$$

$$f_\infty^r(\pi) = \left\| \max_{1 \leq i \leq p} I(v_i) - \min_{1 \leq i \leq p} I(v_i) \right\|_\infty. \quad (8)$$

The cost is calculated efficiently by comparisons and subtractions. When the path π extend to a new pixel v_{p+1} , its maximal difference cost is calculated incrementally as

$$\begin{aligned} f_\infty^d(\pi') &= \max_{1 \leq i \leq p+1} \|I(v_1) - I(v_i)\|_\infty, \\ &= \max \left(\max_{1 \leq i \leq p} \|I(v_1) - I(v_i)\|_\infty, \|I(v_1) - I(v_{p+1})\|_\infty \right), \\ &= \max \left(f_\infty^d(\pi), \|I(v_1) - I(v_{p+1})\|_\infty \right). \end{aligned} \quad (9)$$

Similarly, the range of values can also be computed incrementally. The costs of all root-originated paths can be computed very efficiently based on the incremental computing.

3.4 Global Objective Function

The global objective function is defined as the sum of path-costs for all the vertices (pixels):

$$\sum_{v_p \in V} f(\pi(v_p)), \quad (10)$$

where $\pi(v_p)$ is a path connecting one root and v_p , $f(\pi)$ takes either $f_\infty^d(\pi)$ or $f_\infty^r(\pi)$. Unlike the variables in the existing objective functions for superpixel segmentation, the variables are paths. Therefore, a path connecting to a root denoting a superpixel need to be determined for each pixel. Based on the inductive definition in Sect. 2.2.2, the paths for all pixels must be determined progressively. In each step, the existing root-originated paths are extended one step to reach a new pixel v_p by:

$$\pi(v_p) = \tau^* e v_p \quad (11)$$

$$\tau^* = \arg \min_{\tau \in \Pi} f(\tau e v_p), \quad (12)$$

where $\tau e v_p$ is an one-edge-extension of τ to v_p , Π is the set of existing root-originated paths, $\tau \in \Pi$ means that τ is an existing root-originated path.

The root-originated paths for a graph with two roots illustrated in Fig. 3c and a path-cost function in Eq. 4 are found as follows: First, $\Pi = \{v_1, v_2\}$ as they are trivial. Second, since $f^r(v_1 e_1 v_2) = 1 < 2 = f^r(v_3 e_2 v_2)$, $\pi(v_2) = v_1 e_1 v_2$, $\Pi = \{v_1, v_2, v_1 e_1 v_2\}$. Third, $\pi(v_5) = \arg \min_{v_1 e_1 v_2 e_6 v_5} f(v_1 e_1 v_2 e_6 v_5) = v_1 e_1 v_2 e_6 v_5$, then $\Pi = \{v_1, v_2, v_1 e_1 v_2 e_6 v_5\}$. Fourth, $\pi(v_4) = \arg \min_{v_1 e_5 v_4} f(v_1 e_5 v_4) = v_1 e_5 v_4$, then Π is updated to be $\Pi = \{v_2, v_1 e_1 v_2 e_6 v_5, v_1 e_5 v_4\}$. Finally, $\pi(v_6) = \arg \min_{v_3 e_7 v_6} f(v_3 e_7 v_6) = v_3 e_7 v_6$.

Fig. 3 Path-cost function. **a** Depicts a path with alternating values, **b** two path-cost functions, where, d is the maximal difference between origin and the remaining pixels on path, and r is the range of values of

Without the requirement $\tau \in \Pi$, a path $\tau = v_3e_2v_2 \notin \Pi$ can be extended to v_5 such that $f(v_3e_2v_2e_6v_5) = 3 < 4 = f(v_1e_1v_2e_6v_5)$. In this case, $v_3e_2v_2e_6v_5$ has minimal cost but $v_3e_2v_2$ is not a root-originated path. With this contradiction, the paths with optimal costs do not form a forest at all. The algorithms in Falcão et al. (2004) cannot deal with this contradiction. Instead, they accept only monotonic-incremental and smooth path-cost functions. By enforcing $\tau \in \Pi$, the root-originated paths are extended step by step, and the above contradiction is resolved.

3.5 Rooted Spanning Superpixel Algorithm

The inductive definition of root-originated paths allows the rooted spanning forest to be found progressively in a manner as Dijkstra's algorithm (1959). In each step, it needs to sort all candidates to select the best one as Eq. 12. Although a balanced queue can be employed to store the candidates, the total time complexity of Dijkstra's algorithm is $O(m + n \log n)$, where m and n are the numbers of edges and vertices respectively.

