Computers & Graphics 54 (2016) 38–46

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Issue on CAD/Graphics 2015

Video segmentation with L_0 gradient minimization

Xuan Cheng [a], Yuanli Feng [a], Ming Zeng [b], Xinguo Liu [a,*]

[a] State Key Lab of CAD&CG, Zhejiang University, China
[b] Software School, Xiamen University, China

ARTICLE INFO

Article history:
Received 12 April 2015
Received in revised form
9 July 2015
Accepted 12 July 2015
Available online 21 July 2015

Keywords:
Video segmentation
L_0 gradient minimization
Gradient sparsity
Fused coordinate descent

ABSTRACT

Video segmentation is an important preprocessing step for many computer vision and graphics tasks. Its main goal is to group the voxels in the video volume with similar appearance and motion into spatio-temporally consistent supervoxels. In this paper, we formulate video segmentation as an L_0 gradient minimization problem, so that the spatio-temporal coherence can be effectively enforced through a gradient sparsity pursuit way. In our method, the appearance and motion descriptor space is first built for over-segmented image patches of each video frame. Then the L_0 gradient minimization is performed in the descriptor space, for both spatial and temporal dimensions. To solve the non-convex L_0 norm minimization problem, we extend the fused coordinate descent algorithm from 2D image grids to 3D video volume. We conduct quantitative evaluation of our method in a public video segmentation benchmark LIBSVX. The experimental results demonstrate our superior performance to state-of-the-arts in segmentation accuracy and undersegmentation error, and comparable performance in boundary recall and explained variation.

1. Introduction

As image segmentation for image analysis, video segmentation occupies an important place in the early stage of video analysis. Image segmentation aims to group perceptually similar pixels into *superpixels* [1], and video segmentation generalizes this concept to the grouping of voxels into spatio-temporally consistent *supervoxels* [2]. Such segmentation provides a compact-yet-meaningful representation, which can ease the subsequent high-level tasks such as 3D reconstruction [3], video stylization/abstraction [4], content-based retrieval [5] and activity recognition [6].

Current video segmentation methods could be roughly categorized into two main classes. The first class of methods takes bottom-up strategies (*e.g.* [7,8]). These methods treat the video volume as a 3D graph, and gradually merge the nodes in the graph according to certain merging strategy. Although there exist many plausible merging methods, how to enforce both spatial and temporal coherence of regions effectively during merging process still needs more explorations. The second class involves the trajectory cue (*e.g.* [9–11]). These methods usually extract a set of sparse trajectories throughout the entire video, and then cluster the trajectories based on color and motion cues. The performance of such methods highly rely on the quality of trajectories, and a post-processing step is needed to convert the sparse trajectories clustering to the dense video segmentation.

In this paper, we focus on the methods of the first category. We propose a novel merging approach to group the voxels in the video volume. Our intuition comes from a piece-wise constant model called L_0 gradient minimization [12], which is firstly proposed to do feature-preserving filtering. The main function of the L_0 gradient minimization model is to achieve *gradient sparsity* in the filtered results, where most gradients tend to be zero while non-zero gradients only exist near the boundaries. The measure of gradient sparsity essentially encodes the segmentation information: the neighboring elements that have zero gradients among them form a group naturally, while the non-zero gradients separate different groups, as shown in Fig. 1. Consequently, the spatio-temporal coherence in video segmentation is enforced through a sparsity pursuit manner, involving the L_0 norm.

However, the L_0 norm is difficult to optimize due to its non-convex nature. Based on the recently proposed fused coordinate descent algorithm [13] for L_0 gradient minimization on images, we extend various elements of the minimization from 2D image grids to 3D video volume. Through setting the weights of the L_0 norm, we can roughly control how many segments are produced, and finally get hierarchical video segmentation results.

In summary, the main contributions of this work include:

- Firstly, we formulate video segmentation as a global L_0 gradient minimization problem, and address the main problem in video segmentation: enforcing the spatio-temporal coherence,

* Corresponding author.
E-mail addresses: chengxuan90@gmail.com (X. Cheng), xgliu@cad.zju.edu.cn (X. Liu).

http://dx.doi.org/10.1016/j.cag.2015.07.012
0097-8493/

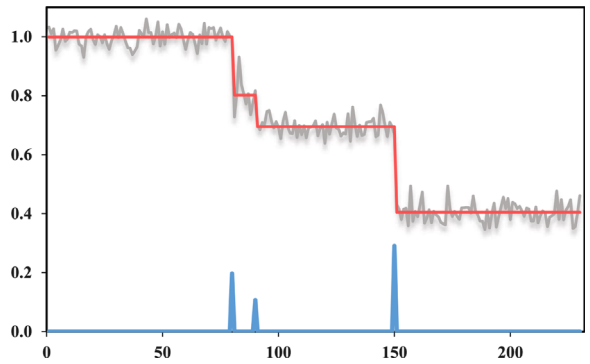

Fig. 1. The gray curve is the noisy input signal, the red curve is the signal filtered by L_0 gradient minimization and the blue curve is the gradient of the filtered signal. Due to the gradient sparsity property, the filtered signal is divided into four segments. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

by gradient sparsity pursuit. To the best of our knowledge, this is the first exploration to introduce sparsity analysis into video segmentation.

- Secondly, we show how to extend the fused coordinate descent algorithm [13] to approximately solve the L_0 gradient minimization in the context of video sequence.
- Thirdly, we conduct experiments on a public video segmentation benchmark LIBSVX [2], and make comparisons with state-of-the-art video segmentation methods.

The rest of the paper is organized as follows: Section 2 introduces some video segmentation methods proposed in recent years, Section 3 reviews the L_0 gradient minimization model and its approximation algorithms, Section 4 presents our video segmentation method, Section 5 shows the experimental results, Section 6 demonstrates two applications of our method, and finally Section 7 concludes this paper.

2. Related work

Bottom-up video segmentation methods usually treat the video as a 3D graph and merge the nodes in the graph gradually. Grundmann et al. [7] propose a hierarchical graph-based method. Their method begins with an over-segmentation of each frame in the video. It then iteratively constructs a region graph over the obtained segmentation, and forms a bottom-up hierarchical tree structure of the regions. In each iteration, the regions are merged with the same technique used in [14]. Paris and Durand [8] extend mean shift from image segmentation to video segmentation. They interpret mean shift as a topological decomposition of the feature space into density modes, and a hierarchical segmentation is obtained by using topological persistence. Fowlkes et al. [15] use the Nyström approximation to apply the Normalized Cuts [16] to the video volume, and make it feasible to perform the eigenvector and eigenvalue analysis in the 3D graph. Sundaram and Keutzer [17] make use of color, texture and motion cues to create a voxel-level affinity matrix for the entire video. Followed by this, the spectral clustering is performed in the affinity matrix to produce supervoxels results. Due to the heavy computation of spectral clustering in the entire video, a parallelized implementation on a GPU cluster is needed in their method. Instead of using GPU cluster, Galasso et al. [18] propose the use of a reduced graph based on superpixels to reduce the computation cost. The reduced graph is reweighted such that the resulting segmentation is equivalent to that of the full graph under certain assumptions. Chang et al. [19] compute the SLIC [20] superpixels of each frame firstly, and then enforce the temporal coherence of the superpixels between frames with a bilateral Gaussian process. Our proposed method falls into this category, but we enhance the spatio-temporal coherence in a sparsity pursuit way.

