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) 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.
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 [23] 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 summarized below.

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

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

$$\Phi(r, s) \triangleq (r, s, \lambda_1 l, \lambda_1 a, \lambda_1 b),$$

where $(r, s)$ is the spatial coordinate, $(l, a, b)$ is the pixel color in the CIELAB color space and $\lambda_1$ is a constant spec-

ified in [25]. Similarly, a RGBD image can be mapped to a $\mathcal{M}_2 \subset \mathbb{R}^6$ 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 $\Phi$ stretches content-dense
3. Overview of Our Method

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

To compute a uniform tessellation on \( M_\zeta \), geodesic centroidal Voronoi tessellation (GCVTs) [12, 26] is a commonly used tool, since it is an intrinsic structure on \( M_\zeta \), 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 \( M_\zeta \), \( \zeta = 2, 3 \), we have the following general results.

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

\[
V(c_i) = \{ x \in M_\zeta : d(y, x, c_i) \leq d(y, x, c_j), \forall j \neq i \},
\]

for \( d(y, x, c_i) \geq d(y, x, c_j) \), \( i = 1, 2, \ldots, K \).

(4)

where \( d(y, x, c_i) \) is the geodesic distance between \( x \) and \( y \) on \( M_\zeta \). 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]:

\[
\min_{z \in M_\zeta} \int_{x \in V(c_i)} d^2(y, x)dx
\]

(5)

Let \( X = \{ x_i \}_{i=1}^N \) be an \( N \)-atom media (i.e., an image or a video) and \( M_\zeta = \Phi(X) \) the stretched manifold in \( \mathbb{R}^d \). We discretize manifold \( M_\zeta \) by a graph \( G = (V, E) \), 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 \( \Phi^{-1}(v_i) = \Phi^{-1}(v_j) \).

• \( n_\zeta = 8 \) 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 \( G \) by \( N = |V| \) and \( |E| \), respectively. Since \( G \) 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 $G$, 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 theoretical bound on the tessellation performance. Replacing geodesic metrics on $M_c$ 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 theoretical 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 $X = \{\Phi(p_i)\}_{i=1}^N$, $\mathcal{X} = \mathcal{M}_c$ and $D = d_{qg}$, and apply it to determine the initial positions of $K$ cluster centers $C = \{c_i\}_{i=1}^K$ on $\mathcal{M}_c$.
- Centroid-free Lloyd refinement (steps 9-17). For each center $c_i \in C$, we quickly find an alternative $\tau_i$ to it. If $\mathcal{E}(\{\tau_j\}_{j=1}^K) < \mathcal{E}(\{c_j\}_{j=1}^K)$, we replace $\{c_j\}_{j=1}^K$ by $\{\tau_j\}_{j=1}^K$ 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 $d_{qg}$ in the manifold space $\mathcal{M}_c$. By Corollary 1 in [42] (with the $K$-subset selection in [2]), the expected approximation ratio is bounded by $\frac{\mathbb{E}[\mathcal{E}(\{c_j\}_{j=1}^K)]}{\mathbb{E}[\mathcal{E}(\{\tau_j\}_{j=1}^K)]} \leq O(1)$. 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 $G$ 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.

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 $\text{first-in-first-out (FIFO) queue}$. 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 \( G \) 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 \( \times \) 321.

4.1. Properties

Since the queue in Algorithm 2 works in a FIFO manner, given an \( N \)-atom media \( X \) and a set of multiple sources \( C = \{c_i\}_{i=1}^n \subseteq V \), 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 \( G \) and is irrelevant to the media content. Each vertex in \( G \) has \( n_v \) 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 \( \Gamma = (N, W, S, E, NW, SW, SE, NE) \), 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 \( d_q(v, C) = \min_{c \in C} d_q(v, c) \). Based on the predefined traversal order \( \{v_1 = c_1, v_2 = c_2, \ldots, v_K = c_K, v_{K+1}, v_{K+2}, \ldots, v_N\} \), we assign an index \( I_i \) to each vertex \( v_i \in V \). Then the q-path \( \tilde{C}v_i = \{v_{I_1} = c, v_{I_2}, \ldots, v_{I_m} = v_i\} \) from a center \( c \in C \) to a vertex \( v_i \in V \setminus C \), satisfies that \( v_{I_a}, v_{I_b}, 1 \leq a < b \leq n, \) the indices \( I_{a} < I_{b} \).

If \( G \) is a regular lattice, i.e., all media atoms have the same color, any q-path is exactly the shortest path on \( G \). When the color variation in \( X \) is large, the manifold \( M_c \) 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 \( c \in C, v, v' \in V \setminus C \) and a shortest path \( \tilde{C}v = \{v_{I_1} = c, v_{I_2}, \ldots, v_{I_m} = v\} \) between \( c \) and \( v \) on \( G \), the q-path \( \tilde{C}v \) output from Algorithm 2 is exactly the shortest path \( \tilde{C}v \), if and only if \( \forall a, b, 1 \leq a < b \leq n' \), the indices \( I_{a} < I_{b} \).

