Streaming Hierarchical Video Segmentation

Chenliang Xu[*], Caiming Xiong[*], and Jason J. Corso

Computer Science and Engineering, SUNY at Buffalo
{chenlian,cxiong,jcorso}@buffalo.edu

Abstract. The use of video segmentation as an early processing step in video analysis lags behind the use of image segmentation for image analysis, despite many available video segmentation methods. A major reason for this lag is simply that videos are an order of magnitude bigger than images; yet most methods require all voxels in the video to be loaded into memory, which is clearly prohibitive for even medium length videos. We address this limitation by proposing an approximation framework for streaming hierarchical video segmentation motivated by *data stream* algorithms: each video frame is processed only once and does not change the segmentation of previous frames. We implement the graph-based hierarchical segmentation method within our streaming framework; our method is the first streaming hierarchical video segmentation method proposed. We perform thorough experimental analysis on a benchmark video data set and longer videos. Our results indicate the graph-based streaming hierarchical method outperforms other streaming video segmentation methods and performs nearly as well as the full-video hierarchical graph-based method.

1 Introduction

The video analysis literature, despite a rich history of segmentation [1], has been dominated by feature-based methods, e.g., [2], over the last decade. More recently, a trend to incorporate an initial oversegmentation into supervoxels followed by various task-specific processes, such as hierarchical grouping [3] and long-range tracking [4, 5], and superpixel flow [6], has surfaced. Many driving applications in video, such as animation [7, 8], require a dense segmentation that feature-based methods cannot naturally provide.

Good progress has been reported on video segmentation in recent years with methods ranging from mean shift [9], region tracking [4], interactive matting [8] and graph-based processing [3]. To evaluate this progress and these methods' comparative suitability for early video oversegmentation into supervoxels, Xu and Corso [10] study five methods, with varying characteristics (segmentation by weighted aggregation (SWA) [11, 12], graph-based (GB) [13], hierarchical graph-based (GBH) [3], mean shift [9], and Nyström normalized cuts [14]). The methods have been evaluated for desirable video segment properties, such as

[*] Equal contribution.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 626–639, 2012.

spatiotemporal uniformity and coherence, explained variation, and spatiotemporal boundary extraction, and compared on a human-labeled video benchmark. The study conclusively reports that the two hierarchical segmentation methods (SWA and GBH) generate space-time segments with the most desirable properties, paving the way for more principled use of video segmentation going forward.

However, there are two extant limitations in the state of practice in video segmentation, and indeed, in those that performed best in [10]. First, most methods build a representation on the entire video and process it at once (e.g. a large graph for whole video). Clearly, they exhaust memory resources on all typical and even some quite large machines (e.g., consider these loading a 320 × 240 video: 3 seconds of video requires 21MB of memory; 2 minutes: 840 MB; 30 minutes: 12.6 GB; 2h: 50GB, and so on). Thus, most existing video segmentation methods are unable to handle even medium-length videos. Those methods that do process the video online, such as streaming mean shift [15] perform comparatively worse on the recent evaluation [10]. Note, [3] presents a hierarchical graph-based algorithm that conclusively performs best in [10]; however, the clip-based version of their method is restricted to the voxel-layer and is not hierarchical, reducing it to graph-based segmentation [13], which performs comparatively worse on the benchmark. We also note that the flow-based work of [6] can also handle videos of arbitrary length, but outputs non-hierarchical segmentations. It is not included in the evaluation; we expect it to perform similarly to those other non-hierarchical methods in [10].

Second, in some applications, such as surveillance and interaction [16], it is impractical to expect the whole video to be available. In these cases, the video is continuously streaming in real time. These limitations seem to have stunted the use of video segmentation; e.g., contrast the relative lack of supervoxels in the video analysis literature with the dominance of superpixels in the image analysis literature [13, 17–22]. Indeed, due to the lack of long-range, efficient supervoxel methods, some researchers resort directly to frame-by-frame superpixel methods, such as [23] who seek a video-based foreground object extraction.

To overcome these limitations, we present a principled approximation to hierarchical video segmentation. Our key contributions are twofold: (1) we propose a streaming hierarchical video segmentation framework that leverages ideas from *data streams* [24] and enforces a Markovian assumption on the video stream to approximate full video segmentation and (2) we instantiate the streaming graph-based hierarchical video segmentation within this approximation framework. Our method is the first streaming hierarchical video segmentation method in the literature, to the best of our knowledge. Fig. 1 illustrates the differences between frame-by-frame, streaming, and full volume processing. Although frame-by-frame processing is efficient and can achieve high-performance in spatial respects, its temporal stability is limited.

We have implemented and evaluated our streaming hierarchical graph-based algorithm both on the existing benchmark [10] and on longer length videos. Our experiment results demonstrate the streaming approximation can perform almost as well as the full-video hierarchical graph-based method and asymptotically approaches the full-video method as the size of the streaming-window increases.