Trajectory based methods usually leverage pre-computed trajectories to enforce the spatio-temporal coherence in video segmentation. Current optical flow techniques [21,22] could produce accurate and dense fields, and the trajectories are formed by tracking a set of sparse points in the dense optical flow field. Brox and Malik [23] use the reliable tracked trajectories [24] to do spectral clustering, and the pair-wise distances between these trajectories are defined as the maximum difference of their motion over time. Later, Ochs and Brox [25] improve on this method by defining trajectories similarities on high order tuples rather than pairs, since pairwise affinities can only model translations, while 3-affinities can capture rotation and scaling. Lezama et al. [10] propose a new trajectories clustering method, which adds the occlusion reasoning, in the form of depth ordering constraints, into the trajectories grouping cost function. Fragkiadaki et al. [26] find that the discontinuities of embedding density between spatially neighboring trajectories are strong indicators of object boundaries. So the trajectories are merged according to the discontinuity evidence. Palou and Salembier [11] introduce the Binary Partition Tree [27] to trajectories merging, and obtain a hierarchical trajectory tree. All the above trajectories clustering methods need a post-processing step (*e.g.* [28]) to turn the sparse trajectories segmentation into the dense video segmentation.

3. L_0 gradient minimization review

The L_0 gradient minimization model is proposed by Xu et al. [12] for edge-preserving image smoothing:

$$\min_X \sum_i (X_i - I_i)^2 + \lambda \sum_i \sum_{j \in N(i)} |X_i - X_j|_0 \tag{1}$$

where I is the input image, X is the smoothed image, i and j denote the pixel index, I_i denotes the RGB value of i, $N(i)$ denotes the neighbor set of i, $|.|_0$ denotes the L_0 norm and λ denotes the weight of the L_0 norm. The L_0 norm of a vector counts the number of non-zero elements in this vector, which directly measures the sparsity. The intention of L_0 gradient minimization on image is to remove the small scale details while preserve the large scale variations in intensity. The weight λ can control the level of details in the output image.

In this paper, we view the L_0 gradient minimization model from another perspective. Due to the gradient sparsity enhanced by the L_0 norm, many neighboring pixels in the smoothed image will have the same RGB value, so that they are grouped as a region naturally. Meanwhile, the pixels in different regions have different RGB values because of the existence of non-zero gradients. Based on this key observation, we think the L_0 gradient minimization could be a suitable model to segment images and videos.

To solve the non-convex optimization problem Eq. (1), Xu et al. [12] proposed to decompose the original problem into a sequence of computationally tractable L_0–L_2 problems. However, as mentioned in the work of Cheng et al. [13], the sparsity would be corrupted in the iterations of L_0–L_2 optimization. Thus, Cheng et al. [13] proposed a fused coordinate descent algorithm to approximate the optimization. The spirit of their algorithm is to optimize only one pixel at a time, and each neighboring pixels (i,j) are fused together once their RGB values are equal. The merged pixels (i,j) will be optimized together in the next iteration, and in this way the constraint $|X_i - X_j|_0 = 0$ is enforced implicitly.

4. Proposed video segmentation method

The direct way to apply the above L_0 gradient minimization model in the video is to treat the temporal dimension as a third spatial dimension:

$$\min_X \sum_i (X_i - V_i)^2 + \lambda_s \sum_i \sum_{j \in N_s(i)} |X_i - X_j|_0 + \lambda_t \sum_i \sum_{j \in N_t(i)} |X_i - X_j|_0 \quad (2)$$

where V is the input video sequence, X is the filtered result, i and j index the voxels in the video volume, V_i denotes the RGB value of i, $N_s(i)$ denotes the intra-frame spatial neighbors of i, $N_t(i)$ denotes the inter-frame temporal neighbors of i, λ_s is the spatial weight and λ_t is the temporal weight. So the L_0 gradient minimization is performed in the RGB space, for the "*xyt*" dimensions. During optimization, neighboring voxels are gradually merged into supervoxels, and the merged supervoxel uses its mean RGB value as the feature. However, the mean RGB value is insufficient to discriminate homogeneous from textured regions, because the color variation is completely lost in each supervoxel. Moreover the vital motion information is not fully utilized. To overcome these issues, we use a region-based measure instead of a voxel-based measure, and thus propose a size-weighted L_0 gradient minimization model.

4.1. Overview

Fig. 2 presents the whole procedure of our video segmentation method. For an input video sequence, we begin by oversegmenting each frame into a set of image patches. Each patch is connected with its intra-frame neighbors (the green lines in Fig. 2(b)), and its inter-frame neighbors (the blue lines in Fig. 2(b)) along the optical flow direction. To define every patch, we make use of both color and motion cues. The descriptor of a patch consists of the normalized *Lab* histogram and the normalized optical flow histograms, which offers a much richer description for discriminating different regions in the complex video scene. We use the patches obtained from over-segmentation to form a graph as indicated in Fig. 2(c). Then, we perform segmentation with the size-weighted L_0 gradient minimization model, and solve the optimization problem using the extended fused coordinate descent algorithm (Fig. 2(d)). Finally, the video segmentation results are obtained from the merged nodes, with the same color denoting the same spatio-temporal region in Fig. 2(e).

4.2. Size-weighted L_0 gradient minimization

Now we focus on the formulation of video segmentation as an L_0 gradient minimization problem:

$$\min_X \sum_p n_p \cdot Dist(X_p, D_p) + \lambda_s \sum_p \sum_{q \in N(p)} l^s_{p,q} |X_p - X_q|_0 + \lambda_t \sum_p \sum_{q \in N(p)} l^t_{p,q} |X_p - X_q|_0 \quad (3)$$

where p and q index the supervoxels in the video, n_p is the number of voxels in p, D_p is the descriptor of p, X_p here is the filtered result in the descriptor space, $Dist(\cdot,\cdot)$ defines the distance of two vectors in the descriptor space, $N(p)$ denotes the set of neighbors of p in both spatial and temporal dimensions, $l^s_{p,q}$ denotes *spatial boundary length* in voxel level between p and q, while $l^t_{p,q}$ denotes the *temporal boundary length*. At the beginning, each supervoxel p contains only a image patch obtained from over-segmentation. Since the distance measure is important for video segmentation, we explain the distance notion in detail firstly.