The proposed solution is motivated by counting sort and bucket sort (Cormen et al. 2009). First, it divides the cost range into a set of equal-sized intervals (buckets) as bucket sort does. Second, it assigns an integer to each interval as required by counting sort. When only colors are considered, the proposed path-cost functions directly take integers in $\{0, 1, \dots, 255\}$. When spatial coordinates are considered, real intervals need to be quantized into integers as the path-cost functions may take nonnegative real values. This quantization has little influence on segmentation.

A FIFO queue ω_y is employed for a bucket to store the candidates whose costs are y . The candidates in one queue are served by FIFO schema, which resolves the sorting operation of bucket sort. The candidates in a queue ω_{y_1} are served before those in ω_{y_2} when $y_1 < y_2$. Since one and only one FIFO queue is employed for each bucket (integer), the order of queues is fixed. Such fixed order resolves the sorting operation of counting sort. Based on these queues, the

the pixels on path. **c** Illustrates a small graph, in which the vertex values are shown in the circles and the selected roots are gray vertices

Algorithm 1 Rooted Spanning Superpixels Algorithm

Require: $I, f(*), g(*), R = \{r_1, r_2, \dots, r_K\}$;
Ensure: L ;

- 1: set label $L(*)$ to zero for all pixels; {labels of pixels}
- 2: set state $\delta(*)$ to true for all pixels; {true/false: unlabelled/labelled}
- 3: set cost $\psi(*)$ to maximal value for all pixels; {path-costs of labeled pixels}
- 4: set cost $\phi(*)$ to maximal value for all pixels; {path-costs of candidates}
- 5: empty queue $\omega_{*,*}$ for all costs and groups; {store the candidates}
- 6: **for** $k = 1$ **to** K **do**
- 7: $x \leftarrow g(r_k)$; {get its group index}
- 8: pushback($\omega_{0,x}, r_k$); {push it into a corresponding queue}
- 9: $L(r_k) \leftarrow k$; {assign a label to it}
- 10: $\psi(r_k) \leftarrow 0$; {record its path-cost}
- 11: $\phi(r_k) \leftarrow 0$; {record its path-cost}
- 12: **end for**
- 13: **for** $y = 0$ **to** Y **do**
- 14: **for** $x = 0$ **to** X **do**
- 15: **while** $\omega_{y,x} \neq \emptyset$ **do**
- 16: $p \leftarrow \text{popfront}(\omega_{y,x})$; {pop the first pixel in the queue}
- 17: **if** $\delta(p)$ **then**
- 18: $\delta(p) \leftarrow \text{false}$; {mark as labeled}
- 19: $k \leftarrow L(p)$; {get its label}
- 20: **for all** $i = 0$ **to** 8 **do**
- 21: $q \leftarrow \text{neig}(p, i)$; {get its i -th neighbor}
- 22: $t \leftarrow \max(\psi(p), \|I(q) - I(r_k)\|_\infty)$; {calculate its cost}
- 23: **if** $\delta(q)$ **and** $\phi(q) > t$ **then**
- 24: $L(q) \leftarrow k$; {propagate the label to it}
- 25: $\psi(q) \leftarrow t$; {update its path-cost}
- 26: $\phi(q) \leftarrow t$; {update its path-cost}
- 27: pushback($\omega_{\phi(q),x}, q$); {push it into a corresponding queue}
- 28: **end if**
- 29: **end for**
- 30: **end if**
- 31: **end while**
- 32: **end for**
- 33: **end for**
- 34: **return** L .

root-originated paths extend very efficiently since no sorting operation is needed. Since candidates with lower costs are served before those with higher costs, the existing root-originated paths always extend to their most similar pixels.

Fig. 4 Path extending based on groups of queues. $\omega_{y,x}$ is a queue for cost y and group x . The one-step-extension of paths in different groups are searched simultaneously. However, they are synchronized by the path-cost indicated by the dashed lines. Only after all paths with same cost are found, can it start to search paths with higher costs

In other words, all pixels are connected to their most similar seeds.

Multiple paths originated from different roots can extend simultaneously. Without loss of generality, let the seeds be partitioned into some groups indexed by $0, 1, \dots, X$ and the costs be quantized into $0, 1, \dots, Y$. Then, the number of seeds in x -th group is denoted as K_x , the queue for cost y and x -th group is denoted as $\omega_{y,x}$. Path extending in the same group is carried out in serial while those in different groups can be carried out in parallel. As shown in Fig. 4, the simultaneous extending is synchronized by the path-cost. Such synchronization guarantees fair competition among different groups and assures coherent superpixels.

