Fast Computation of Content-Sensitive Superpixels and Supervoxels using q-distances

Zipeng Ye[1]*, Ran Yi[1]*, Minjing Yu[2]†, Yong-Jin Liu[1]†, Ying He[3]
[1]Tsinghua University [2]Tianjin University [3]Nanyang Technological University

Abstract

Many computer vision tasks benefit from superpixels/supervoxels, which can effectively reduce the complexity of input images and videos. To compute content-sensitive superpixels/supervoxels, the recent approaches represent the input image or video as a low-dimensional manifold and compute geodesic centroidal Voronoi tessellation (GCVT) on them. Although they can produce high-quality results, these methods are slow due to frequent query of geodesic distances which are computationally expensive. In this paper, we propose a novel approach that not only computes superpixels with quality better than the state-of-the-art, but also runs 6-8 times faster on benchmark dataset. Our method is based on a fast queue-based graph distance (called q-distance) and works for both images and videos. It has an optimal approximation ratio $O(1)$ and a linear time complexity $O(N)$ for N-pixel images or N-voxel videos. A thorough evaluation of 31 superpixel methods on five image datasets and 8 supervoxel methods on four video datasets shows that our method provides an all-in-one solution and consistently performs well under a variety of metrics. We also demonstrate our method on the applications of optimal image and video closure, and foreground propagation.

1. Introduction

Superpixels group similar pixels into atomic regions that can effectively capture low-level features in an image. Similarly, supervoxels are perceptually meaningful atomic regions in a video. Replacing the high amount of pixels/voxels by a moderate number of superpixels/supervoxels (collectively referred to as *superatoms* in this paper) can greatly reduce the complexities of many computer vision algorithms, e.g., saliency detection [19], foreground segmentation [22], 3D reconstruction [4] and scene understanding [18], etc.

As a special over-segmentation in image/video, superatoms — to be perceptually meaningful — should reflect "regularities of nature" [35]. Some commonly used criteria are: (1) *compactness*: the shape of superatoms is regular and thus the neighboring relations among superatoms are also regular; (2) *connectivity*: each superatom is simply connected[1]; (3) *high performance*: superatoms well preserve image/video boundaries and their computation is fast, memory efficient and scalable; (4) *parsimony*: the high performance is achieved with as few superatoms as possible; and (5) *ease of use*: users simply specify the number of superatoms and do not need to tune any other parameters.

1.1. Related work

A large body of superatom generation methods has been proposed and they can be broadly classified into two classes: (1) the traditional approaches with artificially designed features and (2) the deep learning based approaches. Diverse strategies have been applied in the first classes, e.g., graph partitioning [14], clustering [1], contour evolution [23], lattice-based energy optimization [11], and other hierarchical, generative and statistic methods [39, 45]. The second class was typified by two recent works [40, 21]. However, none of the existing methods satisfy all the above-mentioned criteria.

Some recent methods [6, 25, 26, 41, 46] focus on the *parsimony* principle and compute *content-sensitive superatoms* (CSS), which are small in content-dense regions (where the variation of intensity or color or motion is high) and large in content-sparse regions, and thus shed some light on offering a good balance among all other criteria (see Figures 1 and 2). Among the existing CSS methods, two recent approaches [26, 46] (summarized in Section 2) model the input images and videos as low-dimensional manifolds embedded in high-dimensional feature space, and then generate CSS by computing a uniform tessellation — e.g., geodesic centroidal Voronoi tessellation (GCVT) — on them. Although GCVT can produce high-quality CSS, computing it is time consuming, since geodesic distances are computationally expensive to obtain.

*Joint first authors
†Corresponding authors

[1]A region is *simply connected* if any simple closed curve/surface in it can be continuously shrunk into a point without leaving the region.

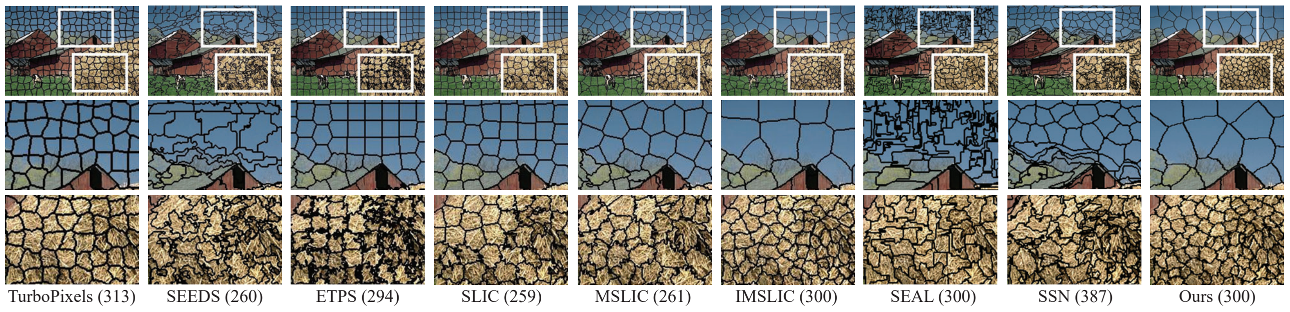


Figure 1. Visual comparison of our method and 8 representative superpixel methods: TurboPixels [23], SEEDS [11], ETPS [45], SLIC [1], MSLIC [25], IMSLIC [26], SEAL [40], SSN [21] and ours. 300 superpixels are specified by the user and the actual numbers of superpixels generated are in parentheses. Only IMSLIC, SEAL and ours allow exact control of the number of superpixels, but our method runs 6-8 times faster than IMSLIC and 4-5 times faster than SEAL. Our method also performs well in terms of under-segmentation error, boundary recall and compactness, and apply to both images and videos. See Section 5 for details.

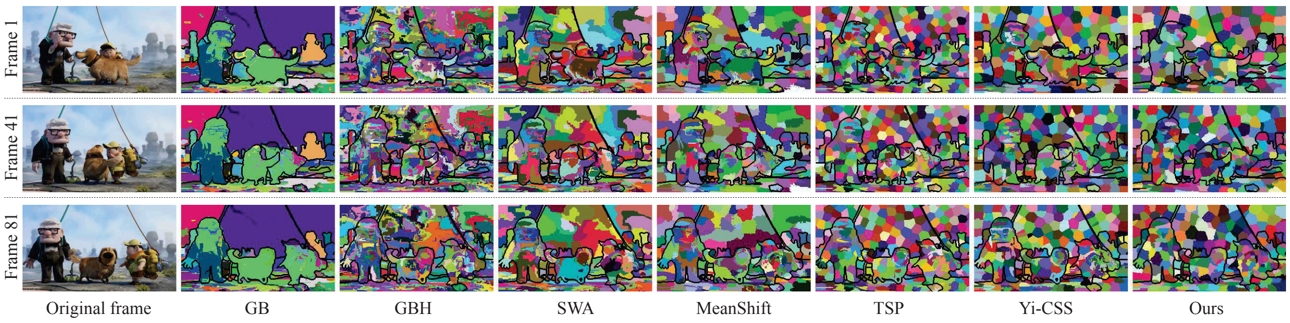


Figure 2. Visual comparison of supervoxels (which have different colors and are clipped on each image frame) computed by GB [14], GBH [17], SWA [32, 33, 10], MeanShift [29], TSP [7], Yi-CSS [46] and our method. All the methods generate approximately 1,000 supervoxels. Our method produces better results than the other methods in terms of UE3D, BRD, SA3D and CO on four video datasets (Figure 8).