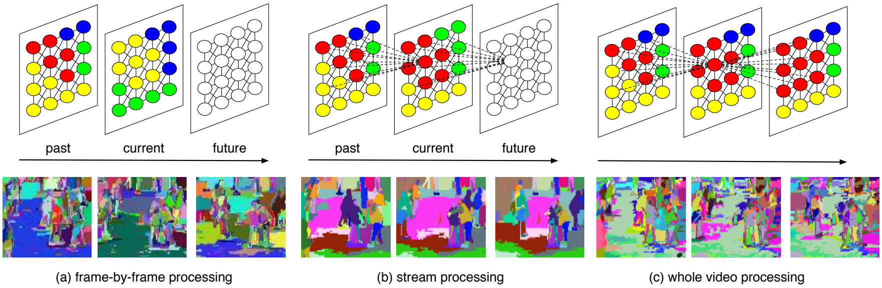

Fig. 1. Three different processing paradigms for video segmentation. (a) Frame-by-frame processing such that each frame is independently segmented, but no temporal information is used. Even though it is fast, the results and temporal coherence are poor. (b) Stream processing segments the current frame only based on a few previously processed frames. It is forward-only online processing, and the results are good and efficient in terms of time and space complexity. (c) 3D volume processing that represents a model for the whole video. It is bidirectional multi-pass processing. The results are best, but the complexity is too high to process long and streaming videos.

However, the streaming approximation maintains better performance of the hierarchical segmentation than full-video, non-hierarchical methods, and the proposed method outperforms other state of the art streaming methods on longer videos.

The remainder of the paper is organized as follows: Sec. 2 presents our streaming hierarchical video segmentation approximation framework, Sec. 3 introduces the graph-based streaming hierarchical method, Sec. 4 discusses our experimental analysis, Sec. 5 concludes the paper.

2 Streaming Hierarchical Video Segmentation – An Approximation for Hierarchical Video Segmentation

Problem Statement. For a given video $\mathcal{V}$, consider an objective function or criterion $E(\cdot|\cdot)$ to obtain the hierarchical segmentation result $\mathcal{S}$ by minimizing:

$$\mathcal{S}^* = \operatorname*{argmin}_{\mathcal{S}} E(\mathcal{S}|\mathcal{V}) \quad . \tag{1}$$

Concretely, let Λ^2 denote the 2D pixel lattice. We think of a video as a function on space-time lattice $\Gamma \doteq \Lambda^2 \times \mathbb{Z}$ mapping to color space $\mathbb{R}^3$, s.t. $\mathcal{V}\colon \Gamma \rightarrow \mathbb{R}^3$. The hierarchical segmentation results in h layers of individual segmentations $\mathcal{S} \doteq \{S^1, S^2, \ldots, S^h\}$ where each layer S^i is a set of segments $\{s_1, s_2, \ldots\}$ such that $s_j \subset \Gamma$, $\cup_j s_j = \Gamma$, and $s_i \cap s_j = \varnothing$ for all pairs of segments. Each layer in the hierarchy gives a full segmentation and the hierarchy itself defines a forest of trees (segments only have one parent). There are several methods, e.g., [3, 15], for pursuing this hierarchical segmentation. However, we are not aware of any method that has an explicit objective function to optimize; they are all based on some criteria to pursue hierarchical segmentation results. We consider them as greedy/gradient descent-like methods to approximate the global or local minimal of an implicit objective function $E(\mathcal{S}|\mathcal{V})$.

Next, we introduce our streaming hierarchical video segmentation framework that approximates a solution for (1) by leveraging ideas from data streams [24].

2.1 Streaming Hierarchical Video Segmentation – An Approximation Framework for $E(\mathcal{S}|\mathcal{V})$

Consider a stream pointer t that indexes frames in a video $\mathcal{V}$ of arbitrary length. By definition, our streaming algorithm may touch each frame of video at most a constant number of times and it may not alter the previously computed results from frames $\hat{t} < t$. Without loss of generality, we can conceptualize this streaming video as a set of m non-overlapping subsequences, $\mathcal{V} = \{V_1, V_2, \cdots, V_m\}$ with k_i frames for subsequence V_i. Thus, hierarchical segmentation result $\mathcal{S}$ could approximately be decomposed into $\{S_1, S_2, \cdots, S_m\}$ as video $\mathcal{V}$ in Fig. 2, where S_i is hierarchical segmentation of subsequence V_i.

Then, to enforce the data stream properties, we make a Markov assumption. Assume hierarchical segmentation S_i of subsequence V_i is only conditioned on the subsequence V_{i-1} and its segmentation result S_{i-1}. Therefore we can obtain a directed Bayesian-like network approximating the hierarchical video segmentation model, see Fig. 2, as:

$$\begin{aligned} E(\mathcal{S}|\mathcal{V}) &= E^1(S_1|V_1) + E^1(S_2|V_2, S_1, V_1) + \cdots + E^1(S_m|V_m, S_{m-1}, V_{m-1}) \\ &= E^1(S_1|V_1) + \sum_{i=2}^{m} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \ , \end{aligned} \tag{2}$$

where, for convenience, we denote $E(\cdot|\cdot), E^1(\cdot|\cdot), E^2(\cdot|\cdot)$ as different (hierarchical) segmentation models for whole video, video subsequence and each layer of video subsequence, respectively. So,

$$\mathcal{S} = \{S_1, \cdots, S_m\} = \operatorname*{argmin}_{S_1, S_2, \cdots, S_m} \left[E^1(S_1|V_1) + \sum_{i=2}^{m} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \right] \ , \tag{3}$$

where $E^1(S_i|V_i, S_{i-1}, V_{i-1})$ is hierarchical segmentation model for each subsequence and $E^1(S_1|V_1)$ is a special case (no information from the previous subsequence).