Distance term: The descriptor of a supervoxel p comprises a *Lab* histogram and a set of optical flow histograms. Since color is relatively stable over a period of time, a *Lab* histogram is sufficient to model the color distribution of a supervoxel regardless of its temporal span. Unlike color, motion can change over time, and it is only consistent within a frame (discussed in [7,11]). Hence, we use the frame-wise optical flow histograms $\{u_i, u_{i+1}, \ldots, u_{i+k}\}$ to describe the motion distribution of the supervoxel p. The temporal span of p is from frame i to frame $i+k$, and u_i is the optical flow histogram in the frame i. Then the distance $Dist(\cdot,\cdot)$ in the descriptor space is the combination of the *Lab* histogram distance d_c and the optical flow histograms distance d_m. The *Lab* histogram distance is simply defined as the chi-square distance, while the optical flow histograms distance is defined as the average chi-square distance of per-frame optical flow histogram over the same period:

$$d_m = \frac{1}{|T|} \sum_{i \in T} d_{chi-square}(u_i^p, u_i^q) \quad (4)$$

where T is the set of common frames of both supervoxels p, q and $|T|$ is the number of common frames. Finally, the distance term $Dist(\cdot,\cdot)$ has the form

$$Dist(\cdot,\cdot) = (1 - (1 - d_c)(1 - d_m))^2 \quad (5)$$

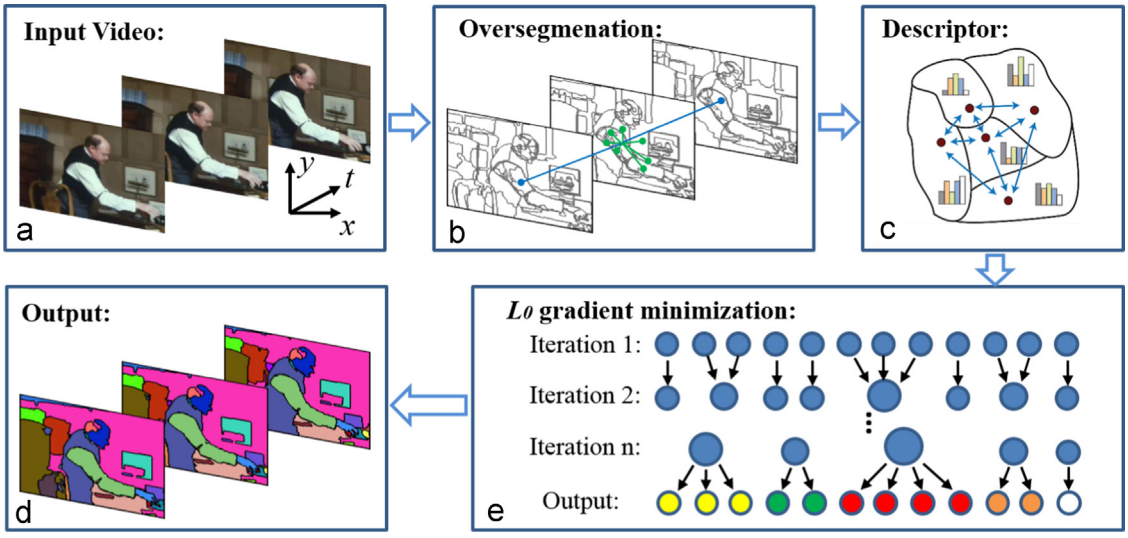

Fig. 2. The overview of our proposed video segmentation method.

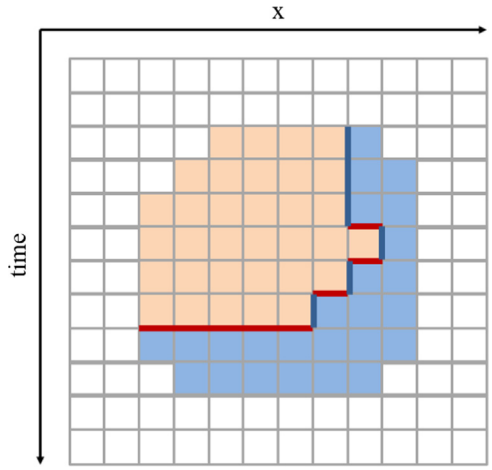

Fig. 3. Horizontal cut of a video sequence. Each cell represents a voxel in the video. Two supervoxels are shown with light red and light blue. The edges with deep blue color indicate the spatial boundary, while the edges with deep red color indicate the temporal boundary. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The distance is close to zero when both color and motion distances are close to zero, and close to one if either one is close to one.

Size weights: Apart from color and motion, the formulation (3) considers size information. Since different supervoxels have a different number of voxels inside, large and small supervoxels should not be treated equally in the optimization. So we use the size information including supervoxel volume and boundary lengths, to weight the distance term and the L_0 gradient terms.

Supervoxel volume: n_p represents the number of voxels in a supervoxel p, and it is used to weight the distance term such that the energy function would make more efforts to fit the large size supervoxels.

Boundary lengths: The non-zero gradients in the voxel level only exist in the boundaries between supervoxels. These boundaries have different lengths, and the longer boundary involves more non-zero gradients. So the boundary lengths should be used as weights for the L_0 gradient terms. As shown in Fig. 3, there are two types of boundary, spatial boundary $l^s_{p,q}$ and temporal boundary $l^t_{p,q}$. The spatial boundary connects two supervoxels along the "*xy*" direction, while the temporal boundary concerns the connection along the "*t*" direction. Each neighboring supervoxels have both types of boundary, which affect the optimization individually.

4.3. Fused coordinate descent based solver

Due to the non-convex nature of the L_0 norm, the optimization of Eq. (3) is a challenging problem. Based on the fused coordinate descent algorithm [13], we propose our solver for Eq. (3) which can find a approximate solution efficiently. The whole algorithm is summarized as Algorithm 1. As in [13], our algorithm repeats coordinate descent step (Lines 2–5) and fusion step (Lines 6–19). The weights λ'_s and λ'_t start from the zero value, and are increased by the step parameters α_s and α_t (Lines 21–22) in the iterations. When either λ_s or λ_t is reached, the iteration terminates (Line 23).

Algorithm 1. Fused coordinate descent based solver.