1.2. Our contributions

In this paper, we propose a novel method for computing *exact* GCVTs, which runs 6-8 times faster than the state-of-the-art GCVT method [26]. Our method has a proved linear time complexity, i.e., $O(N)$ for N-pixel images or N-voxel videos and can guarantee optimal approximation ratio $O(1)$. We evaluate 31 superpixel methods on five image datasets and 8 supervoxel methods on four video datasets, and test them on applications including optimal image and video closure [22] and video foreground propagation [20]. The results show that our method provides an all-in-one solution and consistently performs well under a variety of metrics.

2. Preliminaries

Our method is built upon image and video manifolds, and K-means++, which are briefly summerized below.

2.1. Image manifold $\mathcal{M}_2$

Both MSLIC and IMSLIC [25, 26] adopt an embedding map Φ that lifts a color image I to a 2-manifold $\mathcal{M}_2 \subset \mathbb{R}^3$

$$\Phi(r, s) \triangleq (r, s, \lambda_1 l, \lambda_1 a, \lambda_1 b), \tag{1}$$

where (r, s) is the spatial coordinate, (l, a, b) is the pixel color in the CIELAB color space and λ_1 is a constant spec-

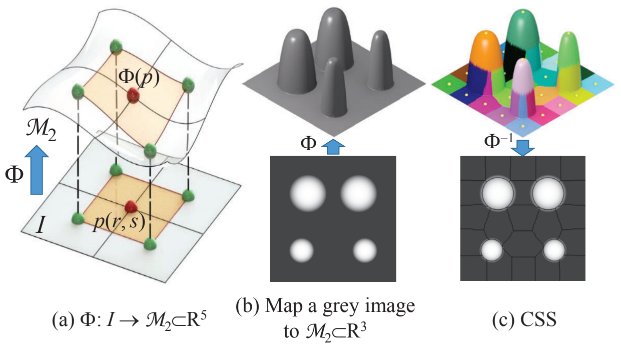

$\Phi(p)$
$\mathcal{M}_2$
Φ
I
$p(r,s)$
Φ
Φ^{-1}
(a) $\Phi: I \to \mathcal{M}_2 \subset \mathbb{R}^5$
(b) Map a grey image to $\mathcal{M}_2 \subset \mathbb{R}^3$
(c) CSS

Figure 3. MSLIC and IMSLIC. (a) Both methods represent pixel $p(r, s)$ (red dot) as a unit square (yellow region) whose corners (green dots) are the centers of their neighboring pixels. The stretching map Φ in Eq.(1) "lifts" a color image I into a curved 2-manifold $\mathcal{M}_2 \subset \mathbb{R}^5$, whose area element is a good measure of image content. (b) To ease visualization, we take a greyscale image as an example, which is mapped to a 3D surface $\mathcal{M}_2 \subset \mathbb{R}^3$. (c) Both MSLIC [25] and IMSLIC [26] compute a regular tessellation $\mathcal{T}$ on $\mathcal{M}_2$ (cells in $\mathcal{T}$ are distinguished by colors). The inverse mapping $\Phi^{-1}(\mathcal{T})$ induces content-sensitive superpixels on I.

ified in [25]. Similarly, a RGBD image can be mapped to a $M_2 \subset \mathbb{R}^6$ as $(r, s, \lambda_1 l, \lambda_1 a, \lambda_1 b, \lambda_2 d)$, where d is the depth.

As Figure 3 shows, the map Φ stretches content-dense

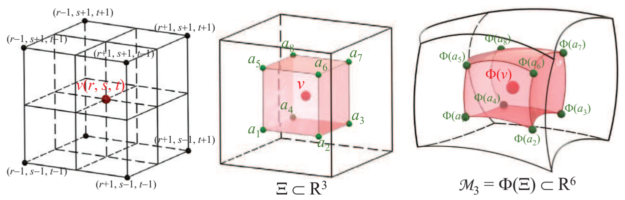


Figure 4. Yi-CSS [46] maps a voxel $v(r, s, t)$ (i.e., the red box in the middle) in a video Ξ into a curved 3-manifold $\mathcal{M}_3$ by $\Phi : \Phi \to \mathcal{M}_3 \subset \mathbb{R}^6$. Each corner a_i of the voxel box is the center of its eight surrounding voxels.

(content-sparse) regions in I into large (small) areas on $\mathcal{M}$. Then a *uniform* tessellation on $\mathcal{M}_2$ yields high quality CSS in I by the inverse mapping Φ^{-1}.

2.2. Video manifold $\mathcal{M}_3$

Yi-CSS [46] maps a video clip Ξ of voxels $v(r, s, t)$ to a 3-manifold $\mathcal{M}_3$ embedded in $\mathbb{R}^6$ (Figure 4)

$$\Phi(r, s, t) \triangleq (r, s, \lambda_1 t, \lambda_2 l, \lambda_2 a, \lambda_2 b) \tag{2}$$

where (r, s) is the pixel coordinate, t is the frame index, (l, a, b) is the pixel color in the CIELAB color space, λ_1 and λ_2 are two constants. Yi et al. [46] showed that akin to the 2-manifold M_2 for image, the inverse mapping Φ^{-1} of a uniform tessellation in $\mathcal{M}_3$ results in good CSS in Ξ.

2.3. K-means++

Given N points $X = \{x_i\}_{i=1}^N$ in a real metric space $\mathcal{X}$ with metric $D(x, y)$, data clustering techniques target on minimizing the following potential function by choosing K cluster centers $\{c_j\}_{j=1}^K$ in $\mathcal{X}$:

$$\mathcal{E}\left(\{c_j\}_{j=1}^K\right) = \sum_{i=1}^{n} \min_{j=1,2,\cdots,K} D^l(x_i, c_j) \tag{3}$$

where the exponent $l \in \mathbb{Z}_+$ is a problem parameter. The centers $\{c_i\}_{i=1}^K$ partition the point set X into K clusters, each of which is a subset $X_i = \{x_s \in X : D^l(x_s, c_i) \leq D^l(x_s, c_j), i \neq j\}$. In particular, when $l = 2$ and D is the Euclidean metric, it is known as the K-means problem [13].

