Content-Sensitive Supervoxels via Uniform Tessellations on Video Manifolds

Ran Yi, Yong-Jin Liu*
Tsinghua University, China
{yr16,liuyongjin}@tsinghua.edu.cn

Yu-Kun Lai
Cardiff University, UK
Yukun.Lai@cs.cf.ac.uk

Abstract

Supervoxels are perceptually meaningful atomic regions in videos, obtained by grouping voxels that exhibit coherence in both appearance and motion. In this paper, we propose content-sensitive supervoxels (CSS), which are regularly-shaped 3D primitive volumes that possess the following characteristic: they are typically larger and longer in content-sparse regions (i.e., with homogeneous appearance and motion), and smaller and shorter in content-dense regions (i.e., with high variation of appearance and/or motion). To compute CSS, we map a video Ξ to a 3-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^6$, whose volume elements give a good measure of the content density in Ξ. We propose an efficient Lloyd-like method with a splitting-merging scheme to compute a uniform tessellation on $\mathcal{M}$, which induces the CSS in Ξ. Theoretically our method has a good competitive ratio $O(1)$. We also present a simple extension of CSS to stream CSS for processing long videos that cannot be loaded into main memory at once. We evaluate CSS, stream CSS and seven representative supervoxel methods on four video datasets. The results show that our method outperforms existing supervoxel methods.

1. Introduction

Supervoxels are perceptually meaningful atomic regions obtained by grouping similar voxels (i.e., exhibiting coherence in both appearance and motion) in the spatiotemporal domain, which over-segment a video while well preserving its structural content. Supervoxels can greatly reduce the computational complexity and have been widely used as a preprocessing step in many vision applications, such as video segmentation [12, 17, 34], foreground object segmentation [13], action segmentation and recognition [14, 21], spatiotemporal object detection [23], spatiotemporal closure in videos [15], and many others.

Depending on the size of video data, methods to compute supervoxels can be classified into *off-line* and *streaming* methods. Off-line methods require the video to be short enough such that all voxels can be loaded into the memory. On the other hand, streaming methods do not have such a limitation on the video length, i.e., video data is accessed sequentially in blocks and the memory available to streaming methods only needs to fit a block. Many methodologies, such as clustering, hierarchical, generative, statistical and graph partitioning methods, have been applied to compute supervoxels; see an excellent survey in [31]. Xu and Corso [31] further select seven representative methods, including five off-line [8, 9, 24, 6, 12] and two streaming [32, 4] methods, to represent the state of the art.

To measure the quality of supervoxels, the following principles are taken into consideration: (1) *Feature preservation*: supervoxels align well with object boundaries in a video; (2) *Spatiotemporal uniformity*: in non-feature regions, supervoxels are uniform and regular in the spatiotemporal domain; (3) *Performance*: computing supervoxels is time-and-space efficient and scales well with large video data; (4) *Easy to use*: users simply specify the desired number of supervoxels and should not be bothered by tuning other parameters; (5) *Parsimony*: the above principles should be maintained with as few supervoxels as possible.

So far, none of existing methods satisfy all these principles. In this paper, we propose a content-sensitive approach to address the *parsimony* principle, and therefore, achieve a good balance among all principles. Our method is motivated by an important observation: the scene layouts and motions of different objects in a video usually exhibit large diversity, and thus the density of video content often varies significantly in different parts of the video. Applying spatiotemporally uniform distribution of supervoxels indiscriminately to the whole video often leads to under-segmentation in content-dense regions (i.e., with high variation of appearance and/or motion), and over-segmentation in content-sparse regions (i.e., with homogeneous appearance and motion). Therefore, computing supervoxels adaptively with respect to the density of video content can achieve the best performance (see Figure 1).

To compute *content-sensitive supervoxels* (CSS), we extend the image manifold concept [19, 20] to a video man-

*Corresponding author

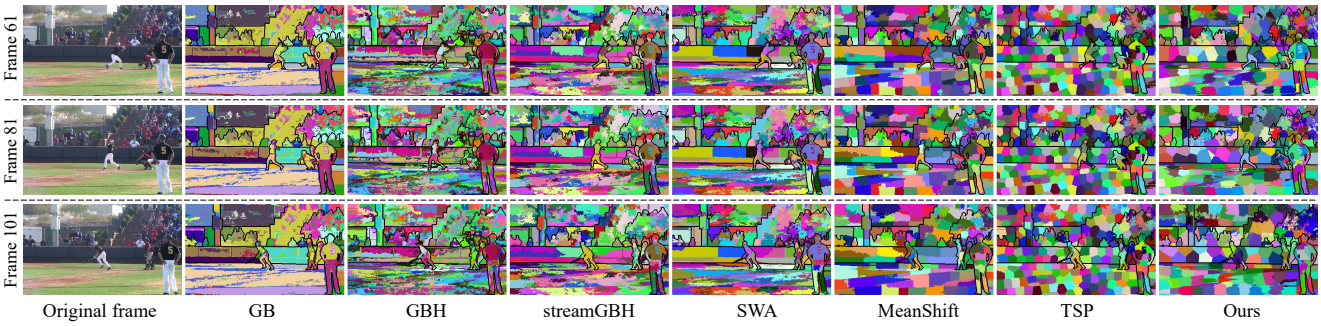


Figure 1. Superpixels (induced by clipping supervoxels on frames #61, #81 and #101) obtained by GB [8], GBH [12], streamGBH [32], SWA [25, 26, 6], MeanShift [24], TSP [4] and our CSS method. All methods generate approximately 500 supervoxels. MeanShift, TSP and our CSS method produce regular supervoxels (and accordingly regular clipped superpixels), while other methods produce highly irregular supervoxels. As shown in Section 5, TSP and CSS methods outperform other methods in terms of commonly used quality metrics pertaining to supervoxels, including UE3D, SA3D, BRD and EV, while CSS method runs $5\times$ to $10\times$ faster than TSP and the peak memory required by CSS is $22\times$ smaller than TSP. Compared to TSP, CSS generates more supervoxels in content-rich areas (e.g., auditorium) and fewer supervoxels in content-sparse areas (e.g., concrete fences and meadow).

ifold, which represents a video Ξ as a 3-manifold $\mathcal{M}$ embedded in the combined color and spatiotemporal space $\mathbb{R}^6$. The volumetric elements in $\mathcal{M}$ give a good measure of content density in Ξ and thus a uniform tessellation on $\mathcal{M}$ efficiently induces CSS in Ξ.