The seeds can be put into either one group or K groups. The former results in a serial algorithm. The latter requires a huge memory for the $K * Y$ queues and is not useful in practice. Since 4 or 8 CPU cores are usually available in a personal computer, it is natural to partition the seeds into 4 or 8 groups, each being processed by one CPU core. A simple way is to group the seeds according to the 4 quadrants separating along central row and central column of the image. The key to achieve maximum speedup is the balanced groups in terms of the number of seeds.

Algorithm 1 describes the *Rooted Spanning Superpixels* (RSS) Algorithm based on the maximal difference. In this algorithm, line 1–5 initialize all variables, loop 6–12 labels the seed pixels, loop 13–33 propagates the labels to all the remaining pixels. It seems that the complexity depends on X and Y , however, many queues are usually empty. Actually, every pixel is labeled once and only once. It means that the complexity is $O(N)$, where N is the total number of pixels. Therefore, the computation time is stable with respect to X , Y and K . The inner loop 14–32 can be carried out either in serial or in parallel. To adopt the range of values, it is necessary to record the maximum/minimum of each channel

in $\psi(*)$ and revise line 22 correspondingly. For supervoxels segmentation, line 20 need to loop over all 26 neighbors.

As in Fig. 1, different labels indicated by different colors are assigned to different seeds, they propagate pixel by pixel as the root-originated paths extend step by step, until all pixels are labeled. In a homogeneous region, all candidates have the same color as seeds and have a zero cost. The path extending and superpixel growing depends on the order of candidates pushed into the queue. It assures superpixels the uniform growing in homogeneous regions since the pixels close to seeds are served before the pixels far away. In a non-homogeneous region, they have different colors and costs. Pixels similar to seeds have lower costs and receive a priority. Therefore, they grow adaptively after they touch object boundaries. When the superpixel growing is parallelized, the order of service for candidates of the same cost may change. However, the final superpixels are influenced only by the boundary pixels in two narrow strips along the central row and central column respectively. Only a few of these pixels have the same cost with respect to neighboring seeds in different groups. Therefore, parallelization has little influence on the final superpixels.

RSS is scalable as it is capable to deal with different kinds of data in a unified way. On one hand, each pixel can take a gray value, a RGB vector, a RGBD vector, or even a vector of features. On the other hand, the data can be a 2-dimensional image, a 3-dimensional volumetric data, or a video.

4 Experimental Results

This section analyzes the effects of path-cost functions and scaling factor, and then compares RSS algorithm with five state-of-the-art superpixel methods using the superpixel benchmark¹ (Stutz et al. 2018), which employs the Berkeley Segmentation Dataset 500 (BSDS) (Arbelaez et al. 2011), Fashionista dataset (Fash) (Yamaguchi et al. 2012), NYU Depth Dataset V2 (NYU) (Silberman et al. 2012), Stanford Background Dataset (SBD) (Gould et al. 2009) and Sun RGB-D dataset (SUN) (Song et al. 2015). All codes for the methods run on a laptop with Intel Xeon(R) CPU E3-1575M @ 3.00GHz $\times 8$ to segment the images.

Boundary Recall (Rec), Undersegmentation Error (UE), Explained Variation (EV) and Compactness (CO) are employed to measure segmentation performance. Rec measures superpixels' adherence to boundary by counting the coincidence between superpixel boundary and ground truth boundary. UE measures segmentation accuracy by calculating the fraction of superpixels leaking ground truth boundary. EV measures superpixels' coherence by computing the proportion of image variation that is explained

¹ <https://github.com/davidstutz/superpixel-benchmark>.

when superpixels are compressed as units of representation. CO measures the compactness of superpixels. Average Miss Rate (AMR), Average Undersegmentation Error (AUE) and Average Unexplained Variation (AUV) are calculated by averaging 1-Rec, UE and 1-EV over a set of $K \in [K_{\min}, K_{\max}]$, and they are employed to summarize the algorithm performance. Algorithm ranking is based on the sum of AMR and AUE. Please refer Stutz et al. (2018) for the details.

RSS algorithm is also compared with state-of-the-art supervoxel methods using the supervoxel benchmark² (Xu and Corso 2016). The supervoxels generated by the top performing methods are available in the benchmark, they are employed in the comparison directly. To demonstrate its scalability, RSS treats a video as 3-dimensional volumetric data and segments it as a whole instead of frame by frame.