Input: image patches of number K, parameters $\lambda_s, \lambda_t, \alpha_s, \alpha_t, \epsilon$.
Initialize: compute $n_p, D_p, N(p), G_p, l^s, l^t$ for each image patch p.
$X_p \leftarrow D_p$, $\lambda'_s \leftarrow 0$, $\lambda'_t \leftarrow 0$, $M \leftarrow K$.
1: **Repeat**
2: // coordinate descent step
3: **for** $p = 1 : M$ **do**
4: solve Eq. (6) for X_p.
5: **end for**
6: // fusion step
7: $p \leftarrow 1$.
8: while $p < M$ **do**
9: **for all** $q \in N(p)$ **do**
10: **if** $Dist(X_q, X_p) < \epsilon$ **then**
11: $G_p \leftarrow G_p \cup G_q$.
12: $D_p \leftarrow Merge(D_p, D_q)$.
13: $n_p \leftarrow n_p + n_q$.
14: $M \leftarrow M - 1$.
15: Delete $n_q, D_q, N(q)$ and G_q.
16: **end if**
17: **end for**
18: $p \leftarrow p + 1$.
19: **end while**
20: Compute $N(p), l^s, l^t$ for each p.
21: $\lambda'_s \leftarrow \lambda'_s + \alpha_s$.
22: $\lambda'_t \leftarrow \lambda'_t + \alpha_t$.
23: **until** $\lambda'_s > \lambda_s$ **or** $\lambda'_t > \lambda_t$
Output: the supevoxels set $\{G_p\}$.

Coordinate descent step: In each cycle of the coordinate descent step, only one variable X_p is optimized with all others fixed. So we could break Eq. (3) into a series of subproblem and each has the form

$$\min_{X_p} E(X_p) = n_p \cdot Dist(X_p, D_p) + \lambda'_s \sum_{q \in N(p)} l^s_{p,q} |X_p - X_q|_0 + \lambda'_t \sum_{q \in N(p)} l^t_{p,q} |X_p - X_q|_0 \quad (6)$$

The $\{X_q \mid q \in N(p)\}$ are assumed to be known from previous iteration. To solve Eq. (6), we examine the function values for the neighbor's value $\{X_q \mid q \in N(p)\}$ and the target value D_p, and find the one giving the smallest value of the objection function $E(X_p)$.

Fusion step: After a pass of coordinate descent for all variables $\{X_p\}$, each variable X_p will either be its neighbors value $\{X_q \mid q \in N(p)\}$ or its target value D_p. This means we are likely to find equal neighboring pairs in the fusion step. If an equal neighbor q of p is found (Line 10), we add the voxels in G_q into the set G_p (Line 11), merge both *Lab* histograms and optical flow histograms (Line 12), and update other attributes correspondingly (Lines 13–15). After fusion step, we re-compute the neighbors and the boundary lengths of each p (Line 20).

Discussions: Assuming that the supervoxel adjacency is sparse and the number of iterations (Lines 1–23) is fixed, the time complexity of Algorithm 1 is nearly linear with the number of input image patches K. Since the optimization of Eq. (3) is highly nonconvex, Algorithm 1 is only an approximation. Our goal is not to find a global minimum which is rather difficult, but to make the original optimization problem easier to tackle and maintain the property of gradient sparsity in the proposed descriptor space.

5. Experiments

In this section, we first present the implementation details of important aspects of our method. We then present a set of qualitative experimental results as well as quantitative comparison against the main state-of-the-art counterparts in the LIBSVX benchmark.[1]

[1] http://www.cse.buffalo.edu/jcorso/r/supervoxels/

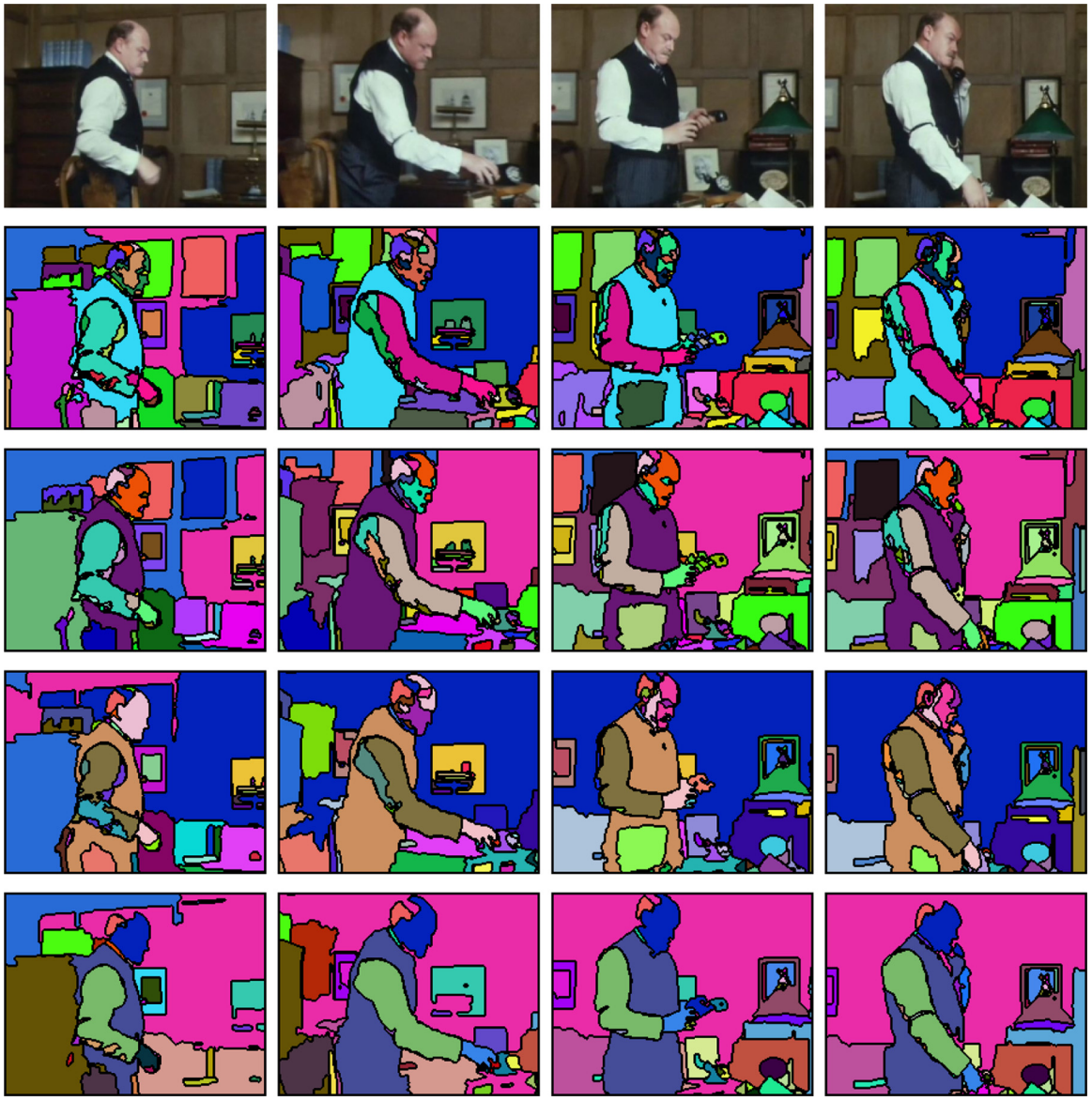

Fig. 4. The video segmentation results in "marple" sequence using our method with different L_0 weights. The first row: input video, the second row: $\lambda_s = 0.04, \lambda_t = 0.02$, the third row: $\lambda_s = 0.04, \lambda_t = 0.04$, the fourth row: $\lambda_s = 0.06, \lambda_t = 0.04$, the fifth row: $\lambda_s = 0.08, \lambda_t = 0.08$.

