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

Abstract

Video segmentation has been used in a variety of computer vision algorithms as a pre-processing step. Despite its wide application, many 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 online, hierarchical video segmentation algorithm with no latency. The new algorithm leverages a graph-based image segmentation technique and recent advances in dense optical flow. Our contributions 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. We conduct a thorough experimental analysis of our algorithm on a benchmark dataset and compare it with state-of-the-art algorithms. The results indicate that our algorithm achieves comparable or better segmentation accuracy than state-of-the-art batch-processing algorithms, and outperforms streaming algorithms despite a significantly lower computation cost, which is required for time-critical applications.

1 Introduction

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. Examples include scene understanding [10], shadow/lighting estimation [14], and robotics [11]. 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. Video segmentation works in a higher dimensional space (i.e., the spatio-temporal 3D volume of the video) instead of the 2D image space in the image segmentation case. The segmentation needs to respect the 3D volume boundaries between objects in the spatio-temporal space instead of just 2D boundaries in images.

Scalability and efficiency. Many existing algorithms pre-load the entire video and batch process all the frames as a 3D spatial-temporal volume. However, this usually requires a

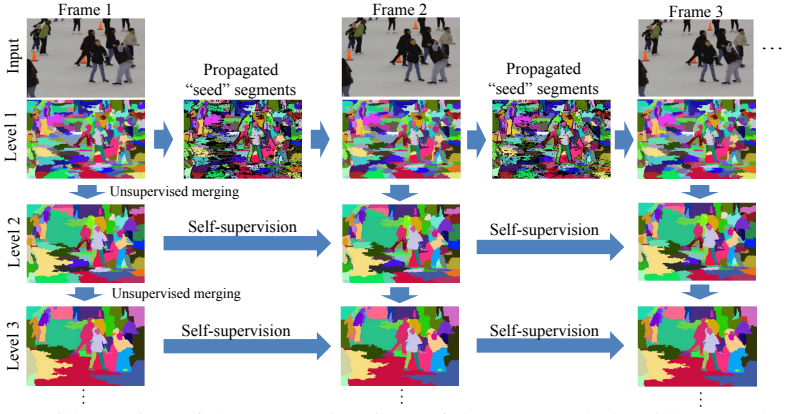


Figure 1: An illustration of the processing flow of the proposed algorithm. Each color corresponds to a supervoxel.

large amount of memory and inevitably limits the maximum length of videos that can be processed. Also, it usually results in a high computation cost and cannot support time-critical applications such as robotics and real-time surveillance. Some recent algorithms divide the video stream into consecutive clips with overlapping frames and batch process each clip sequentially and utilize the overlapping frames to infer the temporal consistency across clips. This way, it achieves streaming processing but still has the algorithmic latency (i.e., wait for forthcoming frames to form the clip). Also, the segmentation quality deteriorates quickly when the clip length approaches to 1 frame [21].

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 (see Section 4.1.) 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 (see Section 4.2.) 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 (see Section 4.3). 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 (see Section 4.4.) 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 [20], and use a wide range of performance metrics to thoroughly compare it with multiple state-of-the-art algorithms (see Section 5.)

2 Related Work

Extended from the 2D image segmentation, video segmentation has attracted considerable research attention recently due to its strong practical relevance and abundant applications.

One popular class of algorithms is the graph-based approaches [2, 6, 16]. In [6], a graph based image segmentation algorithm is proposed. Grandmann et al. [9] extend the approach to a hierarchical video supervoxel segmentation. Our algorithm borrows the graph-based concept. In particular, we segment the first frame using the approach as in [9]. In addition, we leverage the latest dense optical flow to assist propagating the labels to the next frame and create an initial graph of the next frame for graph-based segmentation. The process repeats iteratively on forthcoming frames to produce the supervoxel segmentation of the video.