Given an explicit objective function $E^1(\cdot|\cdot)$, Eq. (3) could be solved using dynamic programming, but because most hierarchical video segmentation methods do not provide an explicit energy function and the configuration space of S_i is too large, we can not use dynamic programming. For our streaming scenario—to obtain $\mathcal{S}$ efficiently—we again make the assumption that the hierarchical segmentation result S_i for the current subsequence V_i never influences segmentation results for previous subsequences, it only influences the next subsequence V_{i+1}. Therefore, Eq. 3 is approximated as:

$$\begin{aligned} \mathcal{S} = \{S_1, \cdots, S_m\} = \Big\{ &\operatorname*{argmin}_{S_1} E^1(S_1|V_1), \cdots, \operatorname*{argmin}_{S_i} E^1(S_i|V_i, S_{i-1}, V_{i-1}), \\ &\cdots, \operatorname*{argmin}_{S_m} E^1(S_m|V_m, S_{m-1}, V_{m-1}), \Big\} \ , \end{aligned} \tag{4}$$

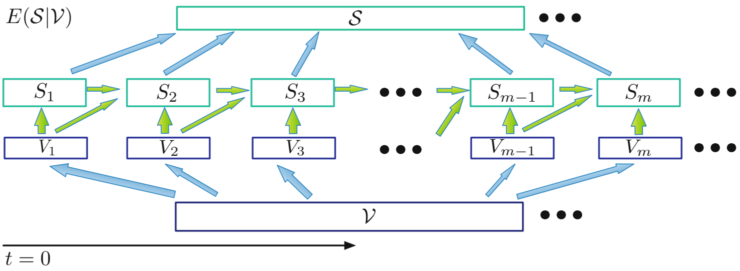

Fig. 2. Framework of streaming hierarchical video segmentation

which can be solved greedily. Start from V_1, obtain S_1, then fix S_1 and obtain S_2 based on S_1, V_1, and V_2. In the same way, obtain $S_3, S_4, \cdots, S_i$ until the whole video is completed.

Therefore, at any time, our approximation framework only needs to store two subsequences into memory, whatever the length of video is, and each subsequence is processed only once. The length k_i of a subsequence V_i is a parameter and can range from 1 to the full length of the video. According to the definition of data streams, we hence name this approximation framework with greedy solution as **streaming hierarchical video segmentation**. And our framework obviously can well handle the two proposed problems of current hierarchical video segmentation mentioned in the introduction: namely, we do not store the whole video at once and in some cases, the whole video may not be available.

2.2 Model for Estimating $S_i|(V_i, S_{i-1}, V_{i-1})$

We can hierarchically segment video of arbitrary length if we know how to solve:

$$S_i = \underset{S_i}{\operatorname{argmin}}\, E^1(S_i|V_i, S_{i-1}, V_{i-1}) \ , \tag{5}$$

where S_i is hierarchical segmentation of subsequence V_i, s.t. $S_i = \{S_i^1, S_i^2, \cdots, S_i^h\}$ with h layers. S_i^j is the segmentation result at the j_{th} layer and the first layer S_i^1 is the video itself, where each voxel is a segment. Continuing with a similar Markov assumption, assume S_i^j only depends on S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j and the subsequence V_{i-1} and V_i, see Fig. 3. Then we can obtain:

$$\begin{aligned} S_i &= \underset{S_i}{\operatorname{argmin}}\, E^1(S_i|V_i, S_{i-1}, V_{i-1}) \\ &= \underset{S_i^2,\cdots,S_i^h}{\operatorname{argmin}} \sum_{j=2}^{h} E^2(S_i^j|V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1}) \ . \end{aligned} \tag{6}$$

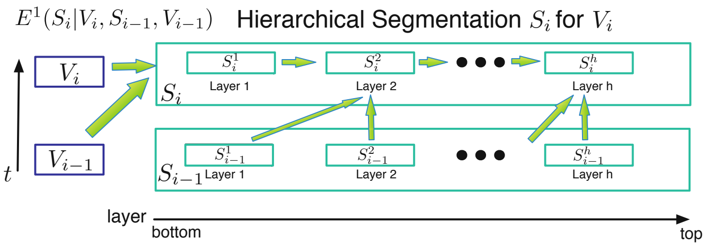

Fig. 3. Sub-framework for hierarchical segmentation for single subsequence V_i

We can hence reformulate this using a greedy approach, as we did in Eq. 4:

$$\begin{aligned} S_i =& \operatorname*{argmin}_{S_i} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \\ =& \Big\{ \operatorname*{argmin}_{S_i^2} E^2(S_i^2|V_i, S_i^1, S_{i-1}^1, S_{i-1}^2, V_{i-1}), \cdots , \\ & \operatorname*{argmin}_{S_i^h} E^2(S_i^h|V_i, S_i^{h-1}, S_{i-1}^{h-1}, S_{i-1}^h, V_{i-1}) \Big\} \ , \end{aligned} \tag{7}$$

where, recall, $E^2(\cdot|\cdot)$ is the conditional segmentation model for a single layer of the subsequence. Since in the first layer, each voxel is a segment, S_i^1 is given. Eq. 7 can be solved by using the greedy solution for Eq. 4. In this way, we transfer complex conditional hierarchical segmentation $S_i|(V_i, S_{i-1}, V_{i-1})$ to a simple layer by layer conditional segmentation $S_i^j|(V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1})$. Thus, we can obtain S_i by pursuing segmentation S_i^j layer by layer.

2.3 Semi-supervised Grouping Method for Estimating $S_i^j|(V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1})$

According to Eq. 7, now we propose a method to estimate S_i^j that:

$$S_i^j = \operatorname*{argmin}_{S_i^j} E^2(S_i^j|V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1}) \ . \tag{8}$$

First, we consider Eq. 8 to pose a semi-supervised grouping problem, given our streaming assumptions (not user input). Given joint segment set $S_{i-1,i}^{j-1} = S_{i-1}^{j-1} \cup S_i^{j-1}$, each small segment is described by a feature-vector on appearance from $V_i \cup V_{i-1}$. S_{i-1}^j is the segmentation result of the j_{th} layer for subsequence V_{i-1}; since it is above the $j-1_{th}$ layer, it is seen as supervised information for segments that belong to S_{i-1}^{j-1}, and these segments are hence supervised

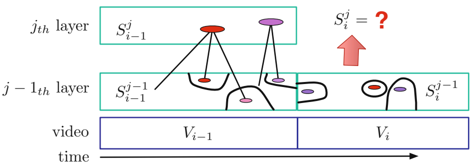

Fig. 4. Illustration of posing Eq. 8 as a semi-supervised problem

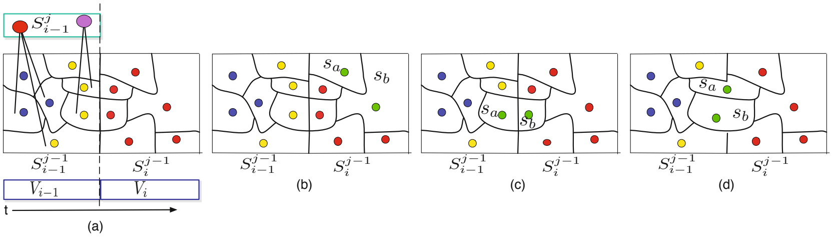

Fig. 5. Example of the three cases in the additional merging criteria: (a), the initial status of semi-supervised grouping; (b-d) three different cases, when s_a and s_b needed to be merged: (b) $s_a \subset S_i^{j-1}$ and $s_b \subset S_i^{j-1}$; (c) $s_a \cap S_{i-1}^{j-1} \neq \emptyset$ and $s_b \subset S_i^{j-1}$; (d) $s_a \cap S_{i-1}^{j-1} \neq \emptyset$ and $s_b \cap S_{i-1}^{j-1} \neq \emptyset$

segments. The goal is to infer S_i^j: the labels of other unsupervised segments in S_i^{j-1}, with fixed labels for segments in S_{i-1}^{j-1}. Fig. 4 illustrates this idea.

Now we design a general semi-supervised algorithm $H_{semi-G}(S_i^{j-1})$ to infer S_i^j. Given any unsupervised grouping method $H_G(S_i^{j-1})$ that obtains segmentation result by merging small segments into large segments iteratively (see Sec. 3 for a graph-based example), we add one more step in its merging process to arrive at semi-supervised grouping algorithm $H_{semi-G}(S_i^{j-1})$. Assume there are two segments s_a and s_b in the current layer, they need to be merged based on the selected grouping method, this merging should be reevaluated according to three cases, which we call the **additional merging criteria** (see an example of each case in Fig. 5 (b-d)):

1. If s_a and s_b both are unsupervised segments, as in Fig. 5(b), then s_a and s_b can be merged.
2. If s_a is an unsupervised segment and s_b contains some supervised segments, as in Fig. 5(c), then s_a and s_b also can be merged, vice versa.
3. If s_a and s_b both contain some supervised segments, as in Fig. 5(d), if they have the same parent, then they are merged, otherwise they are not merged.

Finally, when the merging process stops, we get large merged segments from S_i^{j-1}, they are considered as j_{th} layer segmentation result S_i^j of hierarchical segmentation for subsequence V_i. Then we assign unique indices to these large segments. If the large merged segment contains some supervised segments in S_{i-1}^{j-1}, its index is same as parent node's index in S_{i-1}^j of the supervised segment, or assign it a new index. This ensures consistent indices of same object in continuous subsequences and also proposes new indices for new objects.