5.1. Implementation Details

We use the L_0 gradient minimization on images [13] to over-segment each frame and obtain a set of image patches. The *Lab* histogram has 20 bins in each dimension, and the per-frame optical flow histogram has 16 orientation bins (discretized w.r.t. the angle). Algorithm 1 contains five input parameters which are given by the users. The spatial weight λ_s and temporal weight λ_t can roughly control how many spatio-temporal segments are finally produced, and thus yield a hierarchical video segmentation. A large λ_s or λ_t makes the final results have few segments, as shown in Fig. 4. In our experiments, the range of both λ_s and λ_t is $[0, 0.1]$. If λ_t is set as zero, our method degenerates into per-frame image segmentation. The step parameters α_s and α_t are usually set as one-thousandth of λ_s and λ_t, such that λ_s and λ_t can be reached at nearly the same iteration. The threshold parameter ϵ that judges whether two vectors in the descriptor space are equal is set as 0.05 empirically.

5.2. Evaluation on LIBSVX benchmark

The LIBSVX benchmark [2] is specifically designed for the comprehensive evaluation of video segmentation performance. The dataset usually consists of 8 sequences from xiph.org used in [29]. Each sequence has approximately 85 frames and is densely labeled with semantic pixels, leading to a ground-truth segmentation. Four evaluation metrics are utilized to measure the performance. The Segmentation Accuracy (SA) quantifies what fraction of a ground-truth segmentation is correctly classified by the supervoxels. The Undersegmentation Error (UE) measures what fraction of voxels exceeds the volume boundary of the ground-truth segment when mapping the supervoxels onto it. The Boundary Recall (BR) assesses the quality of spatio-temporal boundary detection: for each segment in the ground-truth and supervoxel segmentations, the 3D boundaries are extracted and the recall is calculated using standard formula. The Explained Variation (EV) is a metric that relates to color statistics in the

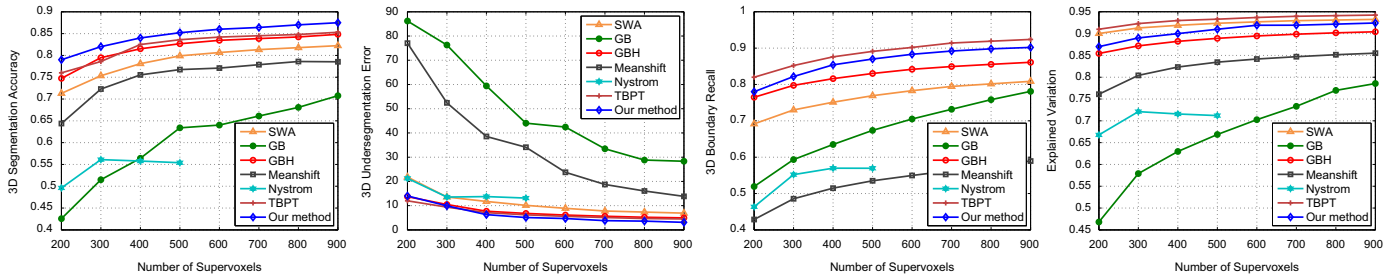

Fig. 5. Qualitative comparison with other video segmentation methods in the LIBSVX: SWA [30], GB [14], GBH [7], Meanshift [8], Nyström [15] and TBPT [11]. The figures from left to right are: Segmentation Accuracy (SA), Undersegmentation Error (UE), Boundary Recall (BR) and Explained Variation (EV).