Two main contributions are made in this paper:

First, simply treating the time dimension in a video in the same way as spatial dimensions leads to a regular 3D lattice representation of voxels in $\mathbb{R}^3$, which is not optimal due to possibly many non-negligible motions and occlusions/disocclusions. To overcome this drawback, existing methods (e.g., [12, 4]) use optical flow to re-establish a connection graph of neighboring voxels between adjacent frames. However, state-of-the-art optical flow estimation methods [28] are still imperfect and may introduce extra errors into supervoxel computation. In our work, we distort the regular 3D lattice structure of videos in the spatiotemporal domain $\mathbb{R}^3$ by mapping it to a curved 3-manifold $\mathcal{M}$ embedded in a high-dimensional combined color and spatiotemporal space $\mathbb{R}^6$, in which the Euclidean distance is a simple and efficient metric to generate CSS.

Second, to quickly compute a high-quality uniform tessellation on the video manifold $\mathcal{M} \subset \mathbb{R}^6$, we propose a splitting-merging scheme that can be efficiently incorporated into the well known K-means++ algorithm [2, 30]. Our scheme has a theoretical constant-factor bi-criteria approximation guarantee, and in practice makes our method obtain good CSS in very few iterations. By applying the streaming version of K-means++ (a.k.a. K-means# [1]), our method can be easily extended to process long videos that cannot be loaded into main memory at once.

2. Preliminaries

Our method uses *restricted centroidal Voronoi tessellation* (RCVT) [19] to compute a uniform tessellation of a 3-manifold $\mathcal{M} \subset \mathbb{R}^6$. Theoretically our method is a *bi-criteria approximations* to the K-means problem [2, 1, 30]. We briefly introduce them before presenting our method.

2.1. Restricted Voronoi tessellation and RCVT

In [19], RCVT is proposed to uniformly tessellate a 2-manifold in $\mathbb{R}^5$. With a simple extension of the proofs therein, we have the following general results.

Let $S_K = \{s_i\}_{i=1}^K$ be a set of generating points and $\mathcal{M}$ be an $l(< d)$-dimensional manifold in $\mathbb{R}^d$ $(d > 2)$. The Euclidean Voronoi cell of a generator s_i in $\mathbb{R}^d$, denoted by $C_{\mathbb{R}^d}$, is

$$C_{\mathbb{R}^d}(s_i) = \{x \in \mathbb{R}^d : \|x - s_i\|_2 \leq \|x - s_j\|_2, \ \forall j \neq i, s_j \in S_K\}. \quad (1)$$

The restricted Voronoi cell $C_{\mathcal{M}}$ is defined to be the intersection of $C_{\mathbb{R}^d}$ and $\mathcal{M}$

$$C_{\mathcal{M}}(s_i) \triangleq \mathcal{M} \cap C_{\mathbb{R}^d}(s_i), \quad (2)$$

and the restricted Voronoi tessellation $RVT(S_K, \mathcal{M})$ is the collection of restricted Voronoi cells satisfying

$$RVT(S_K, \mathcal{M}) = \{C_{\mathcal{M}}(s_i) \neq \emptyset, \forall s_i \in S_K\}. \quad (3)$$

$RVT(S_K, \mathcal{M})$ is a finite closed covering of $\mathcal{M}$, i.e., $\mathcal{M} = \bigcup_{s_i \in S_K} C_{\mathcal{M}}(s_i)$.

The mass centroid of the cell $C_{\mathcal{M}}(s_i)$ is

$$m_i = \frac{\int_{x \in C_{\mathcal{M}}(s_i)} x\rho(x)dx}{\int_{x \in C_{\mathcal{M}}(s_i)} \rho(x)dx} \quad (4)$$

where ρ is a density function on $C_{\mathcal{M}}$. An $RVT(S_K, \mathcal{M})$ is a restricted centroidal Voronoi tessellation (RCVT) if and only if each generator $s_i \in S_K$ is the mass centroid of $C_{\mathcal{M}}(s_i)$.

Theorem 1. *[19] Let $\mathcal{M}$ be an l-manifold embedded in $\mathbb{R}^d$, $d > l \geq 2$, and $K \in \mathbb{Z}_+$ be a positive integer. For an arbitrary set S_K of points $\{s_i\}_{i=1}^K$ in $\mathbb{R}^d$ and an arbitrary tessellation $\{C_i\}_{i=1}^K$ on $\mathcal{M}$, $\bigcup_{i=1}^K C_i = \mathcal{M}$, $C_i \bigcap C_j = \emptyset$, $\forall i \neq j$, define the tessellation energy functional as follows:*

$$\mathcal{E}(\{(s_i, C_i)\}_{i=1}^K) = \sum_{i=1}^{K} \int_{x \in C_i} \|x - s_i\|_2^2 dx \qquad (5)$$

Then the necessary *condition for $\mathcal{E}$ to be minimized is that $\{(s_i, C_i)\}_{i=1}^K$ is an RCVT of $\mathcal{M}$.*

Theorem 1 indicates that RCVT is a uniform tessellation on $\mathcal{M}$, which minimizes the energy $\mathcal{E}$.

2.2. Bi-criteria approximation algorithms

The discretized counterpart of RCVT is the solution to the K-means problem on the manifold domain $\mathcal{M}$. Given a fixed K, denote by $S_K^{opt} = \{s_i^{opt}\}_{i=1}^K$ and $\{C_i^{opt}\}_{i=1}^K$ the (unknown) optimal generator set and tessellation on $\mathcal{M}$ respectively, which minimize the energy $\mathcal{E}$. Let S_K and $\{C_i\}_{i=1}^K$ be the generator set and tessellation output from an algorithm $\mathcal{A}$. The competitive ratio is defined to be the worst-case ratio $r = \frac{\mathcal{E}(\{(s_i, C_i)\}_{i=1}^K)}{\mathcal{E}(\{(s_i^{opt}, C_i^{opt})\}_{i=1}^K)}$ and $\mathcal{A}$ is said to be a b-approximation algorithm if $r \leq b$. An algorithm is called (a, b)-approximation, if it outputs $\{(s_i, C_i)\}_{i=1}^{aK}$ with aK generators and tessellation cells, such that $r = \frac{\mathcal{E}(\{(s_i, C_i)\}_{i=1}^{aK})}{\mathcal{E}(\{(s_i^{opt}, C_i^{opt})\}_{i=1}^K)} \leq b$, where $a > 1$ and $b > 1$.