Many state-of-the-art algorithms such as [9, 13] require to load the entire video. The consequent high memory cost significantly limits the maximum length of video that can be processed. Also, the computation time of this class of algorithms is usually very high. A hierarchical streaming video segmentation algorithm [21] divides the entire video into consecutive K-frame small clips with one frame overlapping between adjacent clips. Each clip is batch-processed and the overlapping frame is used to infer the correspondence of the segmentation results across clips. Although the algorithm reduces to the case that has no algorithmic latency when $K = 1$ (i.e., truly online mode), the quality of the segmentation deteriorates significantly (see both [21] and Section 5.2.) Our algorithm runs in a completely online manner. It does not require to either pre-load the video or periodically load a streaming buffer of the video. It processes every incoming frame and produce the video segmentation up to this frame immediately without waiting for future frames, i.e., no algorithmic latency. This way, both the computation and memory cost of our algorithm are very low (see Table 1 in Section 5.4.)

With the recent maturity of optical flow techniques such as [3], many recent segmentation algorithms start to utilize the optical flow for both intra-frame pixel clustering and inter-frame label consistency. In [7], affinity scores between pre-extracted superpixels across all frames are computed with both motion (long-term point trajectories) and appearance cues. Supervoxel segmentation is realized by a spectral clustering based on these affinity scores. In [13], long term voxel trajectories are computed with the dense optical flow across frames. Supervoxel segmentation is realized by clustering these voxel trajectories with additional appearance cues. Both of these algorithms are essentially still patch-processing algorithms. Computing and introducing the optical flow information further complicates the computation. As a result, they are computationally very expensive[1]. Although we also utilize the dense optical flow, our algorithm uses it in a novel way for label propagation and only requires the flow between consecutive frames at a time (i.e., no requirement for long-term point trajectory). As a result, our algorithm runs in an online manner and is very efficient (see Table 1 in Section 5.4.)

Lately, there are a few efforts that compute superpixels of each frame, such as SLIC [1] and SEEDS [17], and then link them sequentially across consecutive frames [4, 18], or track individual superpixels across frames [19]. However, the granularity of the segmentation (e.g., number of superpixels per frame) is pre-determined. Our algorithm is able to hierarchically segment the video from fine to coarse with layered granularity, which better aligns with human's perception and is more useful for segmenting an object of interest in scene as a complete region instead of multiple small supervixels with similar size and appearance.

3 Supervoxel Segmentation Problem and Evaluation

Any video $\mathcal{V}$ consists of a saptio-temporal 3D lattice of pixels $\mathbb{V} = \mathbb{F} \times \mathbb{Z}$, where $\mathbb{F}$ denotes the frame pixel lattice, and $\mathbb{Z}$ is the frame indices. Then video $\mathcal{V}$ can be considered as the

[1] [13] is reported to take 1000 seconds and 20 GB memory for 3 million voxels. [7] is reported as the slowest algorithm as in [9].

mapping from lattice $\mathbb{V}$ to color space $\mathbb{R}^3$, i.e., $\mathcal{V}:\mathbb{V}\rightarrow\mathbb{R}^3$. A supervoxel segmentation $\mathbb{S}=\{s_i|i=1,2,...\}$ of the video is a set of mutually exclusive subsets of $\mathbb{V}$, which satisfies:

1. $\mathbb{V}=\bigcup_i s_i$.
2. $s_i\cap s_j=\phi$ with $i\neq j$, and i and j are supervoxel indices.
3. Any pixel $p\in s_i$ is labelled by label $\psi_i\in\Psi$.
4. Given frame at time t, F^t, then $s_i\cap F^t$ is a connected component (superpixel) in F^t.

The elements in $\mathbb{S}$ are called supervoxels and the boundaries between the supervoxels are called 3D boundaries $B_{\mathbb{S}}$. Given a video $\mathbb{V}$, the task is to find the segmentation $\mathbb{S}$ such that its difference to the ground-truth segmentation $\mathbb{G}=\{g_i|i=1,2,...\}$ is minimized,

$$\mathbb{S}^*=\arg\min_{\mathbb{S}}\text{diff}(\mathbb{S},\mathbb{G}),$$

where the function $\text{diff}(\cdot,\cdot)$ returns the measure of difference between $\mathbb{S}$ and $\mathbb{G}$.