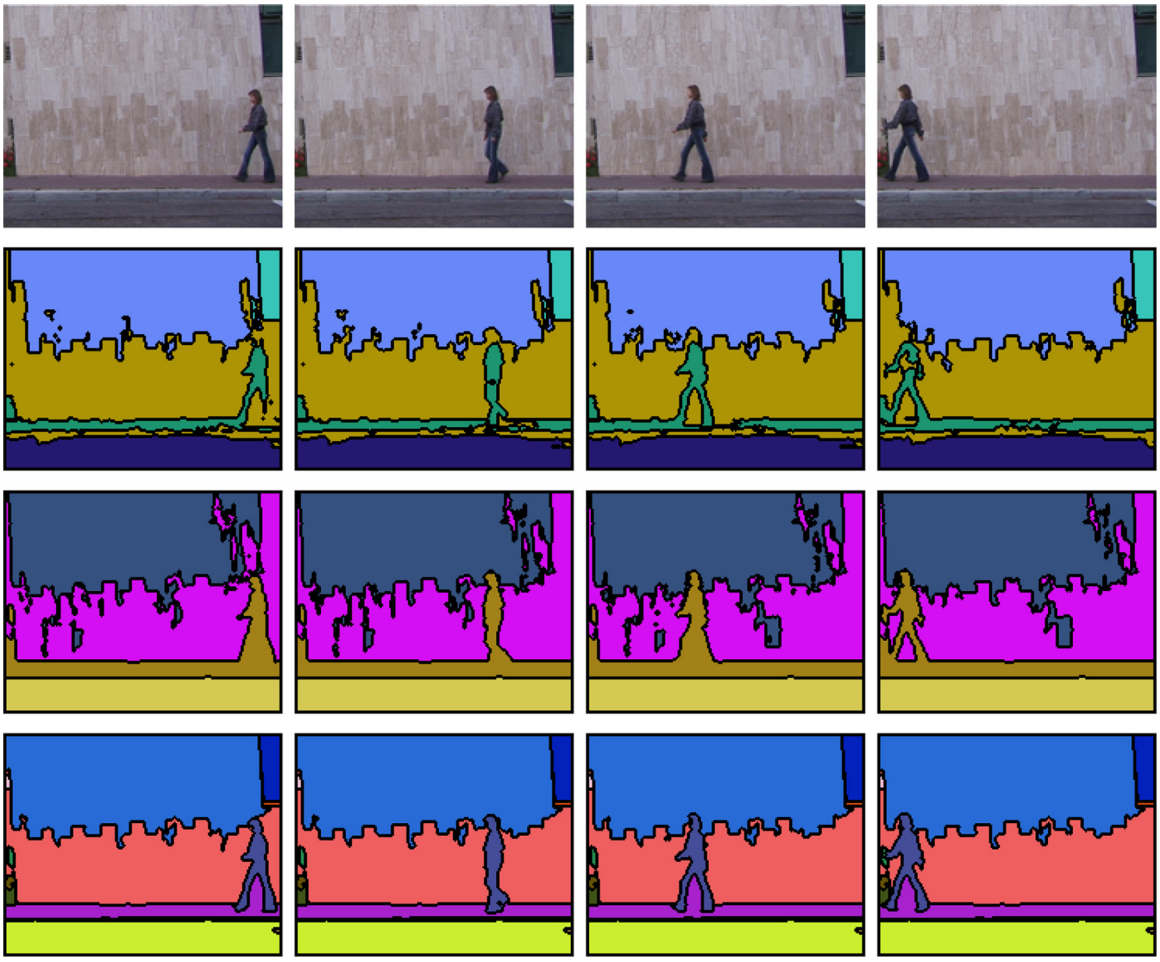

Fig. 6. Segmentation results in the "walking" sequence. From top to bottom: input video, GBH [7], TBPT [11] and our method. The same color denotes the same supervoxels. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

supervoxels. For the detailed definition of these four metrics, we refer readers to [2].

The LIBSVX benchmark requires the evaluated method to produce different number of supervoxels for a testing video, and thus four curves about metrics against number of supervoxels can be generated. So we vary the parameters λ_s and λ_t to get different number of supervoxels and make evaluation in the benchmark. We also compare our method with the methods that have been previously evaluated in the LIBSVX, and the quantitative results are presented in Fig. 5. It could be observed that our method outperforms the other methods in SA and UE, while slightly falls behind TBPT [11] in BR and EV. As the metrics SA and UE directly measure the similarities and differences between the supervoxel segmentations and the ground-truth, we believe that our merging strategy is an effective way to achieve spatio-temporally consistent segmentation. Thanks to the introduction of gradient sparsity into the video segmentation process, our method is able to preserve the principal and meaningful structures in the video volume. On the other hand, as our L_0 gradient minimization model involves the region size information, some trivial or thin structures may not be preserved well. We think this issue leads to our lagging behind TBPT [11] that makes use of pixel-level trajectories, in the BR and EV metrics.

Some qualitative results are presented in Figs. 6–8. In Fig. 6, some trivial regions in the background exist in the segmentation results of GBH [7] and TBPT [11]. Besides, the walking person is merged with the floor in the both methods. Our results have clean background and distinct person regions. In Fig. 7, more trivial

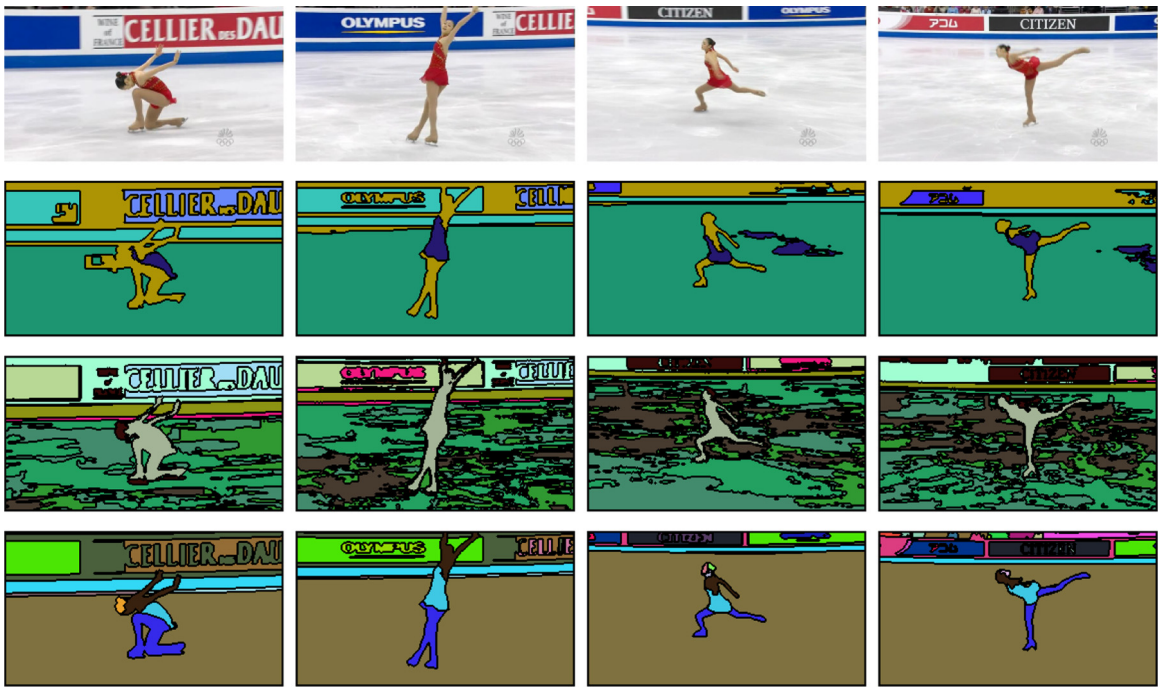

Fig. 7. Segmentation results in the "yunakim" sequence. From top to bottom: input video, GBH [7], TBPT [11] and our method.