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 \( \Omega(v_1), \Omega(v_2) \) and \( \Omega(v_f) \) of three vertices \( v_1, v_2 \) and \( v_f \) on the q-path \( c_1{\tilde{v}}_f \).

Algorithm 2 Computing q-paths and q-distances

**Input:** A sparse graph \( G = (V,E) \) discretizing manifold \( M_c \) and multiple sources \( C = \{c_i\}_{i=1}^n \in V \).

**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 \( v.dist \), a Boolean flag \( v.visit \) and a precedent node \( v.pre \).
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 \( G \) 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 \( \tilde{C}v = \{v_{I_1} = c, v_{I_2}, \ldots, v_{I_{m'}} = v\} \) between \( c \) and \( v \) on \( G \), the q-path \( \tilde{C}v \) output from Algorithm 2 is exactly the shortest path \( \tilde{C}v \), if and only if \( \forall i, 1 \leq i < n' \), the subpath \( \tilde{C}v_{i,j} \) of \( \tilde{C}v \) is contained in the allowable region of \( v_i \).

By Property 2, if at steps 8-10 of Algorithm 2, the q-distance value \( \Phi(v_b).dist \) is updated due to \( \Phi(v_a).dist > \Phi(v_b).dist + l(v_a, v_b) \) and the indices \( b < a \), then any q-path \( \tilde{C}v \) passing through \( v_b \) cannot be the shortest path on \( G \). 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 \( n_v \) 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
c to $v_a$ can be extended further to those neighbors with an index larger than $a$. Property 4 reveals that the number of these extendable neighbors are not smaller than $n_c/2$. As a comparison, in the Dijkstra’s algorithm, the path from $c$ to any $v$ can be extended to any one of $n_c - 1$ neighbors 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 $cv$ is not a shortest path.

**Observation 1.** If a vertex $v$ is wrongly labelled, then all the descendents vertices of $v$ in the tree are wrongly labelled. However, on bumpy manifold $M_c$, the number of descendents 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 $cv$ passes through a high-stretched region like the cliff of a mountain; e.g., $v_y$ and $v_6$ in Figure 5a. For any vertex $v'$ a little bit far beyond the high-stretched region (e.g., $v_{ij}$ in Figure 5a), by Property 3, it has a big chance that the q-path $v_{ij}v'$ can circumvent the high-stretched region and be the shortest path; e.g., the q-path $c_1v_{ij}$ in Figure 5a.

**Observation 2.** On the manifold $M_c$, 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 $cv$ goes through a boundary at $v$.

Figure 5. Since $\Phi$ is a one-to-one mapping, we can visualize the q-paths on both the media $X$ and the manifold $M_c = \Phi(X)$. 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 $c_1v_y$ is a shortest path $c_1v_y$, the vertex $v_y$ is shown in green; otherwise, $v_y$ is shown in red. On a q-path starting from $c_1$, if a vertex $v_y$ is red (e.g., $v_y \in c_1 v_a$ and $v_y \in c_1 v_b$), all the subsequent vertices are also red. (b) The corresponding q-paths on $M_c$. (c) On the q-path $c_1v_y$ which is also a shortest path, allowable regions $\Omega(v_y)$, $\Omega(v_i)$ 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 $c_2$.

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 $G$) 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 $cv'$ will be corrected by another path $c'v'$, 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 $M_c$ satisfies certain assumptions (characterized by the edge length ratio in $G$) and a moderately large number of centers are selected, the clustering $\{V_i\}_{i=1}^K$ on $G$ is exactly the same for using either shortest distance or q-distance. Figure 6 shows a real example.

---

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

3For example, $K \geq 200$ is sufficient for an image of $481 \times 321$ pixels.
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 \(\text{iter}_{\text{max}}\) 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], SUNRGBD [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 measures BR and UE, i.e., Boundary recall distance (BRD) \([28, 43]\), 3D under-segmentation error (UE3D) \([7, 23, 43]\). We also
use the compactness (CO) metric that measures the shape regularity of supervoxels. The results in Figure 8 are averaged on the BuffaloXiph dataset [5], 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. Levenshtein 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 authors6 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, 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 authors6 to compare five representative methods (GB, GBH, MeanShift, TSP and Yi-CSS) and our qd-CSS. Youtube-Objects dataset [30] (126 videos in 10 object classes) with foreground ground-truth, 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.

6www.cs.utexas.edu/~suyog/code_release_public.tar
7NCut 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