There are two advantages of our semi-supervised grouping method. First, from low to high layers, more and more segments will merge with segments from the previous subsequence and the indices of these segments from the current subsequence are the same as the indices of the corresponding segments in the previous subsequence. Thus in a higher layer, the indices of the segments between neighboring subsequences are more consistent than those of a lower layer.

Second, in the third case of the additional merging criteria, we avoid merging s_a and s_b when they contain some supervised segments that connect to different parent nodes in S^j_{i-1}. If we do not handle this case, supervised segments in $s_a \cup s_b$ will be merged and connected to the same parent node thereby contradicting with hierarchical segmentation results S_{i-1} for previously processed subsequence V_{i-1}, and violating the streaming constraints.

3 Graph-Based Streaming Hierarchical Video Segmentation

Felzenszwalb and Huttenlocher [13] propose a graph-based unsupervised grouping method for image segmentation (GB). Their objective is to group pixels that exhibit similar appearance. First represent the image as graph with nodes as pixels and edges connecting nodes according to their spatial relationship. The edge weight is calculated based on the similarity of two nodes. Then the method iteratively merges neighboring regions based on relaxed internal variation $RInt(\cdot)$ of regions, which we will define below. It allows regions to be merged if the weight of the edge connecting them is larger than their relaxed internal variations.

Although Grundmann et al. [3] have already extended the GB method to hierarchical form, we re-derive our version directly from [13] to put it into our streaming approximation framework; [3] do not provide a streaming version of GB. In order to adopt this graph-based grouping method into our streaming hierarchical segmentation framework, we construct a graph over the spatial-temporal video volume with a 26-neighborhood in 3D space-time. This graph is only constructed for the current two subsequences in process, V_i and V_{i-1}; this graph is the first layer of the hierarchy and edge weights are direct color similarity of voxels. Assume we have computed the streaming hierarchical segmentation result up to subsequence V_i using ideas from the previous section, i.e., given the hierarchical segmentation result S_{i-1} of V_{i-1}, we infer the hierarchical segmentation for V_i layer by layer.

Consider the j_{th} layer of V_i as an example. We build a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ based on the $j-1_{th}$ layer of subsequences S_{i-1} and S_i, where each segment in the $j-1_{th}$ layer $\{S^{j-1}_i, S^{j-1}_{i-1}\}$ is a node x_a that is described by a histogram of *Lab* color of all voxels in the segment. If there is one edge between two segments/nodes $x_a \in \mathbf{V}$ and $x_b \in \mathbf{V}$ in the first layer, then there is edge $e_{ab} \in \mathbf{E}$ between segments, and its weight $w(e_{ab})$ is χ^2 distance of the corresponding two histograms. Since $S^j_{i-1} \in S_{i-1}$ is given, we consider S^j_{i-1} to be semi-supervised information for nodes in graph $\mathbf{G}$; thus, all segments/nodes in S^{j-1}_{i-1} are indexed nodes, and their index can not be changed by our semi-supervised grouping method.

According to [13], denote a region R as a subset of connected nodes in graph $\mathbf{G}$. The internal variation of region $RInt(R)$ is the maximum edge weight e_{max} of its Minimum Spanning Tree (MST) plus a relaxation term:

$$RInt(R) = Int(R) + \sigma(R), \text{ with } \quad \sigma(R) = \frac{\tau}{|R|} \tag{9}$$

$$\text{s.t.} \quad Int(R) = \max_{e \in MST(R)} w(e) \ , \tag{10}$$

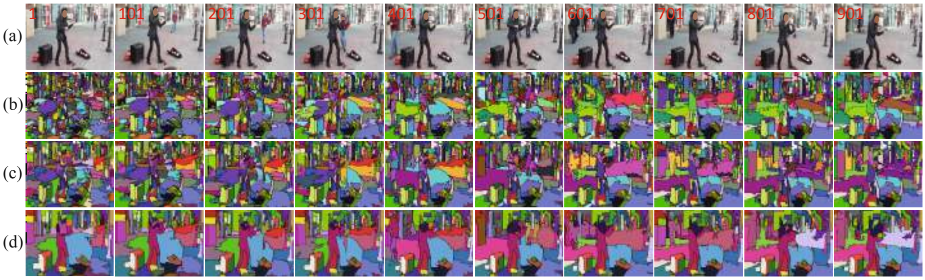

Fig. 6. Example long term video StreamGBH output with $k = 10$. (a) the video with frame number on top-left, (b) the 5th layer, (c) the 10th layer, (d) the 14th layer segmentations. Faces are obscured for privacy concerns in the figure.

where $|R|$ is the number of voxels in region R, and τ is a constant parameter that controls the number of segments to be computed. Our semi-supervised grouping method $H_{semi-G}(\cdot)$ simply combines the merging method in [13] by Eq. 10 and semi-supervised information from S_{i-1}^{j}. First, sort the edges in increasing order in the graph $\mathbf{G}$; and then traverse the sorted edges. At each edge, check whether the edge weight is smaller than the internal variation of both regions incident to the edge: if not, keep traversing; if so, then use the additional merging criteria to decide whether to merge or not. Therefore, based on our proposed semi-supervised $H_{semi-G}(\cdot)$, we finally obtain graph-based streaming hierarchical video segmentation. Fig. 6 shows some example layer-segmentations of the method on a long video (30 seconds) with an equal subsequence length $k = 10$ frames.

