Joint Graph Learning and Video Segmentation via Multiple Cues and Topology Calibration

Jingkuan Song
University of Trento
Trento, Italy
jingkuan.song@unitn.it

Lianli Gao
University of Electronic Science and Technology of China, Chengdu, China
lianli.gao@uestc.edu.cn

Mihai Marian Puscas
University of Trento
Trento, Italy
mihaimarian.puscas@unitn.it

Feiping Nie
Northwestern Polytechnical University
Xi'an, China
feipingnie@gmail.com

Fumin Shen
University of Electronic Science and Technology of China, Chengdu, China
fshen@uestc.edu.cn

Nicu Sebe
University of Trento
Trento, Italy
niculae.sebe@unitn.it

ABSTRACT

Video segmentation has become an important and active research area with a large diversity of proposed approaches. Graph-based methods, enabling top performance on recent benchmarks, usually focus on either obtaining a precise similarity graph or designing efficient graph cutting strategies. However, these two components are often conducted in two separated steps, and thus the obtained similarity graph may not be the optimal one for segmentation and this may lead to suboptimal results. In this paper, we propose a novel framework, *joint graph learning and video segmentation (JGLVS)*, which learns the similarity graph and video segmentation simultaneously. JGLVS learns the similarity graph by assigning adaptive neighbors for each vertex based on multiple cues (appearance, motion, boundary and spatial information). Meanwhile, the new rank constraint is imposed to the Laplacian matrix of the similarity graph, such that the connected components in the resulted similarity graph are exactly equal to the number of segmentations. Furthermore, JGLVS can automatically weigh multiple cues and calibrate the pairwise distance of superpixels based on their topology structures. Most noticeably, empirical results on the challenging dataset VSB100 show that JGLVS achieves promising performance on the benchmark dataset which outperforms the state-of-the-art by up to 11% for the BPR metric.

CCS Concepts

•**Computing methodologies** → **Video segmentation;**

Keywords

Video segmentation; Graph-based method; multiple cues; Topology

MM '16, October 15-19, 2016, Amsterdam, Netherlands
© 2016 ACM. ISBN 978-1-4503-3603-1/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2964284.2964295

1. INTRODUCTION

Video segmentation can be defined as partitioning a video into several disjoint spatial-temporal regions such that each region has consistent appearance and motion. In the last few years, it has achieved an extraordinary success and has become a fundamental problem in a wide range of applications, such as object tracking, activity recognition and video summarization [15, 34, 31, 17]. Segmenting general and unconstrained videos is a challenging research problem due to existent scene and scale ambiguities of the segments [40] as well as the temporal-consistency constraints [14]. Different types of video segmentation algorithms have been recently introduced, from ones based on clustering [30, 9], to graph-based processing [14, 19, 10, 41] and tracking [3, 32, 34].

Among the existing video segmentation techniques, many successful ones benefit from mapping the video elements onto a graph which pixels/superpixels are nodes and edge weights measure the similarity between nodes. Cutting or merging is then applied on this graph to generate the video segments. Most of the existing graph-based methods focus on (i) what features to extract from each node; (ii) how to define a precise similarity graph and (iii) how to cut/merge the nodes effectively.

Meaningful features are necessary for good video segmentation. Previous work has extracted a variety of features [9, 40] from superpixels. To get the similarity graph, a graph topology is firstly designed according to the spatio-temporal neighborhood of the superpixels and the extracted features are used to weigh their edges. While standard similarity measures on the extracted features provide the basic way to calculate the similarity graph [30, 9], more recent work introduces learning a more precise similarity graph in either a supervised [19] or an unsupervised manner [20]. While supervised video segmentation methods [19, 40] can generally achieve better performance, the human annotation is time-consuming and the inherent video object hierarchy may be highly subjective. In contrast, a group of methods improve on cutting techniques [41, 10, 30, 14], which explicitly organize the image elements into mathematically sound structures based on the optimization of the predefined cutting loss function. One representative criterion is the normalized cut [14]. By minimizing a cutting cost objective function, the best segmentation can be obtained. This objective function is further proved to be equivalent to the generalized eigenvalue decomposition problem and a number of follow-ups proposed efficient solutions for this problem [23]. To reduce the computational cost, in [41, 10], fast partitioning methods that identify and remove between-cluster edges to form node clusters are proposed.