3. Overview

Our method relies on a *video manifold* representation. Denote by Ξ an input video with N voxels. Each voxel is represented by $v(x, y, t)$, where (x, y) is the pixel location and t the frame index. Inspired by the success of image manifolds [19, 20], we represent the color in the CIELAB color space. Let $c(v) = (l(v), a(v), b(v))$ be the color at the voxel v. We define a video manifold $\mathcal{M}$ using a stretching map $\Phi : \Xi \to \mathcal{M} \subset \mathbb{R}^6$ that maps all voxels in Ξ into a 3-manifold $\mathcal{M}$ embedded in the 6-dimensional space:

$$\Phi(v) = (\lambda_1 x, \lambda_1 y, \lambda_2 t, \lambda_3 l(v), \lambda_3 a(v), \lambda_3 b(v)), \qquad (6)$$

where we follow [20] to set global stretching factors $\lambda_1 = \lambda_2 = 0.435$ and $\lambda_3 = 1$.

For each voxel $v(x, y, t)$, denote by $\boxdot_v$ the unit cube (i.e., of size $1 \times 1 \times 1$) centered at v. Refer to Figure 2. Let $a_1, a_2, \cdots, a_8$ be eight corner points of $\boxdot_v$, each of which is determined by an average of its eight

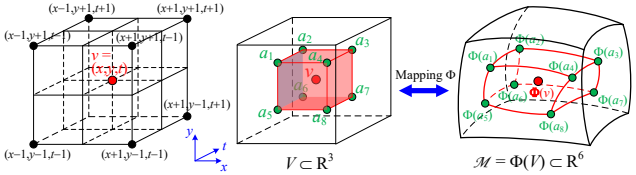


Figure 2. The stretching map $\Phi : \Xi \to \mathcal{M} \subset \mathbb{R}^6$ maps the unit cube $\boxdot_v$ (red box) centered at the voxel $v(x, y, t) \in \Xi$ into a 3-manifold $\mathcal{M}$. Each corner a_i of $\boxdot_v$, $i = 1, 2, \cdots, 8$, is the center of its eight neighboring voxels; e.g., $a_1 = \frac{\sum_{i=-1}^{0}\sum_{j=0}^{1}\sum_{k=-1}^{0} v(x+i, y+j, t+k)}{8}$.

neighboring voxels. We decompose $\boxdot_v$ into six non-overlapped tetrahedrons tet_i, $i = 1, 2, \cdots, 6$, denoted by $\{(a_5, a_6, a_3, a_2)$, (a_5, a_6, a_3, a_7), (a_5, a_8, a_3, a_4), (a_5, a_8, a_3, a_7), (a_5, a_1, a_3, a_2), $(a_5, a_1, a_3, a_4)\}$. We approximate the volume of the *curved* tetrahedron $\Phi(tet(a, b, c, d))$ in $\mathcal{M}$ as

$$V(\Phi(tet(a, b, c, d))) \approx \frac{1}{3} A(\Phi(\triangle_{abc})) \cdot dist(d, \triangle_{abc}) \qquad (7)$$

where $dist(d, \triangle_{abc})$ is the distance from d to the subspace spanned by vectors a, b and c, and $A(\Phi(\triangle_{abc}))$ is the area of the *curved* triangle $\Phi(\triangle_{abc})$, which is approximated by

$$A(\Phi(\triangle_{abc}))) \approx \frac{1}{2} \|\overrightarrow{\Phi(a)\Phi(b)}\|_2 \|\overrightarrow{\Phi(a)\Phi(c)}\|_2 \sin\theta \qquad (8)$$

θ is the angle between vectors $\overrightarrow{\Phi(a)\Phi(b)}$ and $\overrightarrow{\Phi(a)\Phi(c)}$, and $\|u\|_2$ is the Euclidean length of the vector u in $\mathbb{R}^6$. Using the least squares method, $dist(d, \triangle_{abc})$ can be efficiently obtained as the length of the residual vector r:

$$r = d - L(L^T L)^{-1} L^T d, \qquad (9)$$

where L is a 6×3 matrix $L = (a\ b\ c)$, and a, b, c, d are column 6-vectors. Then the volume of $\Phi(\boxdot_v) \subset \mathcal{M}$ is approximated by

$$V(\Phi(\boxdot_v)) = \sum_{i=1}^{6} V(\Phi(tet_i)) \qquad (10)$$

For any region $\Omega \subset \Xi$, the volume of $\Phi(\Omega) \subset \mathcal{M}$ is simply the sum $\Sigma_{v_j \in \Omega} V(\Phi(\boxdot_{v_j}))$. We assume the density $\rho \equiv 1$ everywhere in $\mathcal{M}$.

Our method is based on an important characteristic in the video manifold $\mathcal{M}$: the volume of a region $\Phi(\Omega) \subset \mathcal{M}$ depends on both the volume of $\Omega \subset \Xi$ and the color variation in Ω. The higher variation of colors in Ω, the larger the volume of $\Phi(\Omega)$ and vice versa. We propose an algorithm in Section 4, which is theoretically a constant-factor bi-criteria approximation, to quickly and efficiently compute a uniform tessellation $\{(s_i, C_i)\}_{i=1}^K$ in $\mathcal{M}$. Then the inverse mapping Φ^{-1} will transform $\{C_i\}_{i=1}^K$ into *content-sensitive* supervoxels in Ξ. See Figure 3 for an illustration.

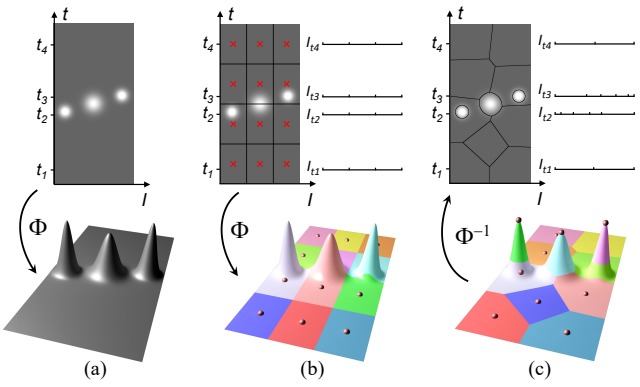


Figure 3. Overview of CSS generation on a synthetic, degenerate gray video Ξ, for easy illustration. In Ξ, the image frame at time t is a degenerate (1D) gray line image I_t. Supervoxels in Ξ can be regarded as a tessellation of Ξ. Clipping supervoxels in each line image I_t results in a set of induced superpixels in I_t. (a) We represent Ξ as a video manifold $\mathcal{M} = \Phi(I, t) \subset \mathbb{R}^3$, whose area elements give a good measure of content density in Ξ. (b) A uniform tessellation of 12 generating points on the domain of Ξ leads to a non-uniform tessellation on $\mathcal{M}$. The induced superpixels on line images I_{t_1}, I_{t_2}, I_{t_3} and I_{t_4} are also uniform. (c) We generate CSS by computing a uniform tessellation $\{(s_i, C_i)\}_{i=1}^{12}$ on $\mathcal{M}$ and the induced superpixels on line images are also content-sensitive. The real examples of induced superpixels on video frames are illustrated in Figure 1.