The K-means++ algorithm iteratively chooses cluster centers [3]. In the beginning, it takes a center c_1 randomly from X. In each subsequent step i, $1 < i \leq K$, a new cluster center is chosen randomly from points $x_s \in X \backslash \{c_j\}_{j=1}^{i-1}$ with probabilities proportional to $\min_{c \in \{c_j\}_{j=1}^{i-1}} D^l(x_s, c)$.

Given a fixed K, let $\{c_j^{opt}\}_{j=1}^K$ be the (unknown) optimal centers which minimizes the potential (3). An algorithm is said to have an approximation ratio α, if for any $\{c_j\}_{j=1}^K$ output from this algorithm, $\frac{\mathcal{E}\left(\{c_j\}_{j=1}^k\right)}{\mathcal{E}\left(\{c_j^{opt}\}_{j=1}^k\right)} \leq \alpha$. It was shown in [42] that for any constant factor $\beta > 1$, selecting βk cluster centers by the K-means++ algorithm leads to an $O(1)$-approximation in expectation.

3. Overview of Our Method

State-of-the-art CSS work [26, 46] maps the input image or video to a ζ-dimensional manifold M_ζ embedded in $\mathbb{R}^d$, $\zeta = 2, 3$, $\zeta < d$. For example, a color image (video) to a 2-manifold M_2 (M_3) in $\mathbb{R}^5$ ($\mathbb{R}^6$). A common characteristic of these manifolds $M_\zeta \subset \mathbb{R}^d$ is that the geodesic metric — which defines the lengths, area or volumes on M_ζ — is a good measure of the content density in images and videos. As a result, a *uniform* tessellation on M_ζ induces a *non-uniform* tessellation on the input image and video, which is content-sensitive superatoms.

To compute a uniform tessellation on M_ζ, geodesic centroidal Voronoi tessellation (GCVTs) [12, 26] is a commonly used tool, since it is an intrinsic structure on M_ζ, which only depends on the geodesic metric without further references to the ambient space $\mathbb{R}^d$. With a simple extension of the proof in [26] from M_2 to M_ζ, $\zeta = 2, 3$, we have the following general results.

Given a set of K generators $C = \{c_i\}_{i=1}^K \subset \mathcal{M}_\zeta$, the geodesic Voronoi tessellation (GVT) on M_ζ is the set of Voronoi cells $\{V(c_i)\}_{i=1}^K$ on $\mathcal{M}_\zeta$:

$$\begin{aligned} & V(c_i) = \{x \in \mathcal{M}_\zeta : d_g(x, c_i) \leq d_g(x, c_j), \forall j \neq i\}, \\ & i = 1, 2, \cdots, K \end{aligned} \tag{4}$$

where $d_g(x, y)$ is the geodesic distance between x and y on $\mathcal{M}_\zeta$. A GVT is a GCVT if each c_i is the mass centroid of its Voronoi cell, defined as the solution to the following problem [26]:

$$\min_{z \in \mathcal{M}_\zeta} \int_{x \in V_g(c_i)} d_g^2(x, z) dx \tag{5}$$

Let $X = \{x_i\}_{i=1}^N$ be an N-atom media (i.e., an image or a video) and $\mathcal{M}_\zeta = \Phi(X)$ the stretched manifold in $\mathbb{R}^d$. We discretize manifold $\mathcal{M}_\zeta$ by a graph $\mathcal{G} = \{V, E\}$, where the vertex set V contains the mappings of all pixels/voxels $\{v_i = \Phi(x_i)\}_{i=1}^N$, and an edge $e = (v_i, v_j) \in E$ is defined when $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ are n_ζ-neighbors in X:

- $n_\zeta = 8$ for $\zeta = 2$: i.e., for X being an image, $\Phi^{-1}(v_i) = (r_i, s_i)$ and $\Phi^{-1}(v_j) = (r_j, s_j)$ are 8-neighbors, which satisfy $\|r_i - r_j\|_2 \leq 1$ and $\|s_i - s_j\|_2 \leq 1$.
- $n_\zeta = 26$ for $\zeta = 3$: i.e., for X being a video, $\Phi^{-1}(v_i) = (r_i, s_i, t_i)$ and $\Phi^{-1}(v_j) = (r_j, s_j, t_j)$ are 26-neighbors, which satisfy $\|r_i - r_j\|_2 \leq 1$, $\|s_i - s_j\|_2 \leq 1$ and $\|t_i - t_j\|_2 \leq 1$.

Denote the number of vertices and edges in $\mathcal{G}$ by $N = |V|$ and $|E|$, respectively. Since $\mathcal{G}$ is a sparse graph, we have $|E| = O(N)$. To evaluate the geodesic metric, we apply Dijkstra's algorithm [9] to compute the shortest paths from multiple sources to all other vertices in $\mathcal{G}$, which runs in $O(N \log N)$ time.

The GCVT computation proposed in [26] has three limitations: 1) it runs slowly due to Dijkstra's algorithm; 2) it solves Eq. (5) using an approximate method that works only for image manifolds and cannot be extended to video manifolds; and 3) it lacks theoretic bound on the tessellation performance. Replacing geodesic metrics on M_ζ by Euclidean metric in $\mathbb{R}^d$, Yi et al. [46] partially addressed these limitations. However, Euclidean metric depends on the embedding space $\mathbb{R}^d$ and often produces tiny fragments and multiple disjoint components in a Voronoi cell.

In this paper, we propose a new GCVT computation method that overcomes the above-mentioned limitations. First, to establish a theoretic bound, we introduce K-means++ into GCVT for obtaining a high quality initialization. Second, we propose a centroid-free Lloyd refinement such that we no longer need to solve the problem (5) and then our method is suitable for both image and video manifolds. Algorithm 1 summarizes these two improvements, which consists of two steps:

- Initialization (steps 1-8). We configure K-means++ with $X = \{\Phi(p_i)\}_{i=1}^N$, $\mathcal{X} = \mathcal{M}_\zeta$ and $D = d_g$, and apply it to determine the initial positions of K cluster centers $C = \{c_i\}_{i=1}^K$ on $\mathcal{M}_\zeta$;
- Centroid-free Lloyd refinement (steps 9-17). For each center $c_i \in C$, we quickly find an alternative $\overline{c}_i$ to it. If $\mathcal{E}\left(\{\overline{c}_j\}_{j=1}^K\right) < \mathcal{E}\left(\{c_j\}_{j=1}^K\right)$, we replace $\{c_i\}_{i=1}^K$ by $\{\overline{c}_i\}_{i=1}^K$ and keep iterating; otherwise the iteration stops. Unlike the method [26] that explicitly computes the mass centroids of Voronoi cells, our refinement does not require an accurate centroid at all. Therefore, we call it *centroid-free* refinement.