Graph cut methods provide well-defined relationships between the segments, but the problem of finding a cut in an arbitrary graph may be NP-hard. More importantly, because the graph similarity learning [13, 33, 35, 12, 25, 26] and the graph cutting are conducted in two separated steps, the learned graph similarity matrix may not be the optimal one for cutting, leading to suboptimal results. To tackle this problem, in this paper we propose a novel video segmentation framework: *Joint Graph Learning and Video Segmentation* (JGLVS), which learns the similarity graph and segmentations simultaneously. To summarize, the main contributions of this paper are:

- Our unsupervised video segmentation framework learns the similarity graph and cutting structure simultaneously to achieve the optimal segmentation results. We derive a novel and efficient algorithm to solve this challenging problem.
- We utilized multiple cues of the superpixels and the weights of different cues are automatically learned. Furthermore, we calibrate the similarity of different superpixels based on their topology structures to make them comparable.
- The proposed JGLVS achieves up to 11% improvement over the state-of-the-art baselines on the largest public dataset VSB100, which validates the effectiveness and efficiency of our approach.

The remainder of this paper is organized as follows. Section 2 discusses some related works. The details of JGLVS are introduced in section 3. Section 4 illustrates the experiments results and we draw a conclusion in section 5.

2. RELATED WORK

The relevant state-of-the-art methods on video segmentation are reviewed in this section. The problem definitions for video segmentation have been diverse.

Motion segmentation focuses on separating point trajectories from an image sequence with respect to their motion [24, 18, 29, 4]. In [29, 4], the segmentation is based on pairwise affinities, while in [28] third order terms are employed to explain not only translational motion but also in-plane rotation and scaling, and [42] models even more general 3D motions using group invariants. The actual grouping in these methods is done using spectral clustering. Differently, in [18], they formulate the segmentation of a video sequence based on point trajectories as a minimum cost multicut problem. Unlike the commonly used spectral clustering formulation, the minimum cost multicut formulation gives natural rise to optimize not only for a cluster assignment but also for the number of clusters while allowing for varying cluster sizes. Similarly, in [5], they utilize improved point trajectories to segment moving object in video by a graph-based segmentation method. And in [8], motion trajectory grouping in a setup similar to [4] is used to perform tracking. Although the grouping in [8] is computed using spectral clustering, repulsive weights computed from segmentation topology are used in the affinity matrix. In [24], they introduced minimal supervision, which is shown to be helpful to improve the performance of motion segmentations. In [22], they propose a framework to segment the objects in relative video shots, while discarding the irrelative video shots.

On the other hand, [41, 20, 9, 10] seek to construct full pixelwise segmentation, where every pixel (not only the moving objects) is assigned one of several labels. They can generally be divided into unsupervised and supervised methods.

A large body of literature exists on unsupervised video segmentation, with methods that leverage appearance [3, 14, 32, 38], motion [3, 16], or multiple cues [41, 20, 9, 10]. Unsupervised supervoxel generation [9, 2] has been widely accepted as a valuable preprocessing step for various techniques, such as graph-based methods [10, 9, 41, 14], hierarchical methods [14, 27, 38] and streaming methods [10, 38, 20]. Graph-based methods map the video elements onto a graph in which pixels/superpixels are nodes, and edge weights measure the similarity between them. Galasso *et al.* [9] proposed a frame-based superpixel segmentation approach (VSS) by extending the ultra-metric contour map [1] to combine with motion-cues and appearance-based affinities for obtaining better video segmentation performance. To deal with the high computational costs of spectral techniques, Galasso *et al.* [10] proposed a spectral graph reduction (SGR) method for video segmentation. They assumed that all pixels within a superpixel are connected by must-link constraints, and then reduced the original graph to a relative small graph such that a density-normalized-cut was preserved. Yu *et al.* [41] proposed an efficient and robust video segmentation framework based on parametric graph partitioning, resulting in a fast and almost parameter free method. On the other hand, hierarchical video segmentation provides a rich multi-scale decomposition of a given video. Grundmann *et al.* [14] proposed a hierarchical graph-based (HGB) video segmentation approach by firstly over-segmenting a volumetric video graph into space-time regions grouped by appearance, and then constructing a "region graph" over the obtained segmentation. Iteratively repeating this process over multiple levels results in a a tree of spatio-temporal segmentations. In order to process long videos, Xu *et al.* [38] proposed a streaming hierarchical video segmentation framework by integrating a graph-based hierarchical segmentation method with a data streaming algorithm (SHGB). This method leveraged ideas from data streams and enforced a Markovian assumption on the video stream to approximate full video segmentation. Li *et al.* [20] proposed a Sub-Optimal Low-rank Decomposition (SOLD) method, which defines a low-rank model based on very generic assumption that the intra-class supervoxels are drawn from one identical low rank feature subspace, and all supervoxels in a period lie on a union of multiple subspaces, which can be justified by natural statistic and observations of videos. In addition, this method adopts the Normalized-Cut (NCut) algorithm with a solved low-rank representation to segment a video into several spatio-temporal regions. To tackle the lack of a common dataset with sufficient annotation and the lack of an evaluation metric, a united video segmentation benchmark was provided by Galasso *et al.* [11] to effectively evaluate the over- and under-segmentation performance of video segmentation methods.

