Video segmentation with \( L_0 \) gradient minimization

Xuan Cheng\(^a\), Yuanli Feng\(^a\), Ming Zeng\(^b\), Xinguo Liu\(^a,\ast\)

\(^a\) State Key Lab of CAD&CG, Zhejiang University, China
\(^b\) Software School, Xiamen University, China

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,
Section 6 demonstrates two applications of our method, and Section 5 shows the experimental results, respectively. Section 3 reviews the structure of the regions. In each iteration, the regions are merged from the video. It then iteratively constructs a region graph over the video. Grundmann et al.[7] propose a hierarchical graph-based method.

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 supervoxel 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 supervoxels 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] supervoxels of each frame 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.

3. $L_0$ gradient minimization review

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

$$\min_X \sum_i |X_i - I_i|^2 + \lambda \sum_{i,j} |X_i - X_j|_0$$

(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 pixel $i$, $N(i)$ denotes the neighbor set of $i$, $|\cdot|_0$ denotes the $L_0$ norm and $\lambda$ 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 $\lambda$ 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:

\[
\min_{X} \sum_{i} (X_i - V_i)^2 + \lambda_s \sum_{i, j \in N_s(i)} |X_i - X_j|_0 + \lambda_t \sum_{i, j \in N_t(i)} |X_i - X_j|_0
\]

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 \( i \), \( \lambda_s \) is the spatial weight and \( \lambda_t \) is the temporal weight. So the \( L_0 \) gradient minimization is performed in the RGB space, for the “x,y,t” 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 over-segmenting 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 \( \text{Lab} \) 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:

\[
\min_{X} \sum_{p} n_p \cdot \text{Dist}(X_p, D_p) + \lambda_s \sum_{p, q \in N_s(p)} f_{p,q} |X_p - X_q|_0 + \lambda_t \sum_{p, q \in N_t(p)} f_{p,q} |X_p - X_q|_0
\]

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. \( \text{Dist}(.,.) \) defines the distance of two vectors in the descriptor space. \( N_s(p) \) denotes the set of neighbors of \( p \) in both spatial and temporal dimensions, \( f_{p,q} \) denotes spatial boundary length in voxel level between \( p \) and \( q \), while \( f_{p,q} \) 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 \( \text{Lab} \) histogram and a set of optical flow histograms. Since color is relatively stable over a period of time, a \( \text{Lab} \) 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 \( \text{Dist}(.,.) \) in the descriptor space is the combination of the \( \text{Lab} \) histogram distance \( d_i \) and the optical flow histograms distance \( d_m \). The \( \text{Lab} \) 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:

\[
d_m = \frac{1}{|T|} \sum_{t \in I} d_{\text{chi-square}}(u^m_t, u^m_t)
\]

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 \( \text{Dist}(.,.) \) has the form

\[
\text{Dist}(.,.) = (1 - (1 - d_i)(1 - d_m))^2
\]
When either summarized as Algorithm 1, our algorithm repeats can of Eq. (3) is a challenging problem. Based on the fused coordinate "xy" direction, 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, small and large 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 \( f_{pq}^s \) and temporal boundary \( f_{pq}^t \). 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 \( \lambda^s \) and \( \lambda^t \) start from the zero value, and are increased by the step parameters \( \alpha_s \) and \( \alpha_t \) (Lines 21–22) in the iterations. When either \( \lambda^s \) or \( \lambda^t \) 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 \( n_p, D_p, N(p), G_p, \Gamma^s, \Gamma^t \) for each image patch \( p \).

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: \( p \leftarrow p + 1 \).
8: while \( p < M \) do
9: for all \( q \in N(p) \) do
10: if \( \text{Dist}(X_p, X_q) < \epsilon \) then
11: \( G_p \leftarrow G_p \cup G_q \).
12: \( D_p \leftarrow \text{Merge}(D_p, D_q) \).
13: \( n_p \leftarrow n_p + n_q \).
14: \( M \leftarrow M - 1 \).
15: Delete \( n_p, D_p, N(q) \) and \( G_q \).
16: end if
17: end for
18: \( p \leftarrow p + 1 \).
19: end while
20: Compute \( N(p), \hat{f}^s, \hat{f}^t \) for each \( p \).
21: \( \lambda^s \leftarrow \lambda^s + \alpha_s \).
22: \( \lambda^t \leftarrow \lambda^t + \alpha_t \).
23: until \( \lambda^s > \lambda^t \) or \( \lambda^t > \lambda^s \).

Output: the supervoxels 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

\[
\min E(X_p) = n_p \cdot \text{Dist}(X_p, D_p) + \lambda^s \sum_{q \in N(p)} \| X_p - X_q \|_0 + \lambda^t \sum_{q \in N(p)} \| X_p - X_q \|_0
\]

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 its 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 \( \Gamma^s \) and \( \Gamma^t \) 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 1–22) 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/
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 $\lambda_s$ and temporal weight $\lambda_t$ can roughly control how many spatio-temporal segments are finally produced, and thus yield a hierarchical video segmentation. A large $\lambda_s$ or $\lambda_t$ makes the final results have few segments, as shown in Fig. 4. In our experiments, the range of both $\lambda_s$ and $\lambda_t$ is $[0, 0.1]$. If $\lambda_t$ is set as zero, our method degenerates into per-frame image segmentation. The step parameters $\alpha_s$ and $\alpha_t$ are usually set as one-thousandth of $\lambda_s$ and $\lambda_t$, such that $\lambda_s$ and $\lambda_t$ can be reached at nearly the same iteration. The threshold parameter $\epsilon$ 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. 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: $\lambda_s = 0.08$, $\lambda_t = 0.08$. 

X. Cheng et al. / Computers & Graphics 54 (2016) 38–46

42
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 $\lambda_s$ and $\lambda_t$ 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
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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>walk</td>
<td>6.0</td>
<td>8.9</td>
<td>1.5</td>
</tr>
<tr>
<td>yunakim</td>
<td>12.0</td>
<td>21.3</td>
<td>19.7</td>
</tr>
<tr>
<td>road</td>
<td>50.0</td>
<td>12.5</td>
<td>55.0</td>
</tr>
<tr>
<td>Average</td>
<td>22.7</td>
<td>14.2</td>
<td>25.4</td>
</tr>
</tbody>
</table>
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

\[
2|z_i - z_j|/(\sqrt{z_i^2 + z_j^2}) < \epsilon_z
\]

where \( z_i \) is the disparity of pixel \( i \) and \( \epsilon_z \) 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 under-segmentation 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

[22] Liu C. Beyond pixels: exploring new representations and applications for motion analysis [Doctoral thesis].
[31] Liu C. Beyond pixels: exploring new representations and applications for motion analysis [Doctoral thesis].