Boundary Recall Distance (BRD), 3D Undersegmentation Error (UE3D), Explained Variation (EV), 3D Segmentation Accuracy (SA3D), Mean Size Variation (MSV) and Temporal Extent (TEX) are employed to evaluate the segmentation performances. The former three correspond to Rec, UE and EV, but BRD and UE3D are based on different versions. SA3D indicates the achievable segmentation accuracy and it is correlated to UE. TEX measures the average temporal extent of supervoxels. MSV measures the size variation of supervoxels along temporal axes. Please refer Xu and Corso (2016) for the details.

The difference between superpixels by serial RSS and parallel RSS is ignorable, and their performance curves overlap with each other. Serial RSS is employed for comparison as the other methods are not parallel.

4.1 Path-Cost Function

Different path-cost functions can be adopted to define root-originated paths and to generate RSS. This experiment compares five cost functions: $f^g(*)$ (Eq. 2), $f_\infty^d(*)$ (Eq. 7), $f_\infty^r(*)$ (Eq. 8), $f_2^d(*)$ and $f_2^r(*)$, where, $f_2^d(*)$, $f_2^r(*)$ are based on L_2 instead of L_∞ norm.

The bare comparison is based on pixel colors but not pixel coordinates since the latter depend on a scaling factor λ whose effects will be analyzed in the following section. The overall performances on BSDS dataset are reported in Table 1. The geodesic distance is outperformed by the other four functions as lower AMR, AUE and AUV are better. Since the geodesic distance is a sum of derivatives, it is sensitive to local variations and noises. The other four functions are based on global characteristics, they overcome this shortcoming and significantly improve the performance.

One can observe that $f^d(*)$ leads to lower AMR, lower AUV and higher AUE than $f^r(*)$ does. It means that bet-

Table 1 Overall performances on BSDS dataset for RSS based on different path-cost functions

	AMR	AUE	AUV	AMR + AUE
$f_\infty^d(*)$	3.88974	7.39789	7.17524	11.28763
$f_2^d(*)$	3.78788	7.60433	6.97447	11.39221
$f_\infty^r(*)$	4.62260	7.17840	7.34485	11.80100
$f_2^r(*)$	4.88122	7.25717	7.22384	12.13839
$f^g(*)$	8.76662	7.28618	9.49756	16.05280

The performances are measured by Average Miss Rate (AMR), Average Undersegmentation Error (AUE), Average Unexplained Variation (AUV) and AMR + AUE

ter boundary adherence and coherence is achieved by $f^d(*)$ while higher segmentation accuracy is achieved by $f^r(*)$. These two functions compensate for each other, however, $f^d(*)$ outperforms $f^r(*)$ for all five dataset except for Fash according to the sum of AMR and AUE.

Moreover, $f_\infty^d(*)$ and $f_\infty^r(*)$ perform a little better than $f_2^d(*)$ and $f_2^r(*)$ respectively except for EV, which is based on L_2 norm. Red and white are as similar as red and black under L_∞ norm. In contrast, red is similar to black more than white under L_2 norm since it averages the differences in three channels. It indicates that more coherent superpixels are assured by L_∞ . Another advantage of L_∞ norm is its computational cost. The L_∞ norm only needs integer comparisons while the L_2 norm needs floating-point computations. $f_\infty^d(*)$ is employed in the rest experiments.

4.2 Balancing Factor

As shown in Fig. 5, the shapes of superpixels can be regularized by the scaling factor $\hat{\lambda}$ in Eq. 6. This factor balances the color similarity and spatial closeness. Given the expected number of superpixels K , $\hat{\lambda}$ is the only parameter to be specified for RSS algorithm.

When $\hat{\lambda} = 0$, superpixel growing depends only on pixel colors and is sensitive to noises. The superpixels have irregular shapes. When $\hat{\lambda} > 0$, superpixel growing depends on both colors and distances, superpixels are regularized to have regular shapes. As $\hat{\lambda}$ increases, the shapes become more regular. When $\hat{\lambda} \geq 512$, the superpixels are squares as superimposed on rightmost image. The maximal normalized coordinate difference between a pixels v at a square's borders and its center v_c is 0.5, i.e., $\max(|\hat{x}(v) - \hat{x}(v_c)|, |\hat{y}(v) - \hat{y}(v_c)|) = 0.5$. The normalized coordinate difference between all the pixels outside a square and the center of this square are larger than 0.5. By multiplying a $\hat{\lambda} \geq 512$, their coordinate differences are larger than 256, which is the maximum value for color difference. Since the maximal difference is dominated by the pixel coordinates, the generated superpixels are squares.

² <http://www.cs.rochester.edu/~cxu22/d/libsvx/>.