Supervised video segmentations [19, 21] can achieve better performance, but the human annotation is time-consuming and the inherent video object hierarchy may be highly subjective. In [21], they address the problem of integrating object reasoning with supervoxel labeling in multiclass semantic video segmentation. They first propose an object augmented dense CRF in spatio-temporal domain, which captures long-range dependency between supervoxels, and imposes consistency between object and supervoxel labels. Then, they develop an efficient mean field inference algorithm to jointly infer the supervoxel labels, object activations and their occlusion relations for a moderate number of object hypotheses. While in [19], they propose to combine features by means of a classifier, use calibrated classifier outputs as edge weights and define the graph topology by edge selection. Learning the topology provides larger performance gains and benefits efficiency due to a sparser structure of the constructed graph. On the other hand, lots of supervised image segmentations have been proposed [37]. In [39], they propose a novel discriminative deep feature learning framework based on stacked autoencoders (SAE) to tackle the problem of weakly supervised semantic segmentation. In [37], they use CNN to train images most only with image-level labels and very few with pixel-level labels for semantic segmentation.

Unsupervised full pixelwise segmentation is the research focus of this paper. A substantial difference between our approach and previous unsupervised work is that, instead of separately obtaining a graph and finding a cut in it, we propose a joint graph learning and video segmentation method by assigning adaptive neighbors for each superpixel and imposing a rank constraint on the Laplacian matrix of the similarity graph, such that the learned graph has exactly K connected components, representing K segmentations.

3. OUR APPROACH

In this section, we first introduce our JGLVS framework, and then elaborate on the details of each component.

3.1 The framework

In our JGLVS framework (see Fig. 1), we propose a novel perspective in solving the graph-based video segmentation problem. Our model makes use of superpixels instead of pixels for two reasons: a great decrease in the number of graph nodes that need to be processed, and an initial, accurate frame-level segmentation.

Firstly, in each temporal sliding window of the video, we extract N superpixels from M successive frames by setting a specific hierarchical level of an image segmentation algorithm [2]. Note that a too small value of N leads to large superpixels, and more under-segmentation errors, while a large value of N is computationally expensive. Then, for each superpixle, a set of features (e.g., appearance, motion and shape features) are extracted. Using these features and the predefined topology structure, our JGLVS framework can learn a similarity graph of superpixels which has exactly K connected components.

3.2 Feature extraction and graph topology construction

For each superpixel, we follow [2, 9] to extract *LAB*, *boundary*, *motion* and *shape* features, and use them to calculate the distance between two superpixels. However, not all of the superpixels are connected. By allowing different edge connections between neighbors, different graph topologies are constructed. Following [14, 19], edges may connect neighbors: *within frame* (if two superpixels share a common part of their contour or are close by in the spatial domain of the frame); *across 1 frame* (connected by coordinate correspondences over time); *across 2 frames* (connected by across-1 correspondences, further propagated over one more frame) and *across > 2 frames* (linked if overlapping with common long-term point trajectories).

We refer to these four types of neighbours as different topological structures $(1, 2, 3, 4)$ and record the topological structure of each pair of superpixels in a $N \times N$ matrix $\mathbf{W}$. Based on these features and topological structures, we can have the following pairwise distances between superpixels: common boundary strength (*cbs*), LAB (*lab*), boundary optical flow (*bof*), superpixel optical flow (*sof*), superpixel shape distance (*ssd*) and superpixel trajectory intersection (*sti*) (See Section 4 for details).