Property 1. *Algorithm 1 is $O(1)$-approximation.*

Proof. The initialization step applies K-means++ with the geodesic metric d_g in the manifold space $\mathcal{M}_\zeta$. By Corollary 1 in [42] (with the K-subset selection in [2]), the expected approximation ratio is bounded by $\frac{\mathbb{E}\left[\mathcal{E}\left(\{c_j\}_{j=1}^k\right)\right]}{\mathcal{E}\left(\{c_j^{opt}\}_{j=1}^k\right)} \leq O(1)$. Since in the centroid-free Lloyd refinement, the potential function decreases strictly, Algorithm 1 is $O(1)$-approximation in expectation. □

Algorithm 1 achieves very good over-segmentation accuracy but runs slowly due to the Dijkstra's algorithm. In the next section, we propose a queue-based graph path (q-path) and distance (q-distance), which runs only in $O(N)$ time for computing these paths to all vertices in $\mathcal{G}$ from multiple sources (Algorithm 2). We refer to the variant of Algorithm 1 — which replaces the shortest paths and distances by q-paths and q-distances — as qd-CSS.

Algorithm 1 Improved CSS (using shortest paths)

```
Input: A media $X$ of $N$ atoms, the desired number of superatoms $K$, the maximal number of iterations $iter_{\max}$.
Output: $K$ content-sensitive superatoms.
 1: Map each atom $x_i \in X$ to a point $\Phi(x_i)$.
 2: Build a graph $\mathcal{G} = \{V, E\}$, where the vertex set $V = \{\Phi(x_i)\}_{i=1}^N$ and the edge set $E = \{(\Phi(x_i), \Phi(x_j)) : x_i$ and $x_j$ are $n_\zeta$-neighbors in $X\}$.
 3: Choose a center $c_1$ uniformly at random from $V$, and initialize $C_1 = \{c_1\}$ and $i = 1$.
 4: while $i < K$ do
 5:     Compute the shortest paths in $\mathcal{G}$ from $C_i$ to all vertices $\Phi(x_j) \in V \backslash C_i$, and record the shortest distance $d_g(\Phi(x_j), C_i) = \min_{c_s \in C_i} d_g(\Phi(x_j), c_s)$.
 6:     Choose a center $c_{i+1}$ from $V \setminus C_i$ with probability proportional to $d_g(\Phi(x_j), C_i)$.
 7:     $C_{i+1} = C_i \cup \{c_{i+1}\}$ and $i \leftarrow i + 1$.
 8: end while
 9: Initialize the set of candidate centers $\overline{C} = C_K$, $\delta = -1.0$ and $iter = 1$.
10: while $\delta < 0$ and $iter \leq iter_{max}$ do
11:     $C_K \leftarrow \overline{C}$.
12:     For each center $c_i \in C_K$, compute the cluster $V_i = \{\Phi(x) \in V : d_g(c_i, \Phi(x)) < d_g(c_j, \Phi(x)), i \neq j\}$.
13:     For each $V_i$, set $\widetilde{c}_i = \sum_{\Phi(x) \in V_i} \Phi(x)/|V_i|$, where $|V_i|$ is the number of vertices in $V_i$.
14:     For each candidate center $\overline{c}_i \in \overline{C}$, renew its position by $\overline{c}_i = \Phi(x)$, where $x$ is the nearest atom in $X$ to $\pi(\widetilde{c}_i)$, where $\pi$ truncates the coordinate $(r, s)$ (for image) or $(r, s, t)$ (for video) of $\widetilde{c}_i$.
15:     Set $\delta = \mathcal{E}\left(\{\overline{c}_j\}_{j=1}^K\right) - \mathcal{E}\left(\{c_j\}_{j=1}^K\right)$.
16:     $iter \leftarrow iter + 1$.
17: end while
18: Output the $K$ clusters $\{\Phi^{-1}(V_i)\}_{i=1}^K$ in $X$, where $\Phi^{-1}(V_i) = \{x : \Phi(x) \in V_i\}$.
```

4. Queue-based Graph Paths and Distances

The bottleneck of the Dijkstra's algorithm is the use of a priority queue for maintaining the to-be-processed nodes, since insertion takes $O(\log n)$ time for a priority queue with n elements. To improve performance, we propose to replace the priority queue by an *first-in-first-out (FIFO) queue*. See Algorithm 2 for the pseudo code.

The q-paths/distances obtained from FIFO queue does not equal to shortest paths/distances. Our key idea is that at manifold regions where q-distances are different from shortest distances, GCVT is prone to placing more generators in them, and therefore after few iterations, all q-distances restricted in the final tessellation are shortest distances. In this section, we present three properties and two observations to demonstrate this idea. Proofs of properties are presented in supplemental material.

Replacing Dijkstra's shortest distances by q-distances on $\mathcal{G}$ in Algorithm 1, our method (named qd-CSS) has a linear time complexity $O(N)$. In practice, qd-CSS runs 6-8 times faster than the state-of-the-art work [26] on images with resolution 481×321.

4.1. Properties

Since the queue in Algorithm 2 works in an FIFO manner, given an N-atom media X and a set of multiple sources $C = \{c_i\}_{i=1}^{n_c} \subset V$, the traversal order of all vertices in V is predefined by the order of visiting neighbors of each vertex, i.e., it only depends on the edge connectivity in $\mathcal{G}$ and is irrelevant to the media content. Each vertex in $\mathcal{G}$ has n_ζ neighbors, whose visiting order can be characterized by the coordinate of $\Phi^{-1}(v)$. For example, when $\zeta = 2$, the neighboring order of v can be determined by (r, s) coordinate of $\Phi^{-1}(v)$ as Γ = (N, W, S, E, NW, SW, SE, NE), where $(r+1, s)$ for east (E), and then southeast (SE), south (S), southwest (SW), west (W), northwest (NW), north (N) and northeast (NE) are defined in the clockwise order.

Let $d_q(v, c)$ be the q-distance from v to c and $d_q(v, C) = \min_{c_i \in C} d_q(v, c_i)$. Based on the predefined traversal order $\{v_1 = c_1, v_2 = c_2, \cdots, v_K = c_K, v_{K+1}, v_{K+2}, \cdots, v_N\}$, we assign an index I_i to each vertex $v_i \in V$. Then the q-path $\widetilde{cv_i} = \{v_{I_{i_1}} = c, v_{I_{i_2}}, \cdots, v_{I_{i_n}} = v_i\}$ from a center $c \in C$ to a vertex $v_i \in V \setminus C$, satisfies that $\forall i_a, i_b$, $1 \leq a < b \leq n$, the indices $I_{i_a} < I_{i_b}$.

If $\mathcal{G}$ is a regular lattice, i.e., all media atoms have the same color, any q-path is exactly the shortest path on $\mathcal{G}$. When the color variation in X is large, the manifold $\mathcal{M}_\zeta$ will be bumpy, but the q-paths still have chance to pass around the ridges and valleys, and be the shortest paths, e.g., green paths in Figures 5(a) and 5(b). The following definition and properties study the conditions under which q-paths are shortest paths. Afterwards, we study the condition under which q-paths are different from shortest paths and present our key observations that when this condition occurs, GCVT will place more centers in the corresponding regions such that after few iterations, the final q-distance-induced tessellation is an exact GCVT.