4. $(O(1), O(1))$-Approximation Algorithm

To obtain a uniform tessellation $\{(s_i, C_i)\}_{i=1}^{K}$ in $\mathcal{M}$, our algorithm consists of the following two steps:

- Initialization (Section 4.1). We apply a variant of the K-means++ algorithm[1] [2, 1, 30] to determine the initial positions of generating points $S_K = \{s_i\}_{i=1}^{K}$.
- Lloyd refinement (Section 4.2). Observing that the classic Lloyd method converges only to a local minimum with a large number of iterations, we propose an efficient splitting-merging scheme to compute $RCVT(S_K, \mathcal{M})$, which helps move the solution out of local minima and ensures a good competitive ratio.

Our algorithm is easy to implement and can obtain high-quality CSS in very few iterations. Theoretically our algorithm is $(O(1), O(1))$-approximation. To ease reading, all proofs in this paper are presented in supplemental material.

4.1. Initialization

We apply a variant of K-means++ algorithm to obtain a provable high-quality initialization of generating points $S_K = \{s_i\}_{i=1}^{K}$. The pseudo-code is summarized in Algorithm 1. In each step, a point in $\mathcal{M}$ is picked up with probability proportional to its current score (defined as its squared distance to the nearest generator picked so far), and added as a new generator. To compute the required probability in the manifold domain $\mathcal{M}$, we consider the positions of mapped voxels $\Phi(v) \in \mathcal{M}$, $\forall v \in \Xi$. With respect to an existing generator $\Phi(v_i)$, the score of a mapped voxel $\Phi(v_j) \in \mathcal{M}$, $j \neq i$, is

$$\begin{aligned} p_{v_i}(v_j) &= \int_{x \in \Phi(\square_{v_j})} \|x - \Phi(v_i)\|_2^2 dx \\ &\approx V(\Phi(\square_{v_j})) \cdot \|\Phi(v_j) - \Phi(v_i)\|_2^2 \end{aligned} \tag{11}$$

Then the score of picking $\Phi(v_j)$ with respect to an existing generator set S is

$$p_S(v_j) = \min_{v_i \in S} p_{v_i}(v_j) \tag{12}$$

Algorithm 1 runs in $O(NK)$ time. A simple adaption of the proofs in [2, 30] shows that using $RVT(S_K, \mathcal{M})$ as the tessellation, Algorithm 1 is an expected $\Theta(\log K)$-approximation algorithm, and furthermore, if we sample βK ($\beta > 1$) generators, the expected approximation ratio of Algorithm 1 is bounded by

$$\mathbb{E}(r) < 8\left(1 + \frac{1+\sqrt{5}}{2(\beta - 1)}\right). \tag{13}$$

Algorithm 1 Initialization

Input: A video Ξ of N voxels and the desired number of supervoxels K.

Output: The initial positions of K generating points $S_K = \{s_i\}_{i=1}^{K}$.

1: Compute $V(\Phi(\square_v))$ for each voxel $v \in \Xi$ (Eq.(10)).
2: Choose a point v_1 from all voxels $v \in \Xi$ with probability proportional to $V(\Phi(\square_v))$.
3: Set $s_1 = \Phi(v_1)$, $S_1 = \{s_1\}$ and $i = 1$.
4: **while** $i < K$ **do**
5: Choose a point v_{i+1} from all voxels $v \in \Xi$ with probability proportional to the score $p_{S_i}(v)$ (Eq.(12)).
6: Set $s_{i+1} = \Phi(v_{i+1})$, $S_{i+1} = S_i \cup \{s_{i+1}\}$ and $i = i + 1$.
7: **end while**

4.2. Lloyd refinement with splitting and merging

Given the initial generators $S_K = \{s_i\}_{i=1}^{K}$, $s_i \in \mathcal{M}$, the classic Lloyd method computes $RCVT(S_K, \mathcal{M})$ iteratively by alternating the following two steps:

- Step 1: Fixing the generator set S_K, compute $RVT(S_K, \mathcal{M})$;
- Step 2: For each cell $C_{\mathcal{M}}$ in $RVT(S_K, \mathcal{M})$, update its generator to be the mass centroid of $C_{\mathcal{M}}$.

[1] In literature, K-means++ algorithm includes a seeding step and a Lloyd refinement step. In our method, we only use the seeding step.

Algorithm 2 CSS generation

Input: A video Ξ of N voxels, the desired number of supervoxels K, num_{random} the number of repeated random sampling of generators in each iteration, and the maximum number of iterations $iter_{max}$.

Output: K content-sensitive supervoxels.

1: Initialize the generators $S_K = \{s_i\}_{i=1}^K$ (Algorithm 1).
2: Set $iter = 0$.
3: **while** $iter < iter_{max}$ **do**
4: Compute $RVT(S_K, \mathcal{M})$.
5: Set $n = 0$.
6: **while** $n < num_{random}$ **do**
7: Randomly pick three generators s_m, s_i, s_j in S_k, in which s_i and s_j are neighbors (Algorithm 4).
8: Check the splitting-merging feasibility of (s_m, s_i, s_j) (Algorithm 3) and put the return values in $(Flag, s'_p, s'_q, s'_k)$.
9: **if** $Flag == TRUE$ **then**
10: Update S_K by splitting s_m into (s'_p, s'_q) and merging (s_i, s_j) into s'_k.
11: **end if**
12: $n = n + 1$;
13: **end while**
14: Compute $RVT(S_K, \mathcal{M})$.
15: **for** each cell $C_{\mathcal{M}}(s_i)$ in RVT **do**
16: Update the generator s_i to be the mass centroid of $C_{\mathcal{M}}(s_i)$.
17: **end for**
18: $iter = iter + 1$;
19: **end while**
20: Compute $\Phi^{-1}(RVT(S_K, \mathcal{M}))$ to obtain K CSS.

This method converges only to a local minimum with a large number of iterations [7].

To compute a high quality $RCVT(S_K, \mathcal{M})$ in very few iterations, we propose a splitting-merging scheme in the Lloyd iteration. The pseudo-code is summarized in Algorithm 2, in which line 4 and lines 15-16 correspond to Steps 1 and 2 in the classic Lloyd method respectively, and lines 5-14 summarize our splitting-merging scheme.