As shown in Table 1, different topological types have different distances. We further define a set of most-likely-linked superpixels $\mathbb{M}^1, \mathbb{M}^2, \mathbb{M}^3$ and $\mathbb{M}^4$ for each topological structure. More specifically, for the case of within frame, we decrease the number of superpixels by changing the threshold of superpixel generation algorithm, and some similar superpixels will merge into one superpixel. These similar superpixels will be selected as the within frame most-likely-linked superpixels. For the case of across 1 or 2 frame, if two superpixels' *ssd* distance is less than a threshold, they will be selected as a pair of across 1 or 2 frame most-likely-linked superpixels. Similarly, if two superpixels' *sti* distance is less than a threshold in the case of across >2 frame, they will be selected as a pair of across >2 frame most-likely-linked superpixels.

Table 1: The corresponding distances for different topological structures

Topology type	Distances
Within frame	lab, sof, cbs, bof
Across 1 frame	lab, sof, ssd, sti
Across 2 frames	ssd, sti
Across >2 frames	sti

3.3 Joint graph learning and video segmentation

Let $\mathbf{D}^t = \{\mathbf{D}_{ij}^t\}_{i,j=1}^N$ denote the t-th distance matrix of a set of N superpixels, where $t \in \{1, ..., T\}$. $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ is the average location information for the superpixels. The goal is to learn the similarity matrix $\mathbf{S}$ between superpixels by using different distances as well as existent spatial information, and that all the superpixels have exact K connected components.

An optimal graph $\mathbf{S}$ should be smooth on different features as well as on the spatial information distribution, which can be formulated as:

$$\min_{\mathbf{S},\alpha} \mathbf{g}(\mathbf{Y},\mathbf{S}) + \mu \sum\nolimits_{t=1}^{T} \alpha^t \mathbf{h}\left(\mathbf{D}^t,\mathbf{S}\right) + \beta \mathbf{r}(\mathbf{S},\alpha) \tag{1}$$

where $\mathbf{g}(\mathbf{Y},\mathbf{S})$ is the penalty function that measures the smoothness of $\mathbf{S}$ on the spatial information $\mathbf{Y}$ and $\mathbf{h}\left(\mathbf{D}^t,\mathbf{S}\right)$ is the loss function that measures the smoothness of $\mathbf{S}$ on the feature $\mathbf{D}^t$. $\mathbf{r}(\mathbf{S},\alpha)$ is a regularizer defined on the target $\mathbf{S}$ and α. μ and β are balancing parameters, and α^t determines the importance of each feature.

The penalty function $\mathbf{g}(\mathbf{F},\mathbf{S})$ should be defined in a way such that close superpixels have high similarity and vice versa. In this paper, we define it as follows:

$$\mathbf{g}(\mathbf{Y},\mathbf{S}) = \sum\nolimits_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} \tag{2}$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are the locations of the superpixels $\mathbf{x}_i$ and $\mathbf{x}_j$. Similarly, $\mathbf{h}\left(\mathbf{D}^t,\mathbf{S}\right)$ is defined as:

$$\mathbf{h}\left(\mathbf{D}^t,\mathbf{S}\right) = \sum\nolimits_{ij} d_{ij}^t s_{ij} \tag{3}$$

The regularizer term $\mathbf{r}(\mathbf{S},\alpha)$ is defined as:

$$\mathbf{r}(\mathbf{S},\alpha) = \|\mathbf{S}\|_F^2 + \gamma \|\alpha\|_2^2 \tag{4}$$

If there is no regularizer on $\mathbf{S}$ (same for α), $\mathbf{S}$ has a trivial solution. Only the nearest data point can be the neighbor of $\mathbf{x}_i$ with the probability of 1. We further introduce the following constraints: $\mathbf{S} \geq 0$, $\mathbf{S1} = \mathbf{1}$, $\alpha \geq 0$ and $\alpha^T\mathbf{1} = 1$, where $\mathbf{1}$ is a column vector with all 1s. This is because that the similarity and weights should be positive, and the sum of similarity and weights is set to be 1.