It is usually difficult to model a single scalar $\text{diff}(\mathbb{S},\mathbb{G})$ to cover all aspects of the segmentation quality (e.g., temporal consistency, boundary identification, etc.) Therefore, it is a common practice to evaluate segmentations against different specific metrics, each reflecting some aspect of the objective function (see Section 5.2.) Furthermore, different people may have different opinions towards the ground truth segmentation. This motivates the hierarchical segmentation as in this work, which is able to automatically segment the video with different granularity. Hence, a thorough evaluation of a supervoxel segmentation algorithm needs to evaluate the objective function/metrics under different granularity.

To facilitate the paper presentation from here on, we define the following index notation conventions: i and j are used as indices of supervoxel or superpixel, u and v are used as pixel coordinates in images, l is used as segmentation layer index, and t is the frame index.

4 Approach

4.1 Graph-based Hierarchical Image Segmentation

Our algorithm starts with the segmentation of the first frame. We adopt the graph-based hierarchical segmentation algorithm [9]. Here we briefly review the algorithm.

The first frame is initially modelled as a graph with each pixel at (u,v) being a vertex. Each vertex is connected to one of its (up to) 8 neighbors by an undirected edge e, whose weight $w(e)$ is the color difference between the two pixels. Initially, each vertex is a superpixel R on its own. The internal variation of any superpixel R is defined as $\text{RInt}(R)\triangleq\text{Int}(R)+\frac{\tau}{|R|}$, where $\text{Int}(R)=\max_{e\in\text{MST}(R)}w(e)$ is the maximum edge weight in the Minimum Spanning Tree (MST) of R, $|\cdot|$ returns the cardinality of a (pixel) set, and τ is a parameter that controls the granularity of the segmentation. The edges are sorted in an ascending order based on their weights. Each edge is then visited in the ascending order to check if it should be cut and the superpixels it connects should merge. Given an edge $e_{a,b}$, which connects pixels a and b, and $a\in R_a$ and $b\in R_b$, then the merging condition is,

$$I(e_{a,b})=\begin{cases}1 & \text{if } \ w(e_{a,b})<\min\left(\text{RInt}(R_a),\text{RInt}(R_b)\right),\\ 0 & \text{otherwise.}\end{cases}$$

By traversing all edges, we have an over-segmention of the frame with many small homogenous color patches called superpixels. We term this as level 1 segmentation (see Figure 1.)

For hierarchical segmentation, each superpixel is treated as a vertex in the graph. The edge weight $w(e)$ is replaced by the χ^2 distance in the $L^*a^*b^*$ color space between two

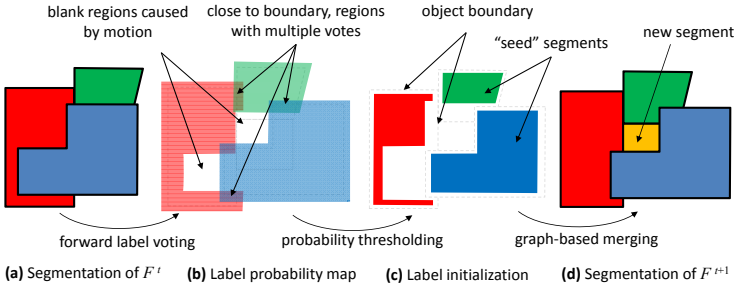


Figure 2: An illustration of the label propagation. The blue object moves towards right from F^t to F^{t+1}. (a) The segmentation of frame F^t. (b) Label probability map after forward label voting. The regions close to object boundaries usually have multiple label votes (with overlapping color) due to Gaussian kernel smoothing (see (2).) Some regions has no label votes due to motion. (c) "Seed" segments propagated from the previous frame after thresholding on the label probability map (see (5).) A new graph with the seed segments is built for F^{t+1}. (d) Final segmentation for F^{t+1}. New appearing object is identified.

neighboring superpixels, and parameter τ increases appropriately to allow further merging of superpixels. After traversing and cutting remaining edges in the ascending weight order, we obtain a higher level segmentation with a new set of superpixels which are parents of those on level 1 (see Figure 1.) Finally, the process above is iteratively repeated to obtain higher level segmentations. The Hierarchy of the segmentations defines a structure of forest. Details of the hierarchical graph-based segmentation can be found in [9].

