An Efficient Online Hierarchical Supervoxel Segmentation Algorithm for Time-critical Applications

Yiliang Xu[1]
yiliang.xu@kitware.com
Dezhen Song[2]
dzsong@cse.tamu.edu
Anthony Hoogs[1]
anthony.hoogs@kitware.com

[1] Kitware Inc.
28 Corporate Drive
Clifton Park, New York, USA

[2] Department of Computer Science and Engineering
Texas A&M University
College Station, Texas, USA

Video segmentation has been an active research topic for the last decade. It is often used as a pre-processing procedure for subsequent vision algorithms. Despite its significant practical relevance, research on video segmentation does not catch up with its counterpart of image segmentation, due to multiple challenges including higher dimensional (3D) segmentation, temporal consistency, scalability and efficiency, and many more. Most existing algorithms require pre-loading all or part of the video and batch processing the frames, which introduces temporal latency and significantly increases memory and computational cost. Other algorithms rely on human specification for segmentation granularity control.

In this paper, we propose an efficient online hierarchical supervoxel segmentation algorithm for time-critical applications. Here by online, we mean the algorithm computes the supervoxel segmentation of the video stream up to the latest frame once it arrives. Therefore the algorithm requires no streaming buffer but the incoming frame and thus runs in the truly online manner. It also automatically segments the video with hierarchical granularity. The main contributions of the work include

1. an efficient, yet effective probabilistic segment label propagation across consecutive frames,
2. a new method for label initialization for the incoming frame, and
3. a temporally consistent hierarchical label merging scheme.

Figure 1 illustrates the processing flow of our algorithm. The algorithm starts with the over-segmentation and the corresponding hierarchical segmentations of the first frame using the hierarchical graph-based segmentation [3]. Then it propagates the over-segmentation labels onto the second frame based on both motion (dense optical flow) and appearance cues to form the "seed" segments and the corresponding new graph for the second frame. The seed segments grow in the second frame and new segments (if any) are naturally generated using the graph-based merging to complete the over-segmentation for the second frame. Finally, higher-level segmentations of the second frame are generated with a self-supervision merging scheme based on the segmentation at the same level in the previous frame. These steps are repeated when the new frame is coming to form the up-to-date video stream segmentation.

We test our algorithm on a public benchmark dataset [5], and use a wide range of performance metrics to thoroughly compare it with multiple state-of-the-art algorithms, namely, Segmentation by Weighted Aggregation (SWA) [1, 4], Graph-Based Hierarchical segmentation (GBH) [3], and Streaming Graph-Based Hierarchical segmentation (StreamGBH) [6]. In particular, SWA and GBH are offline algorithm which load the video at once. According to both [2] and [5], GBH is one of the top-performing algorithms. StreamGBH loads a buffer of K frames at a time. Here we test and compare two of its variations with $K = 10$ and $K = 1$, respectively.

Figure 2(a) shows the 3D boundary PR of all algorithms[1]. SWA appears to have the best PR tradeoff. Our algorithm is comparable to GBH and StreamGBH with $K = 10$, and outperforms StreamGBH with $K = 1$. Figure 2(b) shows the 3D volume precision-recall of all algorithms. Our algorithm is comparable to GBH and SWA and outperforms the two StreamGBH variations.

Table 1 shows that our algorithm is significantly faster than all the other algorithms including the StreamGBH with $K = 1$. This is because our graph-based segmentation is carried out only on individual 2D frames, while that in StreamGBH is carried out on a $(K+1)$-frame 3D volume. Even with $K = 1$, the number of edges that need to be cut (for each frame) in StreamGBH is at least a couple of times of that in our algorithm. Our algorithm is also memory-efficient. Offline algorithms or streaming algorithms requires the memory size proportional to the size of the 3D volume buffer, while our algorithm only requires memory size proportional to the 2D frame size.

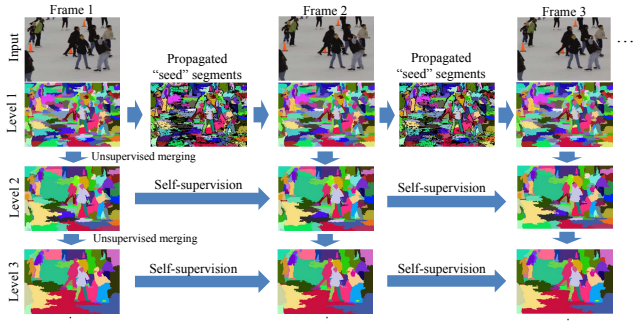
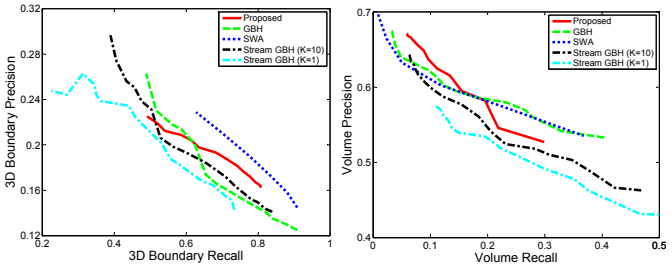


Figure 1: An illustration of the processing flow of the proposed algorithm. Each color corresponds to a supervoxel.



(a) 3D boundary (b) 3D volume

Figure 2: Comparison of Precision-Recall.

		Proposed	StreamGBH		GBH	SWA[2]
			$K = 1$	$K = 10$		
Time (sec.)	per frame[3]	0.72	4.27	8.23	12.96	5.88
	per segmentation	0.03	0.20	0.39	0.62	0.98
Memory (GB)		0.1	0.1	0.5	3.7	8.2

Table 1: Comparison on computation time and memory requirement.

[1] Jason J Corso, Eitan Sharon, Shishir Dube, Suzie El-Saden, Usha Sinha, and Alan Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *IEEE Transactions on Medical Imaging (T-MI)*, 27(5):629–640, 2008.

[2] Fabio Galasso, Naveen Shankar Nagaraja, Tatiana Jiménez Cárdenas, Thomas Brox, and Bernt Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013.

[3] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, pages 2141–2148. IEEE, 2010.

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

[5] Chenliang Xu and Jason J Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012.

[6] Chenliang Xu, Caiming Xiong, and Jason J Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[1] These are not typical PR curves. They are not generated by sweeping a threshold for recognition decision. Instead they are precision and recall pairs by different granularity configurations. The convex shape of the curve does not means the algorithm is worse than random guess.

[2] SWA is set to produce only 6 layers of segmentation to save time.

[3] For each input frame, we produce a number of layers of segmentations.