We can then obtain the objective function for learning the optimal graph by replacing $\mathbf{g}(\mathbf{Y},\mathbf{S})$, $\mathbf{h}\left(\mathbf{D}^t,\mathbf{S}\right)$ and $\mathbf{r}(\mathbf{S},\alpha)$ in (1) using

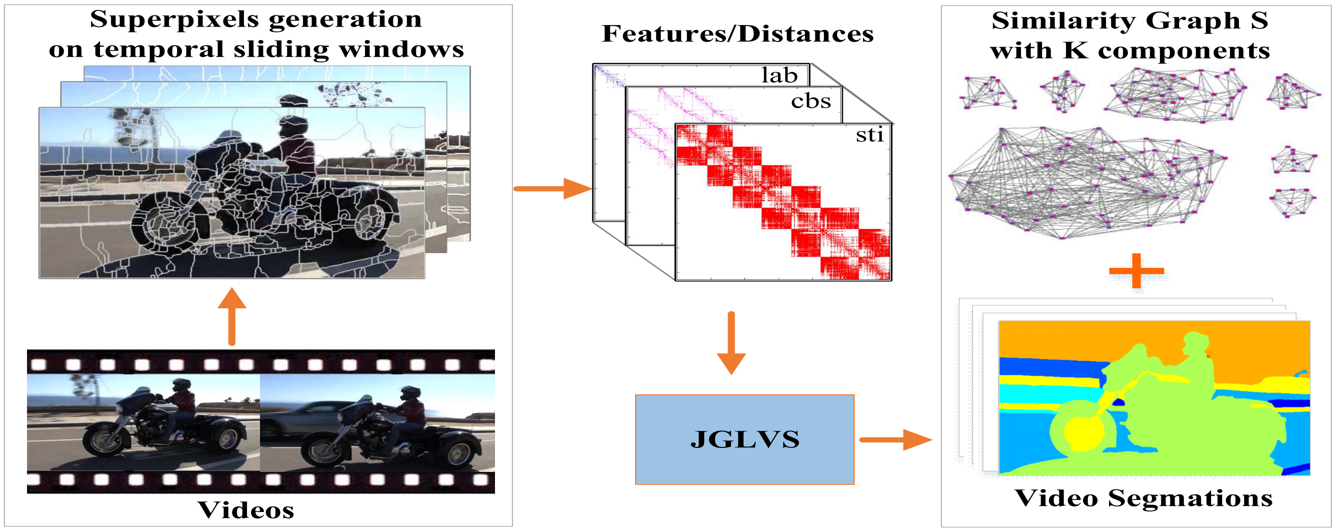


Figure 1: The overview of JGLVS. Superpixels are firstly generated from the overlapping sliding windows, based on which the features and distances are computed. Then, JGLVS is applied to learn the similarity matrix and video segmentations.

(2), (3) and (4), as follows:

$$\begin{array}{l} \min\limits_{\mathbf{S},\alpha} \sum\limits_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} + \mu \sum\limits_{tij} \left(\alpha^t d_{ij}^t s_{ij}\right) \\ +\beta \left\|\mathbf{S}\right\|_F^2 + \beta\gamma \left\|\alpha\right\|_2^2 \\ s.t.,\ \mathbf{S} \geq 0, \mathbf{S}\mathbf{1} = \mathbf{1}, \alpha \geq 0, \alpha^T\mathbf{1} = 1 \end{array} \tag{5}$$

One limitation for this model is that it assumes that all the superpixels have the same types of distances, which conflicts with the video segmentation application where different topologies have different distances. For example, if superpixels (i, k) are across $>$ 2 frames neighbors and (i, j) are within frame neighbors, the similarity between (i, k) are determined by *sti* but the similarity between (i, j) are determined by *lab, sof, cbs* and *bof*. Their distances are not comparable to each other, and we need to calibrate them. Based on the topology type $w_{ij} \in [1,2,3,4]$ of superpixels i and j, we define a calibration function

$$\mathbf{c}^z(x) = (x - \tau^z)/(max^z - \tau^z), z \in [1,2,3,4], \tag{6}$$

where τ^z is the threshold for z-th topology type determined by the mean distance of the set $\mathbb{M}^z$. Then, the objective function becomes:

$$\begin{array}{l} \min\limits_{\mathbf{S},\alpha} \sum\limits_{ij} \left\|y_i - y_j\right\|_2^2 s_{ij} + \mu \sum\limits_{ij} \mathbf{c}^{w_{ij}} \left(\sum\limits_t \alpha^t d_{ij}^t\right) s_{ij} \\ +\beta \left\|\mathbf{S}\right\|_F^2 + \beta\gamma \left\|\alpha\right\|_2^2 \\ s.t.,\ \mathbf{S} \geq 0, \mathbf{S}\mathbf{1} = \mathbf{1}, \alpha \geq 0, \alpha^T\mathbf{1} = 1 \end{array} \tag{7}$$

Forcing the number of connected components to be exactly K seems like an impossible goal since this kind of structured constraint on the similarities is fundamental but also very difficult to handle. In this paper, we will propose a novel but very simple method to achieve this goal.

The matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ obtained in the neighbor assignment can be seen as a similarity matrix of the graph with the N data points as the nodes. For a nonnegative similarity matrix $\mathbf{S}$, there is a Laplacian matrix $\mathbf{L}$ associated with it. According to the definition of Laplacian matrix, for any values of $\mathbf{f}_i \in \mathbb{R}^{K\times 1}$, $\mathbf{L}$ of a similarity matrix $\mathbf{S}$ can be calculated as:

$$\sum\nolimits_{ij} \left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2^2 s_{ij} = 2tr\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right) \tag{8}$$

where $\mathbf{F} \in \mathbb{R}^{N\times K}$ with the i-th row formed by $\mathbf{f}_i$, $\mathbf{L} = \mathbf{D} - \frac{\mathbf{S}^T+\mathbf{S}}{2}$ is called the Laplacian matrix in graph theory, the degree matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ is defined as a diagonal matrix where the i-th diagonal element is $\sum_j \left(s_{ji} + s_{ij}\right)/2$. The Laplacian matrix $\mathbf{L}$ has the following property.

THEOREM 1. *The number K of the eigenvalue 0 of the Laplacian matrix $\mathbf{L}$ is equal to the number of connected components in the graph with the similarity matrix $\mathbf{S}$ if $\mathbf{S}$ is nonnegative.*

Theorem 1 indicates that if $rank(\mathbf{L}) = N - K$, then the superpixels have K connected components based on S. Motivated by Theorem 1, we add an additional constraint $rank(\mathbf{L}) = N - K$ into the (7). Thus, our new similarity graph learning model is to solve:

$$\begin{array}{l} \min\limits_{\mathbf{S},\alpha} \sum\limits_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} + \mu \sum\limits_{ij} \mathbf{c}^{w_{ij}} \left(\sum\limits_t \alpha^t d_{ij}^t\right) s_{ij} \\ +\beta \left\|\mathbf{S}\right\|_F^2 + \beta\gamma \left\|\alpha\right\|_2^2 \\ s.t. \left\{ \begin{array}{l} \mathbf{S} \geq 0, \mathbf{S}\mathbf{1} = \mathbf{1}, \alpha \geq 0, \alpha^T\mathbf{1} = 1 \\ rank\left(\mathbf{L}\right) = N - K \end{array} \right. \end{array} \tag{9}$$

It is difficult to solve the problem (9). Because $\mathbf{L} = \mathbf{D} - (\mathbf{S}^T + \mathbf{S})/2$ and $\mathbf{D}$ also depends on $\mathbf{S}$, the constraint $rank(\mathbf{L}) = N - K$ is not easy to tackle. In the next subsection, we will propose a novel and efficient algorithm to solve this challenging problem.

3.4 Iterative optimization

Suppose e_i is the i-th smallest eigenvalue of $\mathbf{L}$, we know $e_i \geq 0$ since $\mathbf{L}$ is positive semi-definite. It can be seen that the problem (9) is equivalent to the following problem for a large enough value of ρ:

$$\begin{array}{l} \min\limits_{\mathbf{S},\alpha} \sum\limits_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} + \mu \sum\limits_{ij} \mathbf{c}^{w_{ij}} \left(\sum\limits_t \alpha^t d_{ij}^t\right) s_{ij} \\ +\beta \left\|\mathbf{S}\right\|_F^2 + \beta\gamma \left\|\alpha\right\|_2^2 + 2\rho \sum\limits_{i=1}^{K} e_i \\ s.t.,\ \mathbf{S} \geq 0, \mathbf{S}\mathbf{1} = \mathbf{1}, \alpha \geq 0, \alpha^T\mathbf{1} = 1 \end{array} \tag{10}$$

When ρ is set to a large enough value [1], $\sum_i^K e_i$ will be imposed to be close to 0, which results in $rank\left(\mathbf{L}\right) = N - K$.