Now we have the layered supervpixels for the first frame. Next we try to propagate the labels onto the forthcoming frames to form layered supervoxel segmentations.

4.2 Initial Multi-cue Probabilistic Label Propagation

Given the layered segmentations $\{\mathbb{S}_l^{1:t} | l = 1,2,...\}$ up to frame F^t, and the next incoming frame F^{t+1}, we shall extend the segmentations to F^{t+1}.

4.2.1 Label propagation.

We first compute the forward optical flows from F^t to F^{t+1}, denoted as $\overrightarrow{f_u}$ and $\overrightarrow{f_v}$, which "project" pixel coordinates from $(u^t, v^t) \in F^t$ to $(\hat{u}^{t+1}, \hat{v}^{t+1}) \in F^{t+1}$ by

$$(\hat{u}^{t+1}, \hat{v}^{t+1}) = (u^t, v^t) + \left(\overrightarrow{f_u}(u^t, v^t), \overrightarrow{f_v}(u^t, v^t)\right). \tag{1}$$

Here we use the hat notation "^" to indicate the "projected" coordinates. Similar to (1), we compute the backward flows $\overleftarrow{f_u}$ and $\overleftarrow{f_v}$. With the forward flow as in (1), given any pixel at (u^t, v^t), we project its level 1 label, denoted as $\Psi_1^t(u^t, v^t)$, onto F^{t+1} by voting in a neighborhood around $(\hat{u}^{t+1}, \hat{v}^{t+1})$ with Gaussian kernel weights:

$$\omega_{u,v}^{t+1}(\psi | u^t, v^t) = \begin{cases} \mathcal{N}\left((u - \hat{u}^{t+1}, v - \hat{v}^{t+1}), \Sigma_f\right) & \text{if } \Psi_1^t(u^t, v^t) = \psi \text{ and } (u,v) \in \mathbb{N}((u^{t+1}, v^{t+1})), \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathcal{N}(\cdot)$ is the Gaussian function, Σ_f is the 2×2 covariance matrix indicating the Gaussian kernel size, and $\mathbb{N}(\cdot)$ returns the neighborhood of a pixel in the frame.

Due to occlusions in the scene, we cannot trust all the forward optical flows. Here we use the bi-directional consistency to determine if a flow should be filtered out.

$$I_f(u^t,v^t)=\begin{cases}1 \text{ (filtered out)} & \text{if } \|(u^t,v^t)-\left(\overleftarrow{f_u}(\hat{u}^{t+1},\hat{v}^{t+1}),\overleftarrow{f_v}(\hat{u}^{t+1},\hat{v}^{t+1})\right)\|_2>\delta\\ 0 \text{ (not filtered out)} & \text{otherwise,}\end{cases}$$

where δ is a threshold. Then we vote all surviving pixels onto F^{t+1}, and accumulate the label weights at each pixel $(u,v)\in F^{t+1}$: $\omega_{u,v}^{t+1}(\psi)=\sum_{(u^t,v^t)\in F^t}\omega_{u,v}^{t+1}(\psi|u^t,v^t)$. After normalizing these lable weights at each pixel, we obtain a probability map of the the labels for F^{t+1} :

$$p_{u,v}^{t+1}(\psi)=\frac{\omega_{u,v}^{t+1}(\psi)}{\sum_{\psi'}\omega_{u,v}^{t+1}(\psi')}. \tag{3}$$

4.2.2 Motion Cue and Appearance Cue.

The label probability in (3) is based on optical flow, i.e., the object/scene motion information. Though we use a state-of-the-art optical flow algorithm, it is not perfect. Also some correspondence between frames is inevitably missing due to various factors such as occlusion, object disappearing and new object appearing. Therefore, besides the optical flow-based label propagation, we further propose an appearance-based label propagation.