The splitting operation $S : s_m \to (s'_p, s'_q)$ splits a cell $C_{\mathcal{M}}(s_m)$ into two new cells $C(s'_p)$ and $C(s'_q)$. Conversely, the merging operation $M : (s_i, s_j) \to s'_k$ merges two cells $C_{\mathcal{M}}(s_i)$ and $C_{\mathcal{M}}(s_j)$ into a new cell $C(s'_k)$. Splitting reduces the tessellation energy $\mathcal{E}$ and merging increases it. The number of generators does not change by applying a pair of splitting and merging operations $(S, M) : (s_m, (s_i, s_j)) \to ((s'_p, s'_q), s'_k)$. Our goal is to design a pair (S, M) such that $\mathcal{E}$ does not increase.

Definition 1. *The diameter d_i of a cell $C_{\mathcal{M}}(s_i)$, $s_i \in S_K$, is the maximum Euclidean distance between pairs of points in the cell, i.e.,*

$$d_i = \max_{\forall x,y \in C_{\mathcal{M}}(s_i)} \|x - y\|_2 \tag{14}$$

Denote by $p_1(d_i)$ and $p_2(d_i)$ the two points in $C_{\mathcal{M}}(s_i)$ satisfying $\|p_1(d_i) - p_2(d_i)\| = d_i$.

Algorithm 3 Check splitting-merging feasibility

Input: Three generators (s_m, s_i, s_j) in S_K and an $RVT(S_K, \mathcal{M})$.

Output: A Boolean variable $Flag$ indicating the feasibility and three new generators (s'_p, s'_q, s'_k).

1: Compute the mass centroids s'_m, s'_i and s'_j of $C_{\mathcal{M}}(s_m)$, $C_{\mathcal{M}}(s_i)$ and $C_{\mathcal{M}}(s_j)$, respectively.
2: Compute the diameter d_m of the cell $C_{\mathcal{M}}(s_m)$ and the points $p_1(d_m)$ and $p_2(d_m)$ (see Definition 1).
3: Compute two new cells $C'(p_1(d_m))$ and $C'(p_2(d_m))$, which are the Voronoi cells of $p_1(d_m)$ and $p_2(d_m)$ in the domain $C_{\mathcal{M}}(s_m)$.
4: Compute the mass centroids s'_k, s'_p and s'_q of $C_{\mathcal{M}}(s_i) \cup C_{\mathcal{M}}(s_j)$, $C'(p_1(d_m))$ and $C'(p_2(d_m))$, respectively.
5: Compute $\tau_{m,i,j}$ in Eq. (15).
6: **if** $\|s'_p - s'_m\|_2 > \tau_{m,i,j}$ and $\|s'_q - s'_m\|_2 > \tau_{m,i,j}$ **then**
7: **return** $TRUE$ and (s'_p, s'_q, s'_k).
8: **else**
9: **return** $FALSE$ and $(NULL, NULL, NULL)$.
10: **end if**

Theorem 2. *Let s_m, s_i, s_j be three generators in an $RVT(S_K, \mathcal{M})$. For the cells $C_{\mathcal{M}}(s_m)$, $C_{\mathcal{M}}(s_i)$ and $C_{\mathcal{M}}(s_j)$, let m_m, m_i, m_j be their masses, s'_m, s'_i, s'_j be their mass centroids, respectively. For any partitioning of $C_{\mathcal{M}}(s_m)$ into two new cells $C'(p_1(d_m))$ and $C'(p_2(d_m))$, which satisfies $p_1(d_m) \in C'(p_1(d_m))$, $p_2(d_m) \in C'(p_2(d_m))$, $C'(p_1(d_m)) \cap C'(p_2(d_m)) = \emptyset$ and $C'(p_1(d_m)) \cup C'(p_2(d_m)) = C_{\mathcal{M}}(s_m)$, let s'_k, s'_p and s'_q be the mass centroids of $C_{\mathcal{M}}(s_i) \cup C_{\mathcal{M}}(s_j)$, $C'(p_1(d_m))$ and $C'(p_2(d_m))$, respectively. If $\|s'_p - s'_m\|_2 > \tau_{m,i,j}$ and $\|s'_q - s'_m\|_2 > \tau_{m,i,j}$, where*

$$\tau_{m,i,j} = \sqrt{\frac{m_i m_j}{m_m(m_i + m_j)}} \|s'_i - s'_j\|_2 \tag{15}$$

then the pair of operations $(S, M) : (s_m, (s_i, s_j)) \to ((s'_p, s'_q), s'_k)$ does not increase the tessellation energy $\mathcal{E}$.

By Theorem 2, we check the splitting-merging feasibility condition at line 8 of Algorithm 2 and this condition is summarized in Algorithm 3. Note that computing the diameter of an arbitrary region (line 2 of Algorithm 3) is time-consuming. In practice, we compute the axis-aligned bounding box B of $C_{\mathcal{M}}(s_m)$. B is determined by two supporting points p_1 and p_2 in $C_{\mathcal{M}}(s_m)$ and we use them as fast approximations to $p_1(d_m)$ and $p_2(d_m)$.

Algorithm 4 Randomly pick three generators

Input: An $RVT(S_K, \mathcal{M})$ and an expected cell volume $\mathbb{E}(V(C_{\mathcal{M}}))$ (Eq.(18)).
Output: Three generators (s_m, s_i, s_j) in S_K.

Set $S_{dense} = \emptyset$ and $S_{sparse} = \emptyset$.
for each cell $C_{\mathcal{M}}(s_i)$ in $RVT(S_K, \mathcal{M})$ **do**
  Compute the volume $V(C_{\mathcal{M}}(s_i))$.
  if $V(C_{\mathcal{M}}(s_i)) > 4\mathbb{E}(V(C_{\mathcal{M}}))$ **then**
    $S_{dense} = S_{dense} \cup \{s_i\}$.
  else if $V(C_{\mathcal{M}}(s_i)) < \mathbb{E}(V(C_{\mathcal{M}}))/4$ **then**
    $S_{sparse} = S_{sparse} \cup \{s_i\}$.
  end if
end for
if $|S_{dense}| > 2$ **then**
  Randomly pick a generator s_m in S_{dense}.
else
  Randomly pick a generator s_m in S_K.
end if
Collect all neighboring pairs in S_{sparse} in the set $\mathcal{N}$.
if $|\mathcal{N}| > 2$ **then**
  Randomly pick a pair (s_i, s_j) in $\mathcal{N}$.
else
  Randomly pick two neighboring generators s_i and s_j in S_K.
end if
return (s_m, s_i, s_j).