4 Experimental Evaluation

Implementation. We have implemented the graph-based streaming hierarchical video segmentation method (**StreamGBH**), which runs on arbitrarily long videos with low and constant memory consumption. Our implementation adaptively handles streaming video and can process bursts of k frames on the stream, where k can be as few as 1 or as many as the local memory on the machine can handle. When $k = \mathit{all\ frames}$, it is a traditional hierarchical graph-based video segmentation method (GBH) [3]. When the hierarchical level $h = 1$, it is a single layer graph-based streaming video segmentation method (**StreamGB**). When $k = \mathit{all\ frames}$ and $h = 1$, it is a graph-based video segmentation method (GB) by extending Felzenszwalb's graph-based image segmentation [13] to the 3D video volume. StreamGBH and all of these variants are available as part of our **LIBSVX** software library, which is downloadable at `http://www.cse.buffalo.edu/~jcorso/r/supervoxels/`.

We analyze the following two aspects of StreamGBH in the context of the benchmark in [10] and provide details on the specific results in Sec. 4.1 and 4.2. We discuss space and time complexity in Sec. 4.3.

Comparison between Different Subsequence Lengths: Vary k. We explore the effectiveness of the streaming variate k in the performance of our

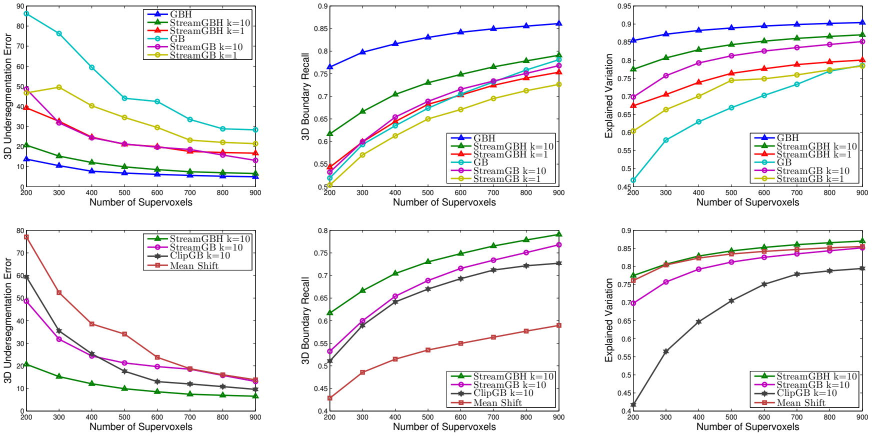

Fig. 7. Quantitative experiments on the benchmark data set. Left: 3D Undersegmentation Error. Middle: 3D Boundary Recall. Right: Explained Variation. Top Row: performance of Streaming GB/GBH with different k against full-video versions. Bottom Row: comparison against streaming methods.

StreamGB and StreamGBH methods. The algorithms are expected to reach their lower bound on performance when $k = 1$, and their upper bound on performance (GB/GBH) when $k = all\ frames$, respectively.

Comparison against State of the Art Streaming Methods. We compare our StreamGB and StreamGBH methods against Grundmann's clip-based graph-based method (CGB) [3] and Paris's streaming mean shift method [15] with fixed $k = 10$. Since neither of their implementations is publicly available, we have implemented Grundmann's CGB method. For Paris's Streaming Mean Shift, we use the source code for whole video processing provided on the author's website `http://people.csail.mit.edu/sparis/#code`) as an upper bound for the streaming mean shift implementation.

4.1 Quantitative Performance: Benchmark Comparisons

Data. We use the recently published benchmark data set (Chen Xiph.org) [25] and video segmentation performance metrics from [10] in the quantitative experiments. Specifically, the video data set is a subset of the well-known `xiph.org` videos that have been supplemented with a 24-class semantic pixel labeling set (the same classes from the MSRC object-segmentation data set [26]). The 8 videos in this set are densely labeled with semantic pixels and have a length of 85 frames each. This data set allows us to evaluate the segmentation methods against human perception.

3D Undersegmentation Error. It measures what fraction of voxels go beyond the volume boundary of the ground-truth segment when mapping the segmentation onto it. Refer to the benchmark paper [10] for the derivation of the measure. Fig. 7 (left column) shows the dependency of 3D Undersegmentation Error on the number of segments. The upper-left of Fig. 7 shows an interesting finding. The curves of StreamGBH are more smooth than StreamGB, and the performance of StreamGBH gradually converges to GBH as the length k of incoming video subsequences increases, which agrees with our approximation framework in Sec. 2.1. The StreamGBH with $k = 10$ has comparative performance with GBH. However the GB method performs worst among all the methods. The reasons are twofold: (1) without a hierarchical structure, the GB method is more dependent on the input parameters for different videos which makes the curve less smooth and (2) without an over-segmentation stage, the GB method is partially dependent on an enforced minimum region size, which is not robust for videos of different lengths. According to reason (2), StreamGB method works more reasonably. The bottom-left of Fig. 7 shows our proposed StreamGBH method achieves the best among all the other streaming methods.