With over-segmentation in F^t, each superpixel with label ψ, denoted as R_ψ^t, represents a homogeneous color patch. We model its appearance by a Gaussian model with its mean and covariance as $\mu_\psi^t=\text{mean}(a(R_\psi^t))$ and $\Sigma_\psi^t=\text{cov}(a(R_\psi^t))$, respectively, where function $a(\cdot)$ returns the appearance values of a set of pixels. In our work, we use the $L^*a^*b^*$ color to represent the appearance of pixels. Considering the pixels in corresponding superpixels across two frame should have similar appearance, we model the probability of any pixel $(u,v)\in F^{t+1}$ belonging to the superpixel P_ψ^{t+1} (corresponding to P_ψ^t) as

$$q_{u,v}^{t+1}(\psi)=\mathcal{N}\left(a(F^{t+1}(u,v))-\mu_\psi^t,\Sigma_\psi^t\right). \tag{4}$$

Fusing both motion-based label probability in (3) and appearance-based label probability in (4), we obtain a final label probability map, $\xi_{u,v}^{t+1}(\psi)=p_{u,v}^{t+1}(\psi)q_{u,v}^{t+1}(\psi)$. The fused label probability map is illustrated in Figure 2(b). It is noticed that regions close to object boundaries are likely to have multiple label votes due to the Gaussian kernel smoothing as in (2.) It is also noticed that some pixels do not have any vote because they were occluded in F^t and appear in F^{t+1}, and thus have no forward optical flow information. With this label probability map, we are ready to propagate the level 1 labels from F^t to F^{t+1} :

$$\psi_{u,v}^*=\arg\max_\psi \xi_{u,v}^{t+1}(\psi),\quad \Psi_1^{(t+1)}(u,v)=\begin{cases}\psi_{u,v}^* & \text{if } \psi_{u,v}^*>\eta\\ \phi\end{cases} \tag{5}$$

where η is a threshold and $\Psi_1^{(t+1)}(u,v)$ being empty set ϕ means the level 1 label for (u,v) is not determined yet.

Figure 2(c) illustrates the initial labeling of F^{t+1} after label propagation using (5). Compared to Figure 2(b), there are more regions/pixels that are not colored/labelled due to relatively low confidence of correspondence in either motion or appearance across two frames. Some pixels are not labelled due to color/illumination changes across two consecutive frames.

Furthermore, those pixels close to the object boundaries having multiple label votes are thus not likely to be labelled due to the voting smoothing as in (2). This is desirable for our initial labeling since it does not ruin the supervoxel boundaries. Our label propagation is essentially different from that in GBH [9] where optical flow is used to construct inter-frame graph.

4.3 Complete the Labelling Using Graph-based Merging

According to (5), the labelled pixels have high confidence of correspondence to superpixels in F^t. We identify these pixel sets as core representatives of the superpixels in F^{t+1} and use them as the "seeds" for completing the labelling. In particular, we first model F^{t+1} as a new graph: For each seed segment, we cut its internal edges (i.e., merging) and compute its internal variation. The rest pixels are treated as individual superpixels on their own. Then we apply the graph-based image segmentation on this new graph, same as the process in Section 4.1. This way, the algorithm gracefully handles the birth and death of superpixels, as well as splitting of old superpixels, instead of human specification for maintaining fixed number of superpixels as in [4, 18]. Figure 2(d) shows the final level 1 labelling of F^{t+1}. Since the edges across boundaries usually have high weights and survive the initial labelling, they are less likely to be cut than those edges connecting the seed segments to other unlabelled pixels (on the same side of the boundary.) This way, the boundaries between superpixels are preserved. If new objects appear, they are unlikely to be labelled in the initial labelling stage as explained earlier. Therefore, they are more likely to be labelled as new superpixels in this stage. Now we have propagated the level 1 labels from F^t to F^{t+1}. Next, we extend the labels to higher levels.