Fig. 5 Shape regularization. The original image and superpixels generated using $\hat{\lambda} = 0, 10, 512$ are depicted respectively from left to right

Fig. 6 Performances based on different scaling factors. Rec, UE, EV and CO vary with respect to $\hat{\lambda}$ and K , which are indicated by the horizontal axis and line colors respectively

Figure 6 reports Rec, UE, EV and CO of superpixels on BSDS dataset. The segmentations are carried out on a coarse, middle and fine level based on $K = 200, 1200, 6000$ respectively, and each is based on $\hat{\lambda} = 0, 1, \dots, 20$. As shown, finer segmentation outperforms coarser segmentation as blue lines are better than green lines and green lines are better than red lines. However, for a fixed K , $\hat{\lambda}$ has minor impacts on segmentation performance especially for fine segmentation. As $\hat{\lambda}$ increases, UE is stable while Rec and EV decrease slowly when $\hat{\lambda} \in [0, 5]$. As indicated by their compactnesses, superpixel shapes are regularized by $\hat{\lambda}$ significantly.

The overall segmentation performance gets better as K increases. Given a K , it gets worse as $\hat{\lambda}$ increases since coherency and adherence receive less priority to uniformity and compactness. As performance is improved with the cost of increasing K , it is important to choose a well-balanced K . Given a K , it is convenient to select an $\hat{\lambda} \in [0, 5]$. A smaller $\hat{\lambda}$ gives segmentation performance more priority to superpixel regularity. However, a large $\hat{\lambda}$ can be employed if uniform and compact superpixels are preferred.

4.3 Comparison with State-of-the-Art Superpixel Methods

ERS (Liu et al. 2011), ETPS (Yao et al. 2015), SEEDS (Van den Bergh et al. 2015), SNIC (Achanta and Süsstrunk 2017) and GBS (Felzenszwalb and Huttenlocher 2004) are employed for comparison. The former three are TOP 3 algorithms reported in Stutz et al. (2018). SNIC is an improved SLIC. Both SLIC and GBS are used very widely. Typical superpixels produced by these methods are demonstrated in Fig. 7.

GBS is based on minimum spanning forest of a graph whose edge weights measure the similarities of neighboring pixels. The segmentation is controlled by a threshold for the edge weights and a threshold for the segment sizes. It does not allow to control the number of superpixels directly, and it generates some widespread superpixels of arbitrary shapes. By introducing the roots of forest into the underlying graph, the number of superpixels can be easily controlled by RSS. Coupling the regularly and uniformly distributed seeds with the path-cost function, the manner of path extending and superpixel growing facilitates region competition and assures superpixels the expected characteristics as demonstrated by the examples. Each RSS covers a local, connected and coherent region. They adhere to the boundaries of flowers, face, shoulder, arms, white and black strips of shirt. Meanwhile, they are neither too large nor too small, and they appear to be compact or even square in homogeneous regions. They allow further regularization by a scaling factor.

As a method based on boundary evolution, SEEDS over-segments near object boundaries while under-segments in the homogeneous regions. ETPS and ERS overcome this shortcoming to some extent. However, noisy superpixel boundaries can still be observed in homogeneous areas. By clustering based on color and distance, SNIC produces compact superpixels with clean boundaries at the cost of sacrificing some segmentation performances. As shown in the last row, some white and black strips of the shirt are mixed in one superpixel. The advantages of RSS over SNIC is that superpixels grow along optimal paths, which assures better coherence and adherence.

The above methods are evaluated and ranked based on all five datasets. The optimal parameters for all methods

Fig. 7 Visual comparison of different superpixels. Two images in the BSDS dataset are segmented respectively into about 200 superpixels. Superpixel boundaries are depicted as green lines on the original images.

Two images and their zoomed windows indicated by white rectangles are depicted row by row. The superpixels by different methods are shown column by column (Color figure online)

Table 2 Overall performances of superpixel methods

	Rank Avg.	BSDS	Fash	NYU	SBD	SUN	AMR	AUE	AMR + AUE
ETPS	1	1	1	1	1	1	1.91	5.94	7.85
RSS	2.8	2	3	3	4	2	2.23	6.56	8.79
ERS	3.0	4	4	2	3	3	3.07	6.16	9.23
SEEDS	3.8	3	5	5	2	4	1.14	8.47	9.61
SNIC	4.2	5	2	4	5	5	3.67	6.32	9.99
GBS	6.0	6	6	6	6	6	8.31	8.43	16.74