Algorithm 5 Streaming CSS generation

Input: A video Ξ of N voxels and the desired number of supervoxels K.
Output: K content-sensitive supervoxels.

Compute the discretized manifold representation $\widetilde{\mathcal{M}} = \{(x_i, y_i, t_i, w_i)\}_{i=1}^{N}$.
Initialize $S = \widetilde{\mathcal{M}}$.
while S cannot be loaded into main memory **do**
  Set $\widetilde{S} = \emptyset$.
  Divide S into l disjoint pieces $\chi_1, \cdots, \chi_l$, such that each piece can be loaded into main memory.
  for each piece χ_i **do**
    Apply Algorithm 2 to compute $1.2K$ generators $S_K(\chi_i)$.
    Compute $RVT(S_K(\chi_i), \chi_i)$.
    for each new generator g_j in $S_K(\chi_i)$ **do**
      Compute the total weight of all points in the cell corresponding to g_j in $RVT(S_K(\chi_i), \chi_i)$ and assign it to g_j as the weight w_j;
    end for
    $\widetilde{S} = \widetilde{S} \cup (g_j, w_j), \forall g_j \in S_K(\chi_i)$.
  end for
  $S = \widetilde{S}$.
end while
Apply Algorithm 2 to S for obtaining K supervoxels.

Corollary 1. *Denote a voxel in the video Ξ as v_i and let*

$$m_{\min} = \min_{v_i \in \Xi} \{V(\Phi(\boxdot_{v_i}))\}, \tag{16}$$

$$d_{\max} = \max_{v_i \in \Xi} \{d(\Phi(\boxdot_{v_i}))\}, \tag{17}$$

where $d(\Phi(\boxdot_{v_i}))$ is the diameter of $\Phi(\boxdot_{v_i}) \subset \mathcal{M}$. Let s_m, s_i, s_j be three generators in an $RVT(S_K, \mathcal{M})$ and s'_m be the mass centroid of $C_{\mathcal{M}}(s_m)$. Let P be the hyperplane which passes through s'_m and is perpendicular to the line connecting $p_1(d_m)$ and $p_2(d_m)$. P partitions $C_{\mathcal{M}}(s_m)$ into $C'(p_1(d_m))$ and $C'(p_2(d_m))$. If $d_m \geq w(\frac{m_m}{m_{\min}}\tau_{m,i,j} + d_{\max})$, where $m_m = V(C_{\mathcal{M}}(s_m))$, $w = \max\{1+\lambda, \frac{1+\lambda}{\lambda}\}$, $\lambda = \frac{\|p_1(d_m) - s'_m\|_2}{\|p_2(d_m) - s'_m\|_2}$ and $\tau_{m,i,j}$ is defined in Eq.(15), then the pair of operations $(S, M) : (s_m, (s_i, s_j)) \rightarrow ((s'_p, s'_q), s'_k)$ does not increase the tessellation energy $\mathcal{E}$, where s'_p, s'_q and s'_k are the mass centroids of $C'(p_1(d_m))$, $C'(p_2(d_m))$ and $C_{\mathcal{M}}(s_i) \cup C_{\mathcal{M}}(s_j)$, respectively.

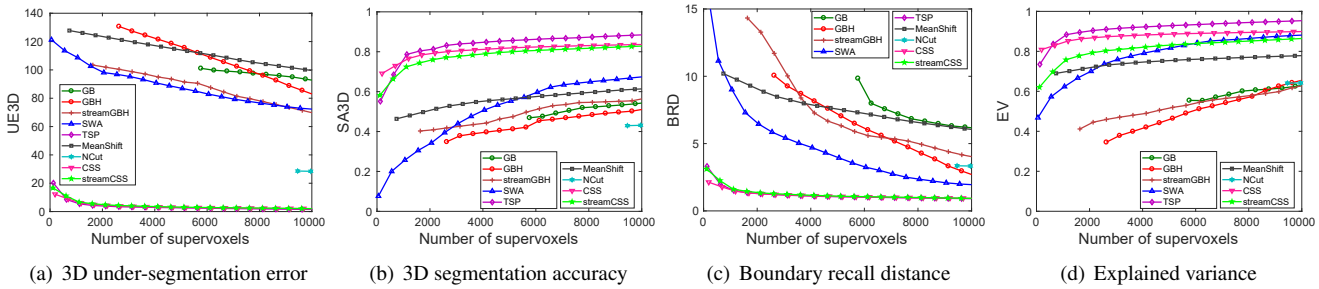
Note that for a region $\Omega \subset \Xi$ with a fixed volume, the higher variation of colors in Ω, the larger the volume of $\Phi(\Omega) \subset \mathcal{M}$ and vice versa. In Algorithm 2, s_i and s_j are chosen to be neighboring generators. Then by Corollary 1, Algorithm 2 has the following characteristics:

- the smaller the volumes of cells $C_{\mathcal{M}}(s_i)$ and $C_{\mathcal{M}}(s_j)$, the more likely they are merged, thus reducing the number of generators in content-sparse regions;
- the larger the volume of a cell $C_{\mathcal{M}}(s_m)$, the more likely it is split, thus producing more generators in content-rich regions.

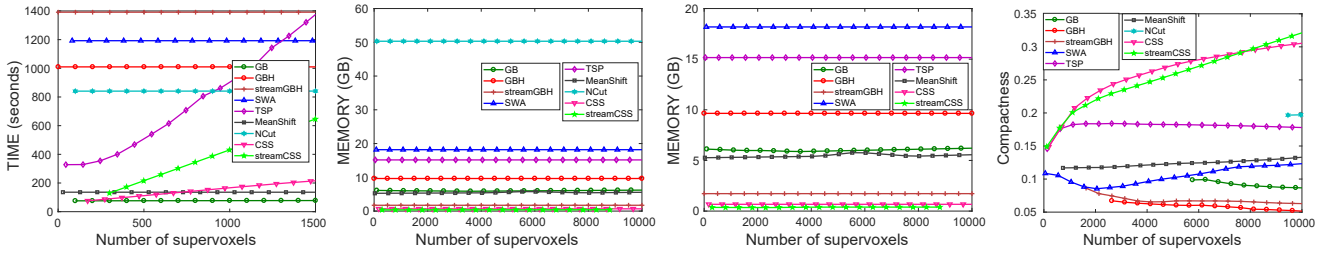
Accordingly, to increase the feasibility of the splitting-merging operation at line 8 of Algorithm 2, we estimate content-dense and content-sparse regions in $RVT(S_K, \mathcal{M})$ and collect their corresponding generators into subsets S_{dense} and S_{sparse} in Algorithm 4. If S_{dense} and S_{sparse} contain sufficient generators, we randomly pick two neighboring generators in S_{sparse} to be merged and pick one generator in S_{dense} to be split; otherwise, we randomly pick three generators in S_K. To estimate the content density of cells, we compute the expected cell volume as the average of K cells over the total volume of video manifold $\mathcal{M}$:

$$\mathbb{E}(V(C_{\mathcal{M}})) = \frac{\sum_{v \in \Xi} V(\Phi(\boxdot_v))}{K} \tag{18}$$