4.4 Self-supervised Hierarchical Labeling

Denote the level $(l+1)$ labelling of F^t as P^t_{l+1} and the level l labelling of F^{t+1} as P^{t+1}_l, the objective here is to infer the level $(l+1)$ labelling of F^{t+1}, denoted as P^{t+1}_{l+1}. First, we transfer the label merging knowledge (from level l to $l+1$) in F^t to that in F^{t+1}. Given any pair of superpixels $R^{t+1}_{l,i}, R^{t+1}_{l,j}$ in P^{t+1}_l, and they have their corresponding superxiels in F^t denoted as $R^t_{l,i}, R^t_{l,j}$. If $R^t_{l,i}$ and $R^t_{l,j}$ were merged on level $l+1$ in F^t, we cut the edges between $R^{t+1}_{l,i}$ and $R^{t+1}_{l,j}$ and merge them correspondingly. This way, the hierarchical relation between superpixels is propagated from F^t to F^{t+1}. Figure 1 illustrates the process of such self-supervised hierarchical labelling. After that, we apply the graph-based merging as described in Section 4.1 on the remaining superpixels.

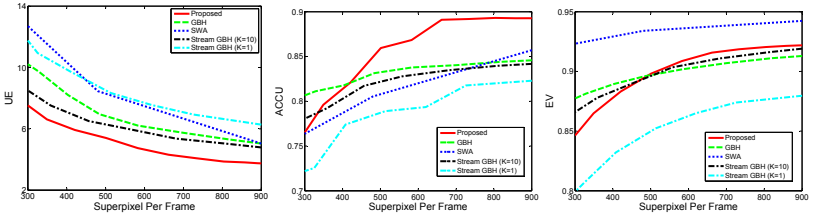
Now we have completed our description of the the hierarchical segmentation from F^t to F^{t+1}. The same process is iteratively repeated when new frame is coming to realize the online hierarchical video segmentation (see Figure 1.)

5 Experiment

To evaluate our new algorithm, we carry out experiments and a thorough quantitative analysis on a well-accepted benchmark dataset, LIBSVX [20], and compare our algorithm with multiple state-of-the-art algorithms.

5.1 Compared Algorithms

We consider three state-of-the-art algorithms, namely, Segmentation by Weighted Aggregation (SWA) [5, 15], Graph-Based Hierarchical segmentation (GBH) [9], and Streaming Graph-Based Hierarchical segmentation (StreamGBH) [21]. In particular, SWA and GBH are offline algorithm which load the video at once. According to both [8] and [20], GBH is one of the top-performing algorithms. StreamGBH loads a buffer of K frames at a time.



(a) 3D UE (b) 3D ACCU (c) EV

Figure 3: Quantitative comparison with granularity-dependent metrics: (a) 3D undersegmentation error (UE), (b) 3D segmentation accuracy (ACCU), (c) explained variation (EV).

Here we test and compare two of its variations with $K = 10$ and $K = 1$, respectively. It is worth emphasizing that the only fair comparison for our algorithm is the StreamGBH with $K = 1$, since it is the only algorithm that is 1) truly online and 2) hierarchical. Though it is not our focus to beat the best-performing algorithms (in terms of segmentation quality), we still include some offline and streaming algorithms for a comprehensive analysis.

5.2 Performance Metrics and Quantitative Comparison

LIBSVX [20] proposes multiple metrics for evaluating different aspects of the algorithm performance. Here we use most of its metrics as briefed below. However, it is also noticed that the metrics in LIBSVX focus very much on the recall/identification of the 2D/3D boundary/volume, and the precision of the boundary/volume identification as well as the temporal consistency of labels is largely ignored. Almost all metrics in [20] are granularity-dependent and finer granularity of segmentation always produces higher performance scores. To overcome this issue and give a more thorough evaluation, we extend some metrics in [20] and also borrow some other metrics from literatures, detailed below.

3D Undersegmentation Error (UE). Introduced in [20], this metric measures the fraction of computed supervoxels exceed the annotated volumes.

$$\mathrm{UE}(g_i) = \frac{\left(\sum_{\{j|s_j \cap g_i \neq \phi\}} |s_j|\right) - |g_i|}{|g_i|}$$

defines the 3D UE for a given annotated supervoxel. Following [20], we average it across all annotated supervoxels in the video as our metric. In general, finer granularity of the segmentation would lead to lower UE. Figure 3(a) summarizes the UEs for different algorithms. Different from [20] which plots UE against the average number of supervoxels, considering different video length leads to different number of supervoxels and the segmentation granularity is visually better perceived by number of super pixels in individual frames, here we plot the 3D UE against the average number of superpixels per frame. It is shown that our algorithm outperforms all the counterparts.