The performances are measured by Average Miss Rate (AMR), Average Undersegmentation Error (AUE) and AMR + AUE. The first column list the superpixel methods. The metrics averaged over 5 datasets are listed in the right 3 columns. The ranks for 5 datasets and their average are listed in the middle 6 columns

Fig. 8 Performance comparison of superpixel methods on the BSDS dataset. It reports BR, UE, EV, their minimums/maximums and standard deviations, standard deviation of superpixel numbers and CO with respect to the number of superpixels

Fig. 9 Efficiency comparison of superpixel methods. The total time consumed by each method to segment all the images in each dataset is divided by the number of images in the dataset to calculate the averaged runtime in seconds for each method and each dataset. The last

column lists for each method the runtime averaged over 5 datasets. The figure depicts the averaged runtime in seconds to generate the expected number of superpixels for a NYU image

are selected by the optimization schema employed in the benchmark. Firstly, for each method and each dataset, AMR, AUE and AUV based on thirteen $K \in [200, 6000]$ are computed. Secondly, for each dataset, all six methods are ranked according to their AMR + AUE. Thirdly, for each method, an average rank is calculated by averaging its ranks over all five datasets. Fourthly, for each method, the average AMR, AUE, and AMR + AUE are also calculated by averaging them over all datasets. Finally, their overall performances are sorted based on their average ranks. As reported in Table 2, it results in an order same as that based on average AMR + AUE. According to the ranking, RSS outperforms all the methods except for ETPS.

The detailed reports for BSDS dataset are presented in Fig. 8. Rec, UE and EV along with their minimums/maximums and standard deviations are presented. In addition, CO and standard deviation of superpixel numbers are also presented. Four facts can be observed from these reports. First is the balance among metrics. Rec, UE and EV are well-balanced by RSS as all these metrics are ranked as the second or third among six. In contrast, SEEDS does not assure a good balance since it has the best Rec and almost the worst UE. The second concerns stability. RSS performs stably from coarse to fine segmentation as indicated by its smooth curves. It performs more stable than ETPS and SNIC. GBS and SEEDS perform unstably as their curves vary sharply. The third is that RSS generates more compact superpixels than SEEDS and ETPS. The last concerns the number of superpixels. Given an expected number, RSS generates a fixed number of superpixels for all images of the same size since it is fixed in segmentation. SNIC and ERS also guarantee a fixed number. But SEEDS and ETPS generate different number of superpixels for different images, which lead to nonzero std K. The number by GBS varies significantly since it depends on the other parameters indirectly.

RSS algorithm (serial version) is much faster than the other methods as depicted in Fig. 9. The time consumed by each method to segment all the images in each dataset is divided by the number of images to get the averaged time to segment one image of each dataset. For each method,

the above averaged runtime is further averaged over all five datasets and presented in the last column. Details for the NYU dataset are illustrated in the figure, where the runtime with respect to the number of superpixels is depicted. As shown, the runtime of RSS is stable with respect to varying number of superpixels while those of ERS, ETPS, SEEDS and GBS are not.

The above reports are based on the optimal parameters including the number of iterations for ETPS and SEEDS. By reducing it to one, the averaged runtime over all five datasets by either SEEDS or ETPS is 0.102 s. It is still much slower than RSS. Besides, it significantly impairs SEEDS's performance. Moreover, 40% computation time can be saved when 4 CPU cores are employed to parallelize RSS.

4.4 Comparison with State-of-the-Art Supervoxel Methods

GBH (Grundmann et al. 2010), SWA (Corso et al. 2008), TSP (Chang et al. 2013) and GBS (Felzenszwalb and Huttenlocher 2004) are employed for comparison. The former three are top performing methods reported in (Xu and Corso 2016). GBH is based on GBS, which is called as GB in this benchmark. A same scaling factor is employed for both spatial and temporal dimensions, i.e. $\hat{\lambda}' = \hat{\lambda} = 3$. Their supervoxels on six successive frames in the ice video of the Chen xiph.org dataset are demonstrated in Fig. 10.

One can observe the adaptivity of RSS in Fig. 10. On one hand, RSS are uniform and compact in the homogeneous regions. On the other hand, they adhere to object boundaries and indicate the peoples' outlines. GBS and GBH segment the videos into a set of nonuniform and irregular volumes, some cover large homogeneous areas while some others contain only a few boundary pixels. Although TSP are uniform and compact, their adherence to object boundaries are not good enough to indicate the peoples' outlines. The overall appearance of supervoxels by SWA is similar to RSS.