Property 2. *For any $c \in C$, $v \in V \setminus C$ and a shortest path $\overline{cv} = \{v_{I_{j_1}} = c, v_{I_{j_2}}, \cdots, v_{I_{j_{n'}}} = v\}$ between c and v on $\mathcal{G}$, the q-path $\widetilde{cv}$ output from Algorithm 2 is exactly the shortest path $\overline{cv}$, if and only if $\forall a, b$, $1 \leq a < b \leq n'$, the indices $I_{j_a} < I_{j_b}$.*

Definition 1. *For each vertex $v_i \in V$, we define an allowable region $\Omega(v_i)$ of v_i, which is a set of vertices satisfying $\Omega(v_i) = \{v_j \in V : j < i\}$.*

In supplemental material, we show that the allowable regions are sufficiently large using the i-ring concept. Figure 5(c) illustrates three allowable regions $\Omega(v_i)$, $\Omega(v_j)$ and $\Omega(v_f)$ of three vertices v_i, v_j and v_f on the q-path $\widetilde{c_1 v_f}$.

Algorithm 2 Computing q-paths and q-distances

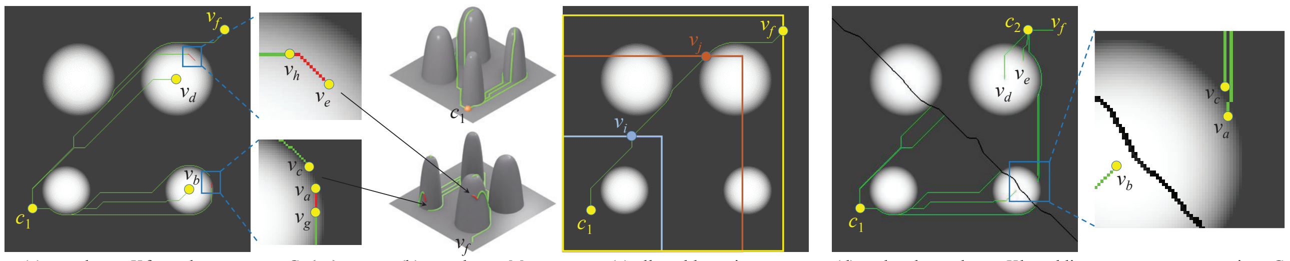
```
Input: A sparse graph $\mathcal{G} = (V, E)$ discretizing manifold $\mathcal{M}_\zeta$ and multiple sources $C = \{c_i\}_{i=1}^{n_c} \in V$.
Output: The q-paths and q-distances from $C$ to all vertices in $V \setminus C$.
 1: For each node $v \in V$, attach three attributes: a distance value $v.dist$, a Boolean flag $v.visit$ and a precedent node ID $v.pre$;
 2: For each source $c_i \in C$, initialize $c_i.dist = 0$, $c_i.visit = TRUE$, and for all other vertices $v \in V$, $v.dist = \infty$, $v.visit = FALSE$
 3: Initialize a queue $Q = C$
 4: while $Q$ is not empty do
 5:   Extract and remove the element $v_a$ from the head of the queue
 6:   for each neighbor $v_b$ of $v_a$ in $\mathcal{G}$ do
 7:     Set $l(v_a, v_b)$ be the length of edge $(v_a, v_b) \in E$
 8:     if $v_b.dist > v_a.dist + l(v_a, v_b)$ then
 9:       $v_b.dist = v_a.dist + l(v_a, v_b)$
10:       $v_b.pre = v_a$
11:     end if
12:     if $v_b.visit == FALSE$ then
13:       Insert $v_b$ into $Q$ at the tail.
14:       Set $v_b.visit = TRUE$.
15:     end if
16:   end for
17: end while
18: For each vertex $v \in V \setminus C$, output the q-distance $v.dist$
19: (optional) For each vertex $v \in V \setminus C$, output the q-path by backtracking the precedent nodes starting from $v$ until a source in $C$ is reached.
```

Property 3. *For any $c \in C$, $v \in V \setminus C$ and a shortest path $\overline{cv} = \{v_{I_{j_1}} = c, v_{I_{j_2}}, \cdots, v_{I_{j_{n'}}} = v\}$ between c and v on $\mathcal{G}$, the q-path $\widetilde{cv}$ output from Algorithm 2 is exactly the shortest path $\overline{cv}$, if and only if $\forall i$, $1 \leq i \leq n'$, the subpath $\overline{cv_{I_i}}$ of $\overline{cv}$ is contained in the allowable region of v_{I_i}.*

By Property 2, if at steps 8-10 of Algorithm 2, the q-distance value $\Phi(v_b).dist$ is updated due to $\Phi(v_b).dist > \Phi(v_a).dist + l(v_a, v_b)$ and the indices $b < a$, then any q-path $\widetilde{cv'}$ passing through v_b cannot be the shortest path on $\mathcal{G}$. The following property shows the condition under which a q-path cannot be a shortest path.

Property 4. *Assume $v \in V$ is in a general position, i.e., it has n_ζ neighbors in V. Then in these neighbors, half of them have indices larger than v.*

In Algorithm 2, when a vertex v_a is extracted and removed from the head of the queue, the q-path from a center



(a) q-paths on X from the center set C={c_1}   (b) q-paths on M_2 (c) allowable regions   (d) updated q-paths on X by adding one more center c_2 into C

Figure 5. Since Φ is a one-to-one mapping, we can visualize the q-paths on both the media X and the manifold $\mathcal{M}_\zeta = \Phi(X)$. For a clear visualization, here we use a grey image for X. (a) Given the center set $C = \{c_1\}$, the q-paths from c_1 to $v_a, v_b, \cdots, v_f$ are illustrated as a tree rooted at c_1 in X. On these paths, if the sub-q-path $\widetilde{c_1 v_x}$ is a shortest path $\overline{c_1 v_x}$, the vertex v_x is shown in green; otherwise, v_x is shown in red. On a q-path starting from c_1, if a vertex v_x is red (e.g., $v_g \in \widetilde{c_1 v_a}$ and $v_h \in \widetilde{c_1 v_e}$), all the subsequent vertices are also red. (b) The corresponding q-paths on $\mathcal{M}_2$. (c) On the q-path $\widetilde{c_1 v_f}$ which is also a shortest path, allowable regions $\Omega(v_i)$, $\Omega(v_j)$ and $\Omega(v_f)$ of three vertices v_i, v_j and v_f are illustrated by bordering with different colors. (d) When one more center c_2 is added into C, the q-paths to v_a, v_c, v_d, v_e and v_f are updated by replacing the center from c_1 to c_2, and all of them are green, i.e., their q-distances to C are also shortest distances. The black line is the bisector between c_1 and c_2.

c to v_a can be extended further to those neighbors with an index large than a. Property 4 reveals that the number of these extendable neighbors are not smaller than $n_\zeta/2$. As a comparison, in the Dijkstras algorithm, the path from c to any v can be extended to any one of $n_\zeta - 1$ neighbors[2] of v. This explains the risk that the q-paths will fail to be the shortest paths.