For each cell $C_{\mathcal{M}}$ in $RVT(S_K, \mathcal{M})$, we compare its volume $V(C_{\mathcal{M}})$ with $\mathbb{E}(V(C_{\mathcal{M}}))$: (1) if $V(C_{\mathcal{M}}) > 4\mathbb{E}(V(C_{\mathcal{M}}))$, we put the generator of this cell into S_{dense}, and (2) if $V(C_{\mathcal{M}}) < \mathbb{E}(V(C_{\mathcal{M}}))/4$, we put the generator of this cell into S_{sparse}. Algorithm 4 summarizes the pseudo-code.



(a) 3D under-segmentation error (b) 3D segmentation accuracy (c) Boundary recall distance (d) Explained variance



(e) Runtime with respect to K (f) Peak memory with respect to K (g) Peak memory without NCut (h) Compactness

Figure 4. Evaluation of seven representative methods and our methods (CSS and streamCSS) on the BuffaloXiph dataset. Due to its high computational cost, NCut is run at a fixed frame resolution of 240×160 downsampled from original videos. CSS, streamCSS and TSP have the best performance in terms of UE3D, SA3D, BRD and EV. CSS and steamCSS is better than TSP in terms of compactness, time and space efficiency. Similar performances are observed on the other three video datasets (SegTrack v2, BVDS and CamVid), which are reported in supplemental material.

In all our experiments, we set $iter_{max} = 20$ and $num_{random} = 20$ in Algorithm 2. We show in Section 5 that our algorithm can obtain high-quality CSS in 20 iterations. We have the following theoretical results.

Theorem 3. *By selecting $(1+\varepsilon)K$ generators, $\varepsilon > 0$, Algorithm 2 is a bi-criteria $\left(1+\varepsilon, 8\left(1+\frac{1+\sqrt{5}}{2\varepsilon}\right)\right)$-approximation algorithm in expectation.*

Theorem 4. *By selecting $(1+\varepsilon)K$ generators, $0 < \varepsilon < 1$, the time and space complexities of Algorithm 2 are $O(NK)$ and $O(N+K)$, respectively.*

4.3. Streaming CSS algorithm for long videos

Thanks to the streaming K-means algorithm [1], Algorithm 2 is readily extended to a streaming version for handling long videos. The streaming CSS algorithm represents the video manifold $\mathcal{M}$ by an ordered, discretized sequence of weighted points $\widetilde{\mathcal{M}} = \{(x_i, y_i, t_i, w_i)\}_{i=1}^{N}$, where (x_i, y_i, t_i) is the position of voxel v_i in Ξ and $w_i = V(\Phi(\boxdot_v))$. Pseudo-code is summarized in Algorithm 5.

Theorem 5. *If $(1+\varepsilon)K$ generators, $0 < \varepsilon < 1$, are selected by Algorithm 2, Algorithm 5 is $(O(1), O(1))$-approximation.*

5. Experiments

We implemented CSS (Algorithm 2) and streamCSS (Algorithm 5) in C++ and tested them on a PC with Intel Core E5-2683V3 and 256GB RAM running Linux. Source code is available[2]. We compare CSS and streamCSS with seven representative methods selected in [31], including NCut [27, 10, 9], SWA [25, 26, 6], MeanShift [24], GB [8], GBH [12], streamGBH [32] and TSP [4]. Since CSS and streamCSS adopt a random initialization, we report the average results of 20 initializations. The performances are evaluated on four video datasets, i.e., BuffaloXiph [5], SegTrack v2 [18], BVDS [29, 11] and CamVid [3], which have groundtruth labels drawn by human annotators.

In the original implementation of above seven methods collected in the LIBSVX benchmark [31], the clustering of supervoxels does not consider the connectivity of voxels. Therefore, the voxels that have the same supervoxel label (corresponding to a cluster) can have many disjoint components. In video applications such as detecting spatiotemporal closure for foreground object segmentation [15], the characteristic of multiple disjoint components in one label makes supervoxels' performance very bad. In our evaluation, for fairness we extract all the connected components in each supervoxel and relabel them. Therefore, more supervoxels are produced and accordingly the supervoxels's range in Figure 4 are adjusted and different from the ones in the LIBSVX benchmark.

Adherence to object boundaries. As perceptually

[2] http://cg.cs.tsinghua.edu.cn/people/~Yongjin/Yongjin.htm

meaningful atomic regions in videos, supervoxels should well preserve the object boundaries of ground-truth segmentation. 3D under-segmentation error (UE3D), 3D segmentation accuracy (SA3D) and boundary recall distance (BRD) are standard metrics in this aspect [4, 16, 31]. UE3D and SA3D are complementary to each other and both measure the tightness of supervoxels that overlap with ground-truth segmentation. BRD measures how well the groundtruth boundaries are successfully retrieved by the supervoxel boundaries. As shown in Figures 4(a)-(c), CSS, streamCSS and TSP have the highest SA3D, and the smallest UE3D and BRD, demonstrating their ability to adhere to object boundaries.

Explained variation (EV). EV is a standard metric that measures the color variations in supervoxels [22, 31]. A large EV means the color in each supervoxel is close to homogeneity. As shown in Figure 4(d), CSS, streamCSS and TSP have the largest EV.

Computational cost. We record runtime and peak memory of all nine methods. All methods are implemented in C or C++ except NCut (Matlab running with 8 threads) and TSP (Matlab with MEX). As shown in Figure 4(e), GB, CSS and MeanShift are three fastest methods. In particular, CSS is $5\times$ to $10\times$ faster than TSP. As shown in Figures 4(f)-(g), streamCSS and CSS are two methods that use smallest peak memory. In particular, the peak memory of streamCSS and CSS is $22\times$ smaller than TSP.

Three more metrics – mean size variation (MSV), temporal extent (TEX) and label consistency (LC) – are used in [31]. MSV and TEX measure the size variation and average temporal extent of all supervoxels in a video. Since our work advocates to adapt the size of supervoxels according to video content density, these two metrics are no longer suitable. Instead, we qualitatively compare the *content sensitivity* and propose to use the *compactness* measure below. LC is evaluated using groundtruth optical flow. However, as aforementioned, optical flow is only an optional preprocessing tool to video applications and may introduce extra error into supervoxel evaluation.