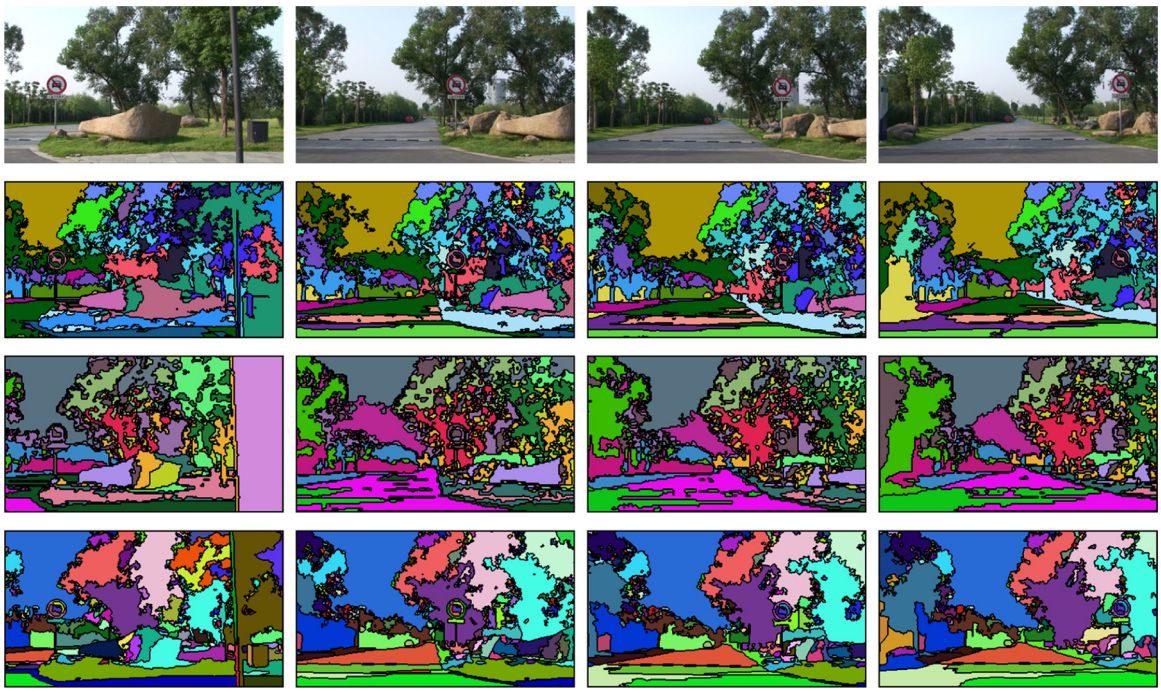

Fig. 8. Segmentation results in the "road" sequence. From top to bottom: input video, GBH [7], TBPT [11] and our method.

regions in the floor appear in the results of GBH and TBPT, while our method avoids this issue. In Fig. 8, our method gets more smooth segmentation for the tree scenes. For the above three examples, we have tuned the parameters slightly of our method and the compared methods, so that all three methods can get nearly the same number of segments for each video to achieve a fair comparison. Complete results and comparisons can be found in the supplementary material.

The time consumptions of our method, GBH and TBPT on the three sequences used in Figs. 6–8, are reported in Table 1. All the three methods are run on an Intel Core i3 CPU@2.93 GHz. The

Table 1
The time consumption (minutes) on the three sequences "walk", "yunakim" and "road". The 'walk" sequence has 80 frames and 180×144 image size. The "yunakim" sequence has 100 frames and 280×156 image size. The "road" sequence has 140 frames and 384×216 image size.

Sequence	GBH [7]	TBPT [11]	Our method
walk	6.0	8.9	1.5
yunakim	12.0	21.3	19.7
road	50.0	12.5	55.0
Average	22.7	14.2	25.4

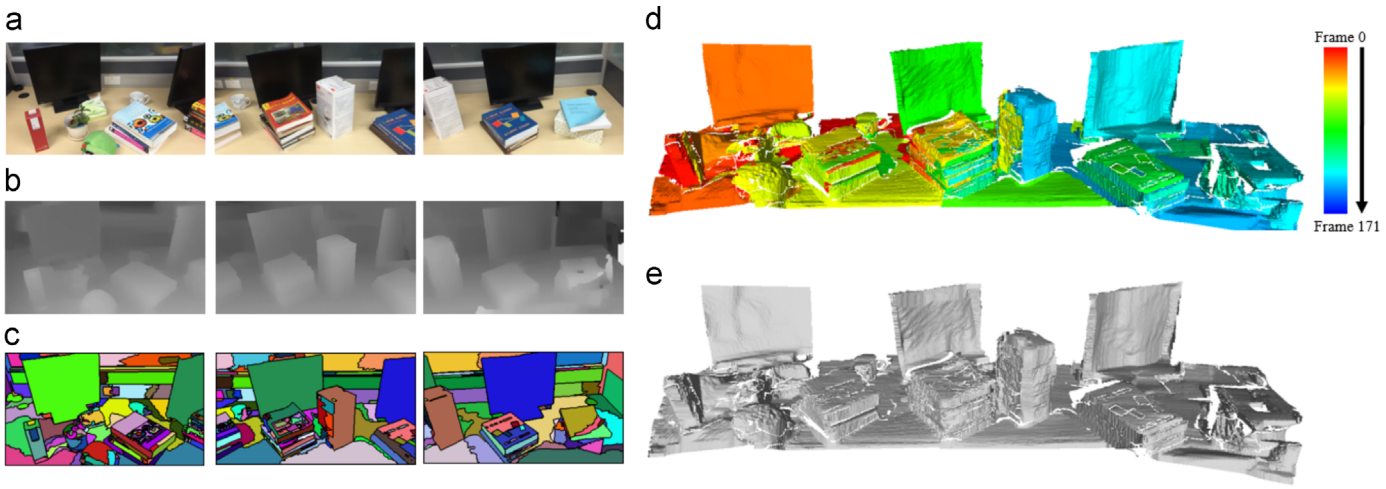

Fig. 9. The 3D geometry reconstruction of a "desktop" scene. (a) The input video sequence. (b) The depth map of each frame is recovered with Zhang et al.'s method [31]. (c) The spatio-temporal segmentation is computed with our method. (d) The selected frame maps are visualized in 3D way. The index of a selected frame is coded with unique color. (e) The reconstructed 3D mesh model. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 10. The first row and second row: the "waterski" sequence and our stylized results. The third row and fourth row: the "coraline" sequence and our stylized results.

running time of each testing sequence does not include the computation time for optical flow. Our method was implemented using C/C++ without intensive program optimization. The implementation of GBH comes from the LIBSVX benchmark, while the executable program of TBPT is provided by the original authors. Except for TBPT, both our method and GBH are without any parallelization. There is still room to speed up our method, such as the parallelization of the fused coordinate descent algorithm.

6. Applications

6.1. 3D geometry reconstruction

Accurate 3D geometry reconstruction of a static scene from a video sequence is always a hot research topic in computer graphics and virtual reality. Many efforts have been made to estimate dense depth map and camera parameters for each frame in the video (*e.g.* [31–33]). The estimated depth maps are then fused to generate a complete 3D geometry model, since each frame only captures a portion of the scene. The direct way of fusion is to register all depth maps together using the estimated camera parameters of each frame. But it will result in significant data redundancy, because neighboring depth maps often contain large overlapping content.

Video segmentation is a feasible method to overcome this issue. The spatio-temporally consistent supervoxels are good representations of the objects in the scene. For each supervoxel, it has different corresponding 2D regions in different frames. We select a frame with maximal region area such that the geometry details can be well preserved in the final reconstructed mesh model. Then we project the pixels of each supervoxel in its selected frame to 3D space using depth information and camera parameters. Each pixel i is connected with its neighbors $\{j\}$ that satisfy depth continuity to form the triangle faces. Similar to [3], the depth continuity criteria is defined as

$$2|z_i - z_j| / (z_i + z_j) < \epsilon_z \tag{7}$$

where z_i is the disparity of pixel i and ϵ_z is a threshold which is set as 0.3. Fig. 9 presents an example of using our video segmentation

method to facilitate the 3D geometry reconstruction of a desktop scene.