The q-paths from the set of centers C to all vertices $v \in V \setminus C$ can be visualized by trees with roots at centers in C (Figure 5a). In these trees, the parent of each vertex v is the precedent vertex of v found in Algorithm 2. We say a vertex v is *wrongly labelled* (red points in Figure 5a), if its q-path $\widetilde{cv}$ is not a shortest path.

Observation 1. *If a vertex v is wrongly labelled, then all the descendent vertices of v in the tree are wrongly labelled. However, on bumpy manifold $\mathcal{M}_\zeta$, the number of descendent vertices of a wrongly labelled vertex is small.*

Observation 1 can be explained by that when v is wrong, it means that the q-path $\widetilde{cv}$ passes through a high-stretched region like the cliff of a mountain; e.g., v_g and v_h in Figure 5a. For any vertex v' a little bit far beyond the high-stretched region (e.g., v_c in Figure 5a), by Property 3, it has a big chance that the q-path $\widetilde{v_c v'}$ can circumvent the high-stretched region and be the shortest path; e.g., the q-path $\widetilde{c_1 v_c}$ in Figure 5a.

Observation 2. *On the manifold $\mathcal{M}_\zeta$, more stretched a region is, the higher possibility this region is on the boundary of superatoms. Therefore, during the tree propagation starting from a center c, when a vertex v becomes wrongly labelled, it is likely that the q-path $\widetilde{cv}$ goes through a boundary at v.*

[2]Among the eight neighbors, one is already on the path.

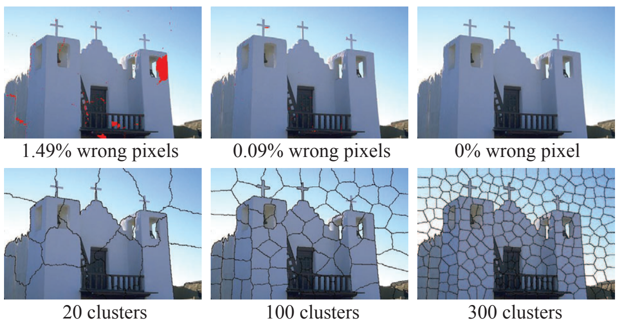
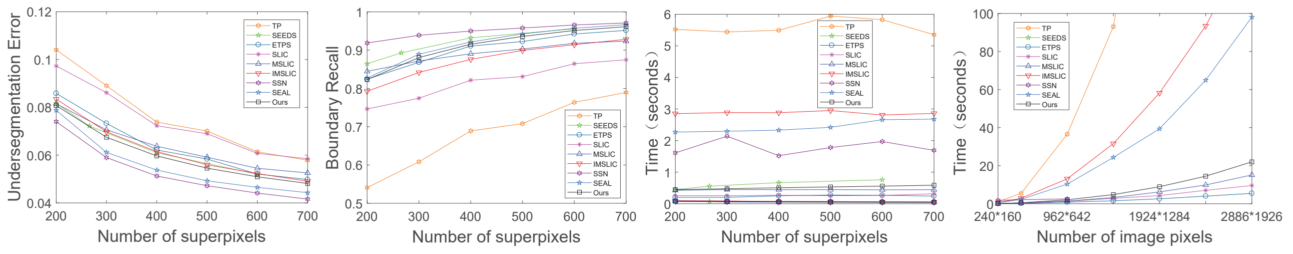


Figure 6. When more centers are selected into C, the number of wrongly-labelled pixels (i.e., their q-distances are not the shortest distance on the graph $\mathcal{G}$) decrease dramatically. Increasing the number of clusters can effectively reduce the number of wrongly-labelled pixels. We did not show the results of IMSLIC [26], since they are almost visually identical to ours when $K = 20, 100$ and exactly the same when $K = 300$.

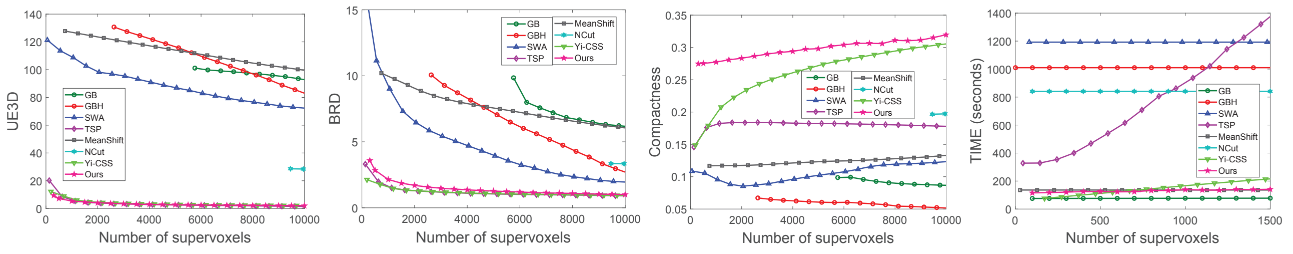
It can be regarded that q-paths put a heavy penalty on the distance when passing through the boundary, and this characteristic is desired in our CSS application. In the initialization phase of qd-CSS, The farther away the point is from existing centers, the higher the probability that this point is selected as the next center. Then, when more centers are added iteratively, the wrong path $\widetilde{cv'}$ will be corrected by another path $\widetilde{c'v'}$, given that c and c' come from different side of the boundary (Figure 5d). In supplemental material, we prove a proposition, indicating that if the shape of manifold $\mathcal{M}_\zeta$ satisfies certain assumptions (characterized by the edge length ratio in $\mathcal{G}$) and a moderately large number[3] K of centers are selected, the clustering $\{V_i\}_{i=1}^K$ on $\mathcal{G}$ is exactly the same for using either shortest distance or q-distance. Figure 6 shows a real example.

[3]For example, $K \geq 200$ is sufficient for an image of 481×321 pixels.



(a) Under-segmentation error (b) Boundary recall (c) Running time with respect to K (d) Running time with respect to N

Figure 7. Evaluation of nine superpixel methods on the BSDS500 dataset for $K \in [200, 700]$. Our method (qd-CSS) is 6-8 times, 4-5 times and 3-4 times faster than IMSLIC, SEAL and SSN, respectively. Although ETPS and SLIC run faster than qd-CSS, qd-CSS has lower UE and higher BR. See text for details. More comparisons of 31 superpixel methods on five datasets are presented in supplemental material.