Figure 11 reports their performances. RSS outperforms the other methods according to three metrics for segmentation performance. It achieves the best BRD (lower is better)

Fig. 10 Visual comparison of different supervoxels. The 80 frames ice video in the Chen xiph.org dataset is segmented into about 2000 supervoxels. Different supervoxels are depicted as different colors. Six

successive frames are shown from top to bottom. The original image, and supervoxels generated by GBS, GBH, SWA, TSP and RSS are shown from left to right respectively (Color figure online)

and EV. Its UE3D is good, but its SA3D is low. It seems to contradict the correlation between SA3D and UE3D. Actually, SA3D also depends on the temporal dimension. When a supervoxel covers more frames, the size of correct segment increases and a higher accuracy is achieved. The other temporal metrics TEX and MSV are not good. Better SA3D, TEX and MSV can be achieved when the scaling factors $\hat{\lambda}$, $\hat{\lambda}'$ are optimized for the video. A better solution is to treat a video as a sequence of images instead of a volumetric data.

As demonstrated in Fig. 9, RSS is three times faster than GBS, which is almost an order of magnitude faster than GBH, SWA and TSP as reported in Xu and Corso (2016).

5 Conclusion

This paper proposes a new approach for superpixel segmentation. An image is represented as a graph, some regularly and

Fig. 11 Performance comparison of superpixel methods on the Chen xiph.org dataset. It reports BRD, UE3D, EV, SA3D, TEX and MSV with respect to the number of supervoxels

evenly distributed seed pixels serve as the roots, the maximal difference and range of values are proposed as path-cost functions to measure both the color similarity and spatial closeness between the seeds and rest pixels via some paths, superpixel segmentation is formulated as searching an rooted spanning forest of the graph with respect to the roots and cost function, and it is achieved by extending the root-originated path pixel by pixel.

The new formulation integrates some positives of different formulations and achieves a good balance among the expected characteristics: the number and locality of superpixels are controlled by the seeds; the coherency, adherence and adaptivity are assured by root-originated path extending dominated by a cost function defined on paths; the uniformity and compactness are regularized by a scaling factor; and connectivity is maintained by the region growing. Its segmentation performance is ranked as the second among state-of-the-art superpixel methods. Moreover, it is the fastest algorithm and allows parallelization. Finally, RSS algorithm is scalable as it deal with different kinds of data in the same way.

Its advantages are reinforced by some potential applications. RSS can be applied to various fields reviewed in the introduction. In addition, it is able to find a region for a given seed and a cost threshold, the regions based on different seeds and cost thresholds can serve as global object

proposals, which is especially useful for network extraction. Moreover, in light of recent work such as Tu et al. (2018), RSS promises new applications in CNN-based semantic segmentation as it is capable to deal with high dimensional features extracted by CNN.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2274–2282.
- Achanta, R., & Süsstrunk, S. (2017). Superpixels and polygons using simple non-iterative clustering. In *2017 IEEE conference on com-*