Content sensitivity. Figure 1 (last column) shows a typical result of CSS. More qualitative results are illustrated in supplemental material. By clipping supervoxels in image frames, these results clearly show that CSS well captures object boundaries in a video and the supervoxels are content sensitive, i.e., small in content-dense regions and large in content-sparse regions. The content sensitive feature is due to the characteristic that regions of high appearance and motion variance have large volumes in $\mathcal{M}$.

Compactness. In many real-world video applications, the solution relies on minimizing an energy function defined on a spatiotemporal supervoxel graph in a video clip. The shape regularity of supervoxels has a direct influence on the complexity of this spatiotemporal supervoxel graph, and thus, affects the application performance. It was observed that compact supervoxels usually have better segmentation performance than non-compact ones [33]. Note that for any connected region $\Omega \subset \mathbb{R}^3$, the isoperimetric inequality holds:

$$A(\Omega) \geq 3V(\Omega)^{\frac{2}{3}} V(B_1)^{\frac{1}{3}} \tag{19}$$

where B_1 is a unit sphere, $A(\Omega)$ and $V(\Omega)$ are bounding surface area and volume of Ω respectively, and the equality holds when Ω is a sphere. Therefore, for a given supervoxel over-segmentation $\tilde{S} = \{\tilde{s}_1, \tilde{s}_2, \cdots, \tilde{s}_K\}$, the compactness metric C is defined as [33]

$$C(\tilde{S}) = \sum_{\tilde{s}_i \in \tilde{S}} Q(\tilde{s}_i) \frac{|\tilde{s}_i|}{N}, \text{where } Q(\tilde{s}_i) = \frac{6\pi^{\frac{1}{2}} V(\tilde{s}_i)}{A(\tilde{s}_i)^{\frac{3}{2}}}, \tag{20}$$

$|\tilde{s}_i|$ is the number of voxels in $\tilde{s}_i$. The larger the compactness value is, the more regular the shape of supervoxels is. As shown in Figure 4(h), CSS and streamCSS have the largest compactness values.

6. Conclusion

In this paper, we propose a simple yet efficient algorithm which obtains content-sensitive supervoxels by computing RCVT – a uniform tessellation – on a video manifold $\mathcal{M}$. $\mathcal{M}$ is constructed by mapping a video into a combined color and spatiotemporal space $\mathbb{R}^6$ and the volume elements on $\mathcal{M}$ reflect the density of video content. We propose a splitting-merging scheme and use it in the classic Lloyd method such that a high quality RCVT can be computed in very few iterations. Our algorithm is easily extended to a stream version for handling long videos. In addition to its easy implementation, our algorithm is theoretically an $(O(1), O(1))$-approximation. Experimental results show that our method and TSP outperform other six representative methods (NCut, SWA, MeanShift, GB, GBH and streamGBH) in terms of metrics UE3D, SA3D, BRD and EV. Our method is better than TSP in terms of compactness and time and space efficiency.

Acknowledgment

This work was supported by the Natural Science Foundation of China (61725204, 61521002, U1736220, 61661130156), Royal Society-Newton Advanced Fellowship (NA150431) and Beijing National Research Center for Information Science and Technology.

References

[1] N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS '09, pages 10–18, 2009. 2, 4, 7

[2] D. Arthur and S. 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. 2, 4

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision*, ECCV '08, pages 44–57. Springer, 2008. 7

[4] J. Chang, D. Wei, and J. 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. 1, 2, 7, 8

[5] A. Y. Chen and J. J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proceedings of Western New York Image Processing Workshop*, 2010. 7

[6] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. L. Yuille. Efficient multilevel brain tumor segmentation with integrated Bayesian model classification. *IEEE Trans. Med. Imaging*, 27(5):629–640, 2008. 1, 2, 7

[7] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999. 5

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

[9] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004. 1, 7

[10] C. C. Fowlkes, S. J. Belongie, and J. 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

[11] F. Galasso, N. S. Nagaraja, T. J. Cárdenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 3527–3534, 2013. 7

[12] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa. Efficient hierarchical graph-based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 2141–2148, 2010. 1, 2, 7

[13] S. D. Jain and K. Grauman. Supervoxel-consistent foreground propagation in video. In *13th European Conference on Computer Vision*, ECCV '14, pages 656–671, 2014. 1

[14] Y. Ke, R. Sukthankar, and M. Hebert. Spatio-temporal shape and flow correlation for action recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'07, 2007. 1

[15] A. Levinshtein, C. Sminchisescu, and S. J. Dickinson. Optimal image and video closure by superpixel grouping. *International Journal of Computer Vision*, 100(1):99–119, 2012. 1, 7

[16] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009. 8

[17] C. Li, L. Lin, W. Zuo, S. Yan, and J. Tang. SOLD: sub-optimal low-rank decomposition for efficient video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 5519–5527, 2015. 1

[18] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, ICCV '13, pages 2192–2199, 2013. 7

[19] Y.-J. Liu, C. Yu, M. Yu, and Y. He. Manifold SLIC: a fast method to compute content-sensitive superpixels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 651–659, 2016. 1, 2, 3

[20] Y.-J. Liu, M. Yu, B.-J. Li, and Y. 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, 3

[21] J. Lu, R. Xu, and J. J. Corso. Human action segmentation with hierarchical supervoxel consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 3762–3771, 2015. 1

[22] A. P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '08, 2008. 8

[23] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *13th European Conference on Computer Vision*, ECCV '14, pages 737–752, 2014. 1

[24] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '07, 2007. 1, 2, 7

[25] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '00, pages 1070–1077, 2000. 2, 7

[26] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006. 2, 7

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

[28] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. 2

[29] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2233–2240. IEEE, 2011. 7

[30] D. Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In *Annual Conference on Neural Information Processing Systems (NIPS) 2016*, pages 604–612, 2016. 2, 4

[31] C. Xu and J. J. Corso. Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 119(3):272–290, 2016. 1, 7, 8

[32] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV '12, pages 626–639, 2012. 1, 2, 7

[33] R. Yi, Y.-J. Liu, and Y.-K. Lai. Evaluation on the compactness of supervoxels. *Submitted for publication*, 2018. 8

[34] C.-P. Yu, H. Le, G. Zelinsky, and D. Samaras. Efficient video segmentation using parametric graph partitioning. In *IEEE International Conference on Computer Vision (ICCV 2015)*, pages 3155–3163, 2015. 1