According to the Ky Fan's Theorem [7], we have:

$$\sum_{i=1}^{K} e_i = \min_{\mathbf{F}\in\mathbb{R}^{N\times K}, \mathbf{F}^T\mathbf{F}=\mathbf{I}} tr\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right) \tag{11}$$

Therefore, the problem (10) is further equivalent to the following problem:

$$\begin{aligned} &\min_{\mathbf{S},\mathbf{F},\alpha} \sum_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} + \mu \sum_{ij} \mathbf{c}^{w_{ij}} \left(\sum_t \alpha^t d_{ij}^t\right) s_{ij} \\ &+\beta \left\|\mathbf{S}\right\|_F^2 + \beta\gamma \left\|\alpha\right\|_2^2 + 2\rho tr\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right) \\ &s.t., \left\{ \begin{array}{l} \mathbf{S} \geq 0, \mathbf{S1} = \mathbf{1}, \alpha \geq 0, \alpha^T\mathbf{1} = 1 \\ \mathbf{F} \in R^{N\times K}, \mathbf{F}^T\mathbf{F} = \mathbf{I} \end{array} \right. \end{aligned} \tag{12}$$

Compared with the original problem (9), (12) is much easier to solve. We propose an iterative method to minimize the above objective function (12).

Firstly, we initialize $\alpha^t = 1/T$ and then $\mathbf{S}$ by the optimal solution to the problem (7). Once these initial values are given, in each iteration, we first update $\mathbf{F}$ given $\mathbf{S}$ and α, and then update $\mathbf{S}$ and α by fixing the other parameters. These steps are described below:

Update F: By fixing $\mathbf{S}$ and α, the problem (12) is equivalent to optimizing the following objective function:

$$\min_{\mathbf{F}\in R^{N\times K}, \mathbf{F}^T\mathbf{F}=\mathbf{I}} tr\left(\mathbf{F}^T\mathbf{L}\mathbf{F}\right) \tag{13}$$

The optimal solution $\mathbf{F}$ to the problem (13) is formed by the K eigenvectors of $\mathbf{L}$ corresponding to the K smallest eigenvalues.

Update S: By fixing $\mathbf{F}$ and α, we can obtain $\mathbf{S}$ by optimizing (12). It is equivalent to optimize the following objective function:

$$\begin{aligned} &\min_{\mathbf{S}\geq 0, \mathbf{S1}=\mathbf{1}} \sum_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 s_{ij} + \rho \sum_{ij} \left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2^2 s_{ij} \\ &+\beta \left\|\mathbf{S}\right\|_F^2 + \mu \sum_{ij} \mathbf{c}^{w_{ij}} \left(\sum_t \alpha^t d_{ij}^t\right) s_{ij} \end{aligned} \tag{14}$$

It can be reformulated as:

$$\begin{aligned} &\min_{\mathbf{S}\geq 0, \mathbf{S1}=\mathbf{1}} \sum_i \left(\beta \mathbf{s}_i {\mathbf{s}_i}^T + \left(\mathbf{a}_i + \mu\mathbf{b}_i + \rho\mathbf{c}_i\right) {\mathbf{s}_i}^T\right) \\ &\Rightarrow \min_{\mathbf{S}\geq 0, \mathbf{S1}=\mathbf{1}} \sum_i \left(\mathbf{s}_i {\mathbf{s}_i}^T + \tfrac{\mathbf{a}_i + \mu\mathbf{b}_i + \rho\mathbf{c}_i}{\beta} {\mathbf{s}_i}^T\right) \end{aligned} \tag{15}$$

where $\mathbf{a}_i = \{a_{ij}, 1 \leq j \leq n\}$ with $a_{ij} = \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2$, $\mathbf{b}_i = \{b_{ij}, 1 \leq j \leq n\}$ with $b_{ij} = \sum_t \alpha^t d_{ij}^t$ and $\mathbf{c}_i = \{c_{ij}, 1 \leq j \leq n\} \in \mathbb{R}^{1\times n}$ with $c_{ij} = \left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2^2$. It is further equivalent to:

$$\min_{\mathbf{S}\geq 0, \mathbf{S1}=\mathbf{1}} \sum_i \left(\mathbf{s}_i + \tfrac{\mathbf{a}_i + \mu\mathbf{b}_i + \rho\mathbf{c}_i}{2\beta}\right)_2^2 \tag{16}$$

Then each $\mathbf{s}_i$ can be efficiently solved by using a quadratic programming solver, which will be introduced in the next subsection (solution for problem (16)).