3D Segmentation Accuracy (ACCU). This metric [20] measures the fraction of an annotated segment that is correctly classified. Ideally, each computed supervoxel overlaps with only one annotated segment. For each annotated segment g_i, we collect the set of computed supervoxels that each has more than half of its voxels (i.e. majority of its voxels) overlapping with g_i, then the 3D Segmentation Accuracy (ACCU) of g_i is defined as

$$\mathrm{ACCU}(g_i) = \frac{\sum_{\{j||s_j \cap g_i| > |s_j|\}} |s_j \cap g_i|}{|g_i|}.$$

Figure 3(b) summarizes the comparison between different algorithms using ACCU. It is

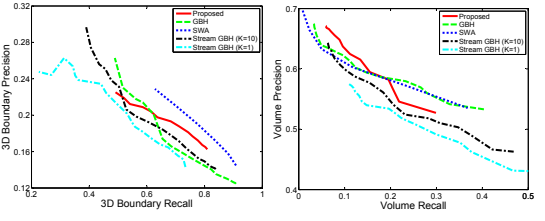
shown that our algorithm has a consistently higher ACCU than StreamGBH with $K = 1$, and generally outperforms all other algorithms.

Explained Variation (EV). Unlike other metrics, explained variation [12, 20] is independent with the human annotation:

$$\mathrm{EV} = \frac{\sum_i (\mu_i - \mu)^2}{\sum_i (x_i - \mu)^2},$$

where x_i is the actual voxel value, μ is the global voxel mean and μ_i is the mean value of the supervoxel that contains x_i. Figure 3(c) summarizes the EV scores for different algorithms. SWA shows to have the best performance on EV. Our algorithm is comparable to GBH and StreamGBH with $K = 10$, and consistently outperforms StreamGBH with $K = 1$ across different granularity.

3D Boundary Recall and Precision. In [20], 3D boundary recall is proposed as a metric. However, it inevitably favors finer granularity. Higher 3D boundary recall is always achieved at the price of finer granularity, while the precision of the 3D boundaries is ignored. Here we extend the 3D boundary recall metric in [20] to both recall and precision,



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

Figure 4: Comparison of Precision-Recall.

$$R_b = \frac{|B_{\mathbb{S}} \cap B_{\mathbb{G}}|}{|B_{\mathbb{G}}|}, \; P_b = \frac{|B_{\mathbb{S}} \cap B_{\mathbb{G}}|}{|B_{\mathbb{S}}|}.$$

Figure 4(a) shows the 3D boundary PR of all algorithms[2]. SWA appears to have the best PR tradeoff. Our algorithm is comparable to GBH and StreamGBH with $K = 10$, and outperforms StreamGBH with $K = 1$.

3D Volume Recall and Recision. Besides good boundaries, what is equally important for video segmentation is to correctly identify the spatio-temporal 3D volume. Here we adopt the volume recall and precision as proposed in [8]:

$$R_v = \frac{\sum_{g \in \mathbb{G}} (\max_{s \in \mathbb{S}} |s \cap g| - 1)}{\sum_{g \in \mathbb{G}} |g| - \mathrm{card}(\mathbb{G})}, \; P_v = \frac{\left(\sum_{s \in \mathbb{S}} \max_{g \in \mathbb{G}} s \cap g\right) - \max_{g \in \mathbb{G}} |g|}{\sum_{\{s,g | s \cap g \neq \phi\}} |s| - \max_{g \in \mathbb{G}} |g|},$$

where $\mathrm{card}(\cdot)$ returns the cardinality of a set. Figure 4(b) shows the 3D volume precision-recall of all algorithms. Our algorithm is comparable to GBH and SWA and outperforms the two StreamGBH variations.

5.3 Qualitative Comparison

Besides the quantitative comparison above, Figure 5 shows a qualitative comparison based on a set of sample segmentations using different algorithms (see more examples in the supplementary material.) All segmentation results are at similar granularity. It is shown that the results of SWA and our algorithm result are more plausible to human perception, in terms of the intactness/regularity of the object shape.