6.2. Video stylization

Video stylization and abstraction are popular special effects in many application areas such as live broadcast, video games and other entertainments. Spatio-temporal video segmentation is a key component for many video stylization framework (*e.g.* [3,4,7]). With our video segmentation results, the stylization effects can be simply created by color averaging over the spatio-temporal regions. Fig. 10 shows two examples, where our stylized results are consistent and have few flickering effects. The complete results are presented in the supplementary material.

7. Conclusions

In this work, we have proposed a novel unsupervised video segmentation method using L_0 gradient minimization. We explore the possibility of applying gradient sparsity pursuit in the context of video segmentation. And experimental results in the LIBSVX benchmark validate our idea, in which our method improves segmentation accuracy and reduces undersegmentation error while maintaining boundary recall and explained variation to a competitive level. We hope that this work provides a new thinking for the future research of video segmentation.

As many video segmentation methods, the main limitation of our method is that our method has to process the entire video at once. When dealing with large or medium length video data, exhaust memory resources may be needed. We leave the task of adopting our video segmentation method to a streaming scheme as the future work. Another research direction is to find more applications of our video segmentation method. We plan to apply our method into video object segmentation [34] and co-segmentation [35].

Acknowledgments

We thank the editor and the anonymous reviewers for their constructive comments. This work was partially supported by NSFC (Nos. 61379068 and 61402387), and the Open Project Program of State Key Lab of CAD & CG, Zhejiang University (No. A1419).

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2015.07.012.

References

[1] Ren X, Malik J. Learning a classification model for segmentation. In: Proceedings of CVPR, 2003.
[2] Xu C, Corso JJ. Evaluation of super-voxel methods for early video processing. In: Proceedings of ECCV, 2012.
[3] Jiang H, Zhang G, Wang H, Bao H. Spatio-temporal video segmentation of static scenes and its applications. IEEE Trans Multimedia 2015;17(1):3–15.
[4] Wang J, Xu Y, Shum H, Cohen MF. Video tooning. ACM Trans Graph 2004;23(3):574–83.
[5] Lin T, Yang M, Tsai C, Wang YF. Query-adaptive multiple instance learning for video instance retrieval. IEEE Trans Image Process 2015;24(4):1330–40.
[6] Ma S, Zhang J, Ikizler-Cinbis N, Sclaroff S. Action recognition and localization by hierarchical space–time segments. In: Proceedings of ICCV, 2013.
[7] Grundmann M, Kwatra V, Han M, Essa IA. Efficient hierarchical graph-based video segmentation. In: Proceedings of CVPR, 2010.
[8] Paris S, Durand F. A topological approach to hierarchical segmentation using mean shift. In: Proceedings of CVPR, 2007.
[9] Ochs P, Brox T. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In: Proceedings of ICCV, 2011.
[10] Lezama J, Alahari K, Sivic J, Laptev I. Track to the future: spatio-temporal video segmentation with long-range motion cues. In: Proceedings of CVPR, 2011.
[11] Palou G, Salembier P. Hierarchical video representation with trajectory binary partition tree. In: Proceedings of CVPR, 2013.
[12] Xu L, Lu C, Xu Y, Jia J. Image smoothing via L_0 gradient minimization. ACM Trans Graph 2011;30(6):174.
[13] Cheng X, Zeng M, Liu X. Feature-preserving filtering with L_0 gradient minimization. Comput Graph 2014;38:150–7.
[14] Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int J Comput Vis 2004;59(2):167–81.
[15] Fowlkes C, Belongie S, Chung FRK, Malik J. Spectral grouping using the Nyström method. IEEE Trans Pattern Anal Mach Intell 2004;26(2):214–25.
[16] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 2000;22(8):888–905.
[17] Sundaram N, Keutzer K. Long term video segmentation through pixel level spectral clustering on gpus. In: Proceedings of ICCV Workshops, 2011.
[18] Galasso F, Keuper M, Brox T, Schiele B. Spectral graph reduction for efficient image and streaming video segmentation. In: Proceedings of CVPR, 2014.
[19] Chang J, Wei D, Fisher III J. A video representation using temporal superpixels. In: Proceedings of CVPR, 2013.
[20] Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Süsstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell 2012;34(11):2274–82.
[21] Brox T, Malik J. Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Trans Pattern Anal Mach Intell 2011;33(3):500–13.
[22] Liu C. Beyond pixels: exploring new representations and applications for motion analysis [Doctoral thesis].
[23] Brox T, Malik J. Object segmentation by long term analysis of point trajectories. In: Proceedings of ECCV, 2010.
[24] Sundaram N, Brox T, Keutzer K. Dense point trajectories by gpu-accelerated large displacement optical flow. In: Proceedings of ECCV, 2010.
[25] Ochs P, Brox T. Higher order motion models and spectral clustering. In: Proceedings of CVPR, 2012.
[26] Fragkiadaki K, Zhang G, Shi J. Video segmentation by tracing discontinuities in a trajectory embedding. In: Proceedings of CVPR, 2012.
[27] Salembier P, Garrido L. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. IEEE Trans Image Process 2000;9(4):561–76.
[28] Ochs P, Brox T. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In: Proceedings of ICCV, 2011.
[29] Chen AYC, Corso JJ. Learning a classification model for segmentation. In: Proceedings of Image Processing Workshop (WNYIPW), 2010.
[30] Corso JJ, Sharon E, Dube S, El-Saden S, Sinha U, Yuille AL. Efficient multilevel brain tumor segmentation with integrated Bayesian model classification. IEEE Trans Med. Imaging 2008;27(5):629–40.
[31] Zhang G, Jia J, Wong T, Bao H. Consistent depth maps recovery from a video sequence. IEEE Trans Pattern Anal Mach Intell 2009;31(6):974–88.
[32] Newcombe RA, Davison AJ. Live dense reconstruction with a single moving camera. In: Proceedings of CVPR, 2010.
[33] Zhang G, Jia J, Hua W, Bao H. Robust bilayer segmentation and motion/depth estimation with a handheld camera. IEEE Trans Pattern Anal Mach Intell 2011;33(3):603–17.
[34] Jain SD, Grauman K. Supervoxel-consistent foreground propagation in video. In: Proceedings of ECCV, 2014.
[35] Wang C, Guo Y, Zhu J, Wang L, Wang W. Video object co-segmentation via subspace clustering and quadratic pseudo-boolean optimization in an MRF framework. IEEE Trans Multimedia 2014;16(4):903–16.