3D Boundary Recall. The 3D boundary is the shape boundary of a 3D object, composed by surfaces. It measures the spatiotemporal boundary detection: for each segment in the ground-truth and segmentations, we extract the within-frame and between-frame boundaries and measure recall using the standard formula. Fig. 7 (middle column) shows the dependency of 3D Boundary Recall on the number of segments. StreamGBH again performs best and StreamGB is also better than the other two in the bottom-middle of Fig. 7.

Explained Variation. Explained Variation is proposed in [20] as a human-independent metric. It considers the segmentation as a compression method of a video. $R^2 = \frac{\sum_i (\mu_i - \mu)^2}{\sum_i (x_i - \mu)^2}$ sums over i voxels, x_i is the actual voxel value, μ is the global voxel mean and μ_i is the mean value of the voxels assigned to the segment that contains x_i. Fig. 7 (right) shows the dependency of Explained Variation on the number of segments. The bottom-right of Fig. 7 again shows that StreamGBH, StreamGB and mean shift methods have comparative performance when the number of segments is larger than 500.

Mean Duration of Segments. When evaluating the performance among different streaming algorithms, we believe the mean duration of segments is a more important metric, as it measures the temporal coherence of a segmentation method more directly. Fig. 8 shows the dependency of mean duration of segments on number of segments. Our proposed StreamGBH and StreamGB both perform better than the other two. ClipGB performs worst and loses most of the temporal information.

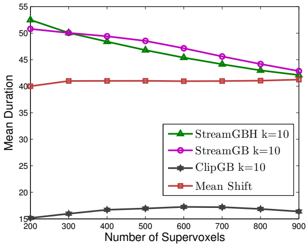

Fig. 8. Mean duration of segments vs. number of segments.

4.2 Qualitative Performance on Long Videos

Here we show some qualitative results on long videos, which necessitate a streaming method. Recall, Fig. 6 shows a long term temporal coherence of StreamGBH. In the segmentation, objects have consistent indices along the time duration as long as the shot is not changed (see below for a shot change detection example). We see different details of an object in the layers of hierarchy. For example, one can find three papers in the violin case in 5th layer, and locate a single musician body pose in 14th layer. Note, however, the person walking behind the musician gets mixed into the musician's segment at frame 401 at higher layers in the hierarchy; this is a shortcoming of our method due to the streaming assumptions. Fig. 9 (top) is a shot cut from a long video with changes in illumination as a person enters a kitchen. StreamGB and StreamGBH have better temporal coherence than the others. However, the arm of the person is only able to be segmented out by StreamGBH; StreamGB has some small noise fragments and Clip-based GB completely loses the arm on Frame 513. We also show how StreamGBH captures a shot change in a video as Frame 931 to Frame 932 in Fig. 9 (bottom). Our proposed StreamGBH has a promising performance even without any motion cues in the above experiments.

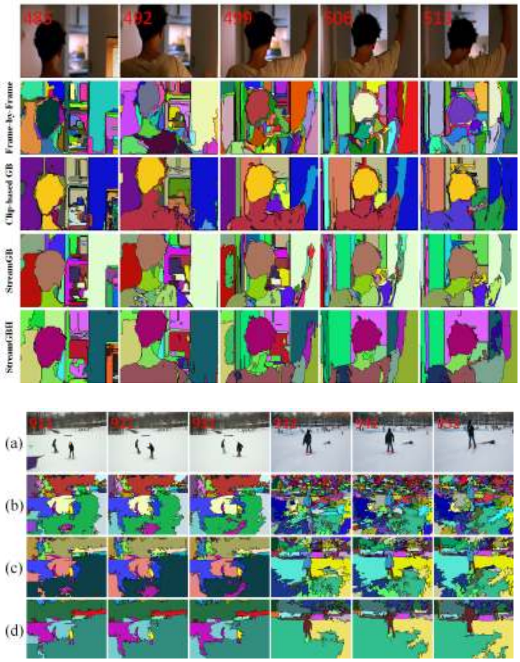

Fig. 9. (top) Qualitative comparison of $k = 10$ for the streaming methods. (bottom) Shot change of StreamGBH with $k = 10$. (a) the video with frame number on top-left, (b) the 5th layer segmentation, (c) the 10th layer segmentation, (d) the 14th layer segmentation.