[2]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.

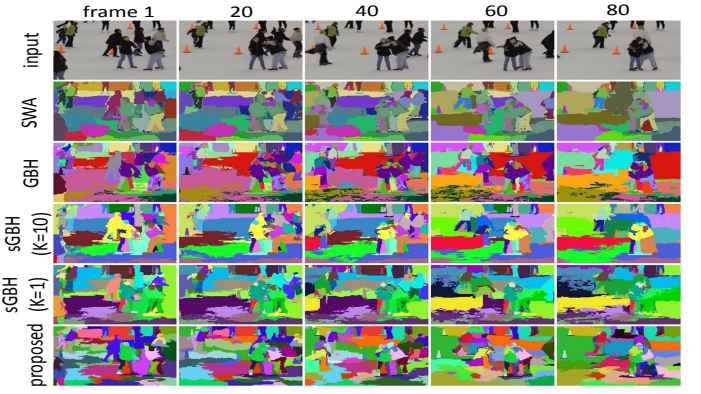


Figure 5: A qualitative comparison of algorithms with similar granularity of segmentation.

		Proposed	StreamGBH		GHB	SWA[3]
			$K=1$	$K=10$		
Time (sec.)	per input frame[4]	0.72	4.27	8.23	12.96	5.88
	per frame 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.

5.4 Computation Cost

Besides the segmentation quality comparison, our algorithm inherently supports online processing, which leads to much more efficient computation. Table 1 summarizes the computation time and memory requirement for all algorithms on the benchmark dataset. All tests are carried out on a laptop PC with a 1.8 GHz dual-core CPU. For fair comparison: 1) We set all algorithms but SWA to produce 21 layers of segmentations. SWA is set to produce 6 layers to save time; 2) We use fixed-length videos (85 frames) since the computational times of batch-processing algorithms (GBH and SWA) are super-linear to the video length. 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. Table 1 summarizes the memory requirements for all algorithms. Our algorithm clearly requires the least memory.

6 Conclusion

We reported a new efficient online hierarchical supervoxel segmentation algorithm, which leverages the graph-based image segmentation technique and the latest advance in dense optical flow. We validated our algorithm on a benchmark dataset and compared it with the state-of-the-art algorithms. The comparison showed that our algorithm achieved comparable segmentation quality to the offline algorithms and outperformed the streaming algorithms, despite significantly less computation time and memory, which supports time-critical tasks.

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

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

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *T-PAMI*, 34(11):2274–2282, 2012.

[2] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *IJCV*, 70(2):109–131, 2006.

[3] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *T-PAMI*, 33(3):500–513, 2011.

[4] Jason Chang, Donglai Wei, and John W Fisher III. A video representation using temporal superpixels. In *CVPR*, pages 2051–2058. IEEE, 2013.

[5] 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.

[6] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[7] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. Video segmentation with superpixels. In *ACCV*, pages 760–774. Springer, 2012.

[8] 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.

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

[10] Ruiqi Guo and Derek Hoiem. Beyond the line of sight: labeling the underlying surfaces. In *ECCV*, pages 761–774. Springer, 2012.

[11] Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Efficient temporal consistency for streaming video scene analysis. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 133–139, 2013.

[12] Alastair P Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *CVPR*, pages 1–8, 2008.

[13] Guillem Palou and Philippe Salembier. Hierarchical video representation with trajectory binary partition tree. In *CVPR*, 2013.

[14] A. Panagopoulos, Chaohui Wang, D. Samaras, and N. Paragios. Simultaneous cast shadows, illumination and geometry inference using hypergraphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(2):437–449, 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.110.

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

[16] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *T-PAMI*, 22 (8):888–905, 2000.

[17] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: superpixels extracted via energy-driven sampling. In *ECCV*, pages 13–26. Springer, 2012.

[18] Michael Van den Bergh, Gemma Roig, Xavier Boix, Santiago Manen, and Luc Van Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013.

[19] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011.

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

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