(a) 3D under-segmentation error (b) Boundary recall distance (c) Compactness (d) Running time

Figure 8. Evaluation of eight supervoxel methods on the BuffaloXiph dataset. Our method has the smallest UE3D and BRD, the highest CO and the second fastest running time. More comparisons on four datasets are presented in supplemental material.

5. Experiments

We implemented qd-CSS[4] in C++ and tested it on a PC with an Intel E5-2698v3 CPU (2.30 GHz) and 128 GB RAM. In addition to the number of superatoms, qd-CSS has only one parameter, the maximal iteration number $iter_{max}$ in Algorithm 1, which is set to 10 in all experiments. Because qd-CSS uses a random initialization, we report the average results of 20 initializations.

Evaluation on superpixels. Figure 7 summarizes the comparison of nine representative methods: TurboPixels [23], SEEDS [11], ETPS [45], SLIC [1], MSLIC [25], IMSLIC [26], SSN [21], SEAL [40] and qd-CSS, in which SSN and SEAL are two deep learning methods. We use three commonly used measures — under segmentation error (UE) [1, 23], boundary recall (BR) [27] and running time — to evaluate the performance of different superpixels: a lower UE value means that superpixels are better overlapped with a ground-truth segmentation, and a higher BR value means that fewer true ground-truth edges are missed. The results in Figure 7 are averaged on BSDS500 datasets, showing that

- Deep learning methods (SSN and SEAL) have the best performance on UE and BR. However, they use GPU to train and run the deep networks, while qd-CSS only uses CPU. In our machine with NVIDIA TITAN Xp 12G, both methods are slower than qd-CSS. When image resolution is higher than 1443×693, SSN does not work. So far the implementations of SSN and SEAL apply to images only. To extend them to videos, one may have to redesign the network architecture and the loss function. Also, re-training the network is necessary. In comparison, qd-CSS provides a low-cost, all-in-one solution to both images and videos.

- In the class of methods with artificially designed features, qd-CSS has the lowest UE and the second highest BR. SEEDS has a better BR than qd-CSS, but it does not take the superpixel number as input. By carefully controlling other parameters, it can only generate superpixels with numbers around 200, 266, 400 and 600. In terms of speed, ETPS and SLIC are two times faster than qd-CSS, while qd-CSS is 6-8 times faster than IMSLIC.

Figure 1 illustrates the visual comparison of these methods. More qualitative and quantitative comparison of 31 superpixel methods (with more measures including achievable segmentation accuracy [24, 41] and compactness [31] on three more datasets, NYUV2 [36], SUNRBGD [37] and Fashionista [44]) are presented in supplemental material.

Evaluation on supervoxels. Figure 8 summarizes the comparison of eight representative methods: NCut [34, 16, 15], SWA [32, 33, 10], MeanShift [29], GB [14], GBH [17], TSP [7], Yi-CSS [46] and qd-CSS. To evaluate their performance, we use the supervoxel counterpart of the meaures BR and UE, i.e., Boundary recall distance (BRD) [28, 43], 3D under-segmentation error (UE3D) [7, 23, 43]. We also

[4]Source code is available http://cg.cs.tsinghua.edu.cn/people/~Yongjin/Yongjin.htm

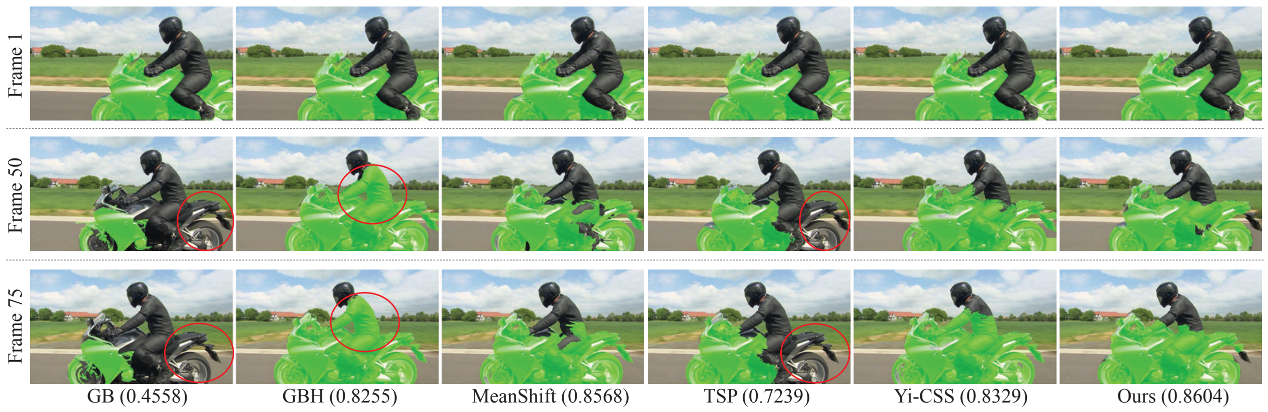


Figure 9. Foreground propagation results of six supervoxel methods on an example in Youtube-objects dataset [30]. Three representative frames are selected. The foreground masks are shown in green. The incorrectly labeled areas are circled in red. The average $F \in [0, 1]$ measure for each example video is shown in the brackets and larger values mean better results.

use the compactness (CO) metric that measures the shape regularity of supervoxels. The results in Figure 8 are averaged on the BuffaloXiph dataset [8], showing that qd-CSS has the smallest UE3D and BRD, the highest CO and the second fastest running time. Figure 2 illustrates the visual comparison of seven methods. More qualitative and quantitative comparison are presented in supplemental material.

6. Applications

Since superpixels and supervoxels are designed to reduce the complexity of downstream computer vision tasks, we directly evaluate them and demonstrate the efficiency of qd-CSS on one image and two video applications.

Optimal image and video closure. Levinshtein et al. [22] propose a novel framework that separates an object from background by finding subsets of superpixels/supervoxels such that the contour of the union of these atomic regions has strong boundary support in the image/video. We use the source code provided by the authors[5] to compare different superpixels/supervoxel methods on an image dataset WHD [5] and a video dataset [38] with ground-truth segmentations. In 31 superpixel methods, qd-CSS and ETPS are selected in Section S3 in supplemental material and are compared for image contour closure. Figure S11 illustrate some qualitative results and the F-measure values (averaged on the WHD dataset) are summarized in Figure S12 in supplemental material, showing that qd-CSS has better performance than ETPS. For optimal video closure by supervoxel grouping, the dataset of Stein et al. [38] in which each sequence has a ground truth segmentation mask, is used to perform a quantitative assessment. Seven representative methods (GB, GBH, NCut, MeanShift, SWA, TSP, Yi-CSS) and our qd-CSS are compared. The average F measures across all sequences are summarized in Figure S13 and some qualitative results are illustrated in Figure S14 in supplemental material. These results show that qd-CSS achieves the best spatiotemporal closure performance.