4.3 Space and Time Complexity

Since our method adopts streaming techniques, at any time, we only need constant memory to keep the previous and current video subsequences in the memory. This constant memory is $O(nk)$, n is the number of voxels per frame and k is the length of video subsequence window; thus it is independent of the video length. For a p-frame video, there are $n \times p$ voxels in total. The complexity of GBH is $O(np \log np)$. Since our StreamGBH considers only k frames at each time, its complexity is $O(np \log nk)$. For an arbitrarily long video, $p \gg k$ and

$p \gg n$ where k and n are constant numbers, thus the complexity of StreamGBH is $O(p)$ whereas GBH is $O(p \log p)$. Therefore, StreamGBH is more efficient in both space and time complexity. For our experiments, the timing result on benchmark [10] for StreamGBH on average is 355 seconds for 85 frames (about 0.25 fps) generating 16 layer hierarchies; our method is not optimized or parallelized.

5 Conclusion, Limitations and Future Work

We have proposed a framework for streaming hierarchical video segmentation and a graph based streaming hierarchical approach (StreamGBH) based on our framework. To the best of our knowledge, it is the first solution for fully unsupervised online hierarchical video segmentation that can handle any arbitrarily long video (and our code is available). We evaluate this approach on a benchmark data set under different metrics and qualitatively on some long videos. The results show that our approach outperforms other state of the art streaming methods in both quantitative and qualitative perspectives. Although our StreamGBH can process videos of arbitrary length, our implementation is not real time. In future, we plan to improve this implementation to leverage parallelization hardware and optimized data structures from data stream algorithms.

Acknowledgements. This work was partially supported by the National Science Foundation CAREER grant (IIS-0845282), the Army Research Office (W911NF-11-1-0090), the DARPA Mind's Eye program (W911NF-10-2-0062), and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20069 The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/NBC, DARPA, ARO, NSF or the U.S. Government.

References

1. Megret, R., DeMenthon, D.: A Survey of Spatio-Temporal Grouping Techniques. Technical report, Language and Media Proc. Lab., U. of MD at College Park (2002)
2. Laptev, I.: On space-time interest points. IJCV (2005)
3. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: CVPR (2010)
4. Brendel, W., Todorovic, S.: Video object segmentation by tracking regions. In: ICCV (2009)
5. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: Spatio-temporal video segmentation with long-range motion cues. In: CVPR (2011)
6. Vazquez-Reina, A., Avidan, S., Pfister, H., Miller, E.: Multiple Hypothesis Video Segmentation from Superpixel Flows. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 268–281. Springer, Heidelberg (2010)

7. Wang, J., Xu, Y., Shum, H., Cohen, M.F.: Video tooning. In: ACM SIGGRAPH, pp. 574–583 (2004)
8. Bai, X., Sapiro, G.: Geodesic matting: A framework for fast interactive image and video segmentation and matting. IJCV 82(2), 113–132 (2009)
9. Paris, S., Durand, F.: A topological approach to hierarchical segmentation using mean shift. In: CVPR (2007)
10. Xu, C., Corso, J.J.: Evaluation of super-voxel methods for early video processing. In: CVPR (2012)
11. Sharon, E., Galun, M., Sharon, D., Basri, R., Brandt, A.: Hierarchy and adaptivity in segmenting visual scenes. Nature 442(7104), 810–813 (2006)
12. Corso, J.J., Sharon, E., Dube, S., El-Saden, S., Sinha, U., Yuille, A.: Efficient multilevel brain tumor segmentation with integrated bayesian model classification. TMI 27(5), 629–640 (2008)
13. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. IJCV 59(2), 167–181 (2004)
14. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the nyström method. TPAMI 26, 2004 (2004)
15. Paris, S.: Edge-Preserving Smoothing and Mean-Shift Segmentation of Video Streams. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 460–473. Springer, Heidelberg (2008)
16. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR (2011)
17. Ren, X., Malik, J.: Learning a classification model for segmentation. In: ICCV, vol. 1, pp. 10–17 (2003)
18. Levinshtein, A., Stere, A., Kutulakos, K.N., Fleet, D.J., Dickinson, S.J., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. TPAMI 31(12), 2290–2297 (2009)
19. Veksler, O., Boykov, Y., Mehrani, P.: Superpixels and Supervoxels in an Energy Optimization Framework. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 211–224. Springer, Heidelberg (2010)
20. Moore, A.P., Prince, S.J.D., Warrell, J., Mohammed, U., Jones, G.: Superpixel lattices. In: CVPR (2008)
21. Mori, G., Ren, X., Efros, A.A., Malik, J.: Recovering human body configurations: Combining segmentation and recognition. In: CVPR, vol. 2, pp. 326–333 (2004)
22. Tighe, J., Lazebnik, S.: SuperParsing: Scalable Nonparametric Image Parsing with Superpixels. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 352–365. Springer, Heidelberg (2010)
23. Lee, Y.J., Kim, J., Grauman, K.: Key-segments for video object segmentation. In: ICCV (2011)
24. Muthukrishnan, S.: Data streams: Algorithms and applications. Foundations and Trends in Theoretical Computer Science 1(2) (2005)
25. Chen, A.Y.C., Corso, J.J.: Propagating multi-class pixel labels throughout video frames. In: Proc. of Western NY Image Proc. Workshop (2010)
26. Shotton, J., Winn, J., Rother, C., Criminisi, A.: TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. IJCV 81(2), 2–23 (2009)