- puter vision and pattern recognition (CVPR) (pp. 4895–4904). IEEE.
- Alexe, B., Deselaers, T., & Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11), 2189–2202.
- Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5), 898–916.
- Arnab, A., Jayasumana, S., Zheng, S., & Torr, P. H. (2016). Higher order conditional random fields in deep neural networks. In *European conference on computer vision* (pp. 524–540). Berlin: Springer.
- Boix, X., Gonfau, J. M., Van de Weijer, J., Bagdanov, A. D., Serrat, J., & González, J. (2012). Harmony potentials. *International Journal of Computer Vision*, 96(1), 83–102.
- Chai, D. (2019). SQL: Superpixels via quaternary labeling. *Pattern Recognition*, 92, 52–63.
- Chang, J., Wei, D., & Fisher, J. W. (2013). A video representation using temporal superpixels. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2051–2058).
- Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603–619.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. Cambridge: MIT Press.
- Corso, J. J., Sharon, E., Dube, S., El-Saden, S., Sinha, U., & Yuille, A. (2008). Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *IEEE Transactions on Medical Imaging*, 27(5), 629–640.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.
- Falcão, A. X., Stolfi, J., & de Alencar, L. R. (2004). The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1), 19–29.
- Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2012). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1915–1929.
- Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 167–181.
- Gadde, R., Jampani, V., Kiefel, M., Kappler, D., & Gehler, P. V. (2016). Superpixel convolutional networks using bilateral inceptions. In *European conference on computer vision* (pp. 597–613). Berlin: Springer.
- Girshick, R. (2015). Fast R-CNN. In *2015 IEEE international conference on computer vision (ICCV)* (pp. 1440–1448). IEEE.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).
- Gould, S., Fulton, R., & Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *2009 IEEE 12th international conference on computer vision* (pp. 1–8). IEEE.
- Grundmann, M., Kwatra, V., Han, M., & Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *2010 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2141–2148). IEEE.
- Guney, F., & Geiger, A. (2015). Displets: Resolving stereo ambiguities using object knowledge. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4165–4175).
- He, S., Lau, R. W., Liu, W., Huang, Z., & Yang, Q. (2015). Supercnn: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3), 330–344.
- Hosang, J., Benenson, R., Dollár, P., & Schiele, B. (2015). What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4), 814–830.
- Levinshstein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., & Siddiqi, K. (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12), 2290–2297.
- Liu, M. Y., Tuzel, O., Ramalingam, S., & Chellappa, R. (2011). Entropy rate superpixel segmentation. In *2011 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2097–2104). IEEE.
- Liu, Y., Jiang, P. T., Petrosyan, V., Li, S. J., Bian, J., Zhang, L., et al. (2018). DEL: Deep embedding learning for efficient image segmentation. In *IJCAI* (pp. 864–870).
- Lucchi, A., Smith, K., Achanta, R., Knott, G., & Fua, P. (2012). Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 31(2), 474–486.
- Mičušik, B., & Koščeká, J. (2010). Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision*, 89(1), 106–119.
- Moore, A. P., Prince, S. J., & Warrell, J. (2010). “lattice cut”-constructing superpixels using layer constraints. In *2010 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2117–2124). IEEE.
- Moore, A. P., Prince, S. J., Warrell, J., Mohammed, U., & Jones, G. (2008). Superpixel lattices. In *IEEE conference on computer vision and pattern recognition, 2008. CVPR 2008* (pp. 1–8). IEEE.
- Mostajabi, M., Yadollahpour, P., & Shakhnarovich, G. (2015). Feed-forward semantic segmentation with zoom-out features. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3376–3385).
- Ren, X., & Malik, J. (2003). Learning a classification model for segmentation. In *Null* (p. 10). IEEE.
- Sharon, E., Brandt, A., & Basri, R. (2000). Fast multiscale image segmentation. In *IEEE conference on computer vision and pattern recognition, 2000. Proceedings* (Vol. 1, pp. 70–77). IEEE.
- Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905.
- Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGB-D images. In *European conference on computer vision* (pp. 746–760). Berlin: Springer.
- Song, S., Lichtenberg, S. P., & Xiao, J. (2015). Sun RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 567–576).
- Stutz, D., Hermans, A., & Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166, 1–27.
- Tsai, Y. H., Yang, M. H., & Black, M. J. (2016). Video segmentation via object flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3899–3908).
- Tu, W. C., Liu, M. Y., Jampani, V., Sun, D., Chien, S. Y., Yang, M. H., et al. (2018). Learning superpixels with segmentation-aware affinity loss. In *IEEE conference on computer vision and pattern recognition (CVPR)*.
- Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2), 154–171.
- Van den Bergh, M., Boix, X., Roig, G., & Van Gool, L. (2015). Seeds: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3), 298–314.
- Vedaldi, A., & Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In *European conference on computer vision* (pp. 705–718). Berlin: Springer.
- Veksler, O., Boykov, Y., & Mehrani, P. (2010). Superpixels and supervoxels in an energy optimization framework. In *European conference on computer vision* (pp. 211–224). Berlin: Springer.

- Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 6, 583–598.
- Wang, J., & Wang, X. (2012). VCells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6), 1241–1247.
- Wang, P., Zeng, G., Gan, R., Wang, J., & Zha, H. (2013). Structure-sensitive superpixels via geodesic distance. *International Journal of Computer Vision*, 103(1), 1–21.
- Wang, S., Lu, H., Yang, F., & Yang, M. H. (2011). Superpixel tracking. In *2011 IEEE international conference on computer Vision (ICCV)* (pp. 1323–1330). IEEE.
- Wertheimer, M. (1938). Laws of organization in perceptual forms. In *A source book of Gestalt psychology* (pp. 71–88).
- Xu, C., & Corso, J. J. (2012). Evaluation of super-voxel methods for early video processing. In *2012 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1202–1209). IEEE.
- Xu, C., & Corso, J. J. (2016). Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 119(3), 272–290.
- Yamaguchi, K., Kiapour, M. H., Ortiz, L. E., & Berg, T. L. (2012). Parsing clothing in fashion photographs. In: *2012 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3570–3577). IEEE.
- Yao, J., Boben, M., Fidler, S., & Urtasun, R. (2015). Real-time coarse-to-fine topologically preserving segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2947–2955).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.