Foreground propagation in videos. Given the first frame with manual annotation for the foreground object, a novel approach is proposed in [20] to propagate the foreground region through time, by using supervoxels to guide the estimates towards long-range coherent regions. We use the source code provided by the authors[6] to compare five representative methods[7] (GB, GBH, MeanShift, TSP and Yi-CSS) and our qd-CSS. Youtube-Objects dataset [30] (126 videos in 10 object classes) with foreground groundtruth, is used to perform a quantitative assessment. The average F measures of 10 classes are summarized in Figure S15 in supplemental material, showing that qd-CSS achieves the best performance in four classes and achieves the best performance averagely over ten classes. Some qualitative results are illustrated in Figure 9.

7. Conclusion

In this paper, we propose qd-CSS for computing content-sensitive superatoms. By using a low-cost q-distances, qd-CSS significantly cuts down the data management overhead. As a result, it runs 6-8 times faster than the state-of-the-art IMSLIC [26]. Moreover, qd-CSS also works with supervoxels for videos thanks to centroid-free refinement. In a thorough evaluation involving 31 superpixel methods and 8 supervoxel methods on benchmark image and video datasets, we observe that qd-CSS achieves good balance among various measures.

Acknowledgment

This work was supported by the Natural Science Foundation of China (61725204, 61521002) and BNRist.

[5]http://www.cs.toronto.edu/∼babalex/spatiotemporal_closure_code.tgz

[6]www.cs.utexas.edu/∼suyog/code_release_public.tar

[7]NCut is not compared due to its high computational cost. SWA is not compared since it requires huge memory for long videos in this dataset.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(11):2012–2281, 2012. 1, 2, 7

[2] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, pages 15–28. Springer-Verlag, 2009. 4

[3] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, 2007. 3

[4] András Bódis-Szomorú, Hayko Riemenschneider, and Luc J. Van Gool. Superpixel meshes for fast edge-preserving surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 2011–2020, 2015. 1

[5] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *Proceedings of the 7th European Conference on Computer Vision-Part II*, ECCV '02, pages 109–124. Springer-Verlag, 2002. 8

[6] Yiqi Cai and Xiaohu Guo. Anisotropic superpixel generation based on mahalanobis distance. *Computer Graphics Forum (Pacific Graphics 2016)*, 35(7):199–207, 2016. 1

[7] Jason Chang, Donglai Wei, and John W. Fisher III. A video representation using temporal superpixels. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 2051–2058, 2013. 2, 7

[8] Albert Y.C. Chen and Jason J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proceedings of Western New York Image Processing Workshop*, 2010. 8

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. 3

[10] Jason J. Corso, Eitan Sharon, Shishir Dube, Suzie El-Saden, Usha Sinha, and Alan L. Yuille. Efficient multilevel brain tumor segmentation with integrated Bayesian model classification. *IEEE Trans. Med. Imaging*, 27(5):629–640, 2008. 2, 7

[11] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015. 1, 2, 7

[12] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999. 3

[13] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2nd. edition, 2004. 3

[14] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 1, 2, 7

[15] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004. 7

[16] Charless C. Fowlkes, Serge J. Belongie, and Jitendra Malik. Efficient spatiotemporal grouping using the Nyström method. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '01, pages 231–238, 2001. 7

[17] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan A. Essa. Efficient hierarchical graph-based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'10, pages 2141–2148, 2010. 2, 7

[18] Saurabh Gupta, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation. *International Journal of Computer Vision*, 112(2):133–149, 2015. 1

[19] Shengfeng He, Rynson W. H. Lau, Wenxi Liu, Zhe Huang, and Qingxiong Yang. Supercnn: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3):330–344, 2015. 1

[20] Suyog Dutt Jain and Kristen Grauman. Supervoxel-consistent foreground propagation in video. In *13th European Conference on Computer Vision*, ECCV '14, pages 656–671, 2014. 2, 8

[21] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *Computer Vision - ECCV 2018 - 15th European Conference*, pages 363–380, 2018. 1, 2, 7

[22] Alex Levinshtein, Cristian Sminchisescu, and Sven Dickinson. Optimal image and video closure by superpixel grouping. *International Journal of Computer Vision*, 100(1):99–119, 2012. 1, 2, 8

[23] Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, Sven J. Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009. 1, 2, 7

[24] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2097–2104, 2011. 7

[25] Yong-Jin Liu, Chengchi Yu, Minjing Yu, and Ying He. Manifold SLIC: a fast method to compute content-sensitive superpixels. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 651–659, 2016. 1, 2, 7

[26] Yong-Jin Liu, Minjing Yu, Bing-Jun Li, and Ying He. Intrinsic manifold SLIC: A simple and efficient method for computing content-sensitive superpixels. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(3):653–666, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[27] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004. 7

[28] Alastair Philip Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices.

In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 7

[29] Sylvain Paris and Frédo Durand. A topological approach to hierarchical segmentation using mean shift. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'07, 2007. 2, 7

[30] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'12, pages 3282–3289, 2012. 8

[31] Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. An evaluation of the compactness of superpixels. *Pattern Recognition Letters*, 43(1):71–80, 2014. 7

[32] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '00, pages 1070–1077, 2000. 2, 7

[33] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006. 2, 7

[34] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000. 7

[35] Kaleem Siddiqi and Benjamin B. Kimia. Parts of visual form: Computational aspects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3):239–251, 1995. 1

[36] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the 12th European Conference on Computer Vision*, ECCV '12, pages 746–760, 2012. 7

[37] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 567–576, 2015. 7

[38] Andrew N. Stein, Derek Hoiem, and Martial Hebert. Learning to find object boundaries using motion cues. In *IEEE 11th International Conference on Computer Vision*, ICCV '07, pages 1–8, 2007. 8

[39] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018. 1

[40] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 568–576, 2018. 1, 2, 7

[41] Peng Wang, Gang Zeng, Rui Gan, Jingdong Wang, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. *International Journal of Computer Vision*, 103(1):1–21, 2013. 1, 7

[42] Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In *Annual Conference on Neural Information Processing Systems*, NIPS '16, pages 604–612, 2016. 3, 4

[43] Chenliang Xu and Jason J. Corso. Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 119(3):272–290, 2016. 7

[44] Kota Yamaguchi, M. Hadi Kiapour, Luis E. Ortiz, and Tamara L. Berg. Parsing clothing in fashion photographs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '12, pages 3570–3577, 2012. 7

[45] Jian Yao, Marko Boben, Sanja Fidler, and Raquel Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 2947–2955, 2015. 1, 2, 7

[46] Ran Yi, Yong-Jin Liu, and Yu-Kun Lai. Content-sensitive supervoxels via uniform tessellations on video manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '18, pages 646–655, 2018. 1, 2, 3, 4, 7