Update α: By fixing $\mathbf{F}$ and $\mathbf{S}$, we can obtain α by optimizing (12). It is equivalent to optimize the following objective function:

$$\begin{aligned} &\min_{\alpha\geq 0, \alpha^T\mathbf{1}=1} \mu \sum_{ij} \mathbf{c}^{w_{ij}} \left(\sum_t \alpha^t d_{ij}^t\right) s_{ij} + \beta\gamma \left\|\alpha\right\|_2^2 \\ &\Rightarrow \min_{\alpha\geq 0, \alpha^T 1=1} \mu \sum_t \alpha^t \sum_{ij} d_{ij}^t s_{ij} / \left(\max^{w_{ij}} - \tau^{w_{ij}}\right) + \beta\gamma \left\|\alpha\right\|_2^2 \\ &\Rightarrow \min_{\alpha\geq 0, \alpha^T 1=1} \mu d\alpha + \beta\gamma \left\|\alpha\right\|_2^2 \end{aligned} \tag{17}$$

[1]In the real implementation, we initialize ρ with 1000, and increase ρ to $\rho \times 2$ if the current number of connected components is less than K, and decrease rho to $\rho/2$ if the current number of connected components is larger than K

where $d = \left\{d^t\right\}_{t=1}^T$, $d^t = \sum_{ij} d_{ij}^t s_{ij} / \left(\max^{w_{ij}} - \tau^{w_{ij}}\right)$ and $\max^{w_{ij}}$ is the max value of $\mathbf{S}$ with the topological structure w_{ij}. Then we can use a quadratic programming solver to obtain α.

We update $\mathbf{F}, \mathbf{S}$ and α iteratively until the objective function (7) converges, as shown in Algorithm 1.

Algorithm 1 Solution for JGLVS

```
Input: Initialized $\alpha$, segmentation number $K$, topology structure
       matrix $\mathbf{W}$, most-likely-linked sets $\mathbb{M}$, parameters $\beta$, $\gamma$, $\mu$, a
       large enough $\rho$;
Output: $\mathbf{S} \in \mathbb{R}^{N\times N}$ with exact $K$ connected components, $\alpha$;
1: Initialize $\mathbf{c}^z(x)$ using $\alpha$, $\mathbf{W}$ and $\mathbb{M}$;
2: Initialize $\mathbf{S}$ by the optimal solution of 7;
3: repeat
4:     Fix $\mathbf{S}$ and $\alpha$, calculate $\mathbf{F}$ according to the solution of prob-
       lem (13);
5:     Fix $\mathbf{F}$ and $\alpha$, update $\mathbf{S}$ by solving the problem (16);
6:     Fix $\mathbf{F}$ and $\mathbf{S}$, update $\alpha$ by solving the problem (17);
7:     Update $\mathbf{c}^z(x)$ using $\alpha$, $\mathbf{W}$ and $\mathbb{M}$;
8: until convergence or max iteration is reached.
9: return $\mathbf{S}$, $\alpha$;
```

3.4.1 Solution for problem (16)

In this subsection, we introduce an efficient solution for problem (16) for determining the regularization parameter β, so that we have fewer parameters to tune. The Lagrangian function of problem (16) is:

$$\begin{aligned} \ell\left(\mathbf{s}_i, \eta, \varepsilon_i\right) = \tfrac{1}{2} \sum_i \left\|\mathbf{s}_i + \tfrac{\mathbf{a}_i + \mu\mathbf{b}_i + \rho\mathbf{c}_i}{2\beta}\right\|_2^2 \\ -\eta(\mathbf{s}_i^T\mathbf{1} - 1) - \mathbf{s}_i^T \varepsilon_i \end{aligned} \tag{18}$$

where η, $\varepsilon_i \geq 0$ are the Lagrangian multipliers, and β is the regularization parameter for each $\mathbf{s}_i$. Let $d_{ij} = a_{ij} + \mu b_{ij} + \rho c_{ij}$. According to the KKT condition, it can be verified that the optimal solution $\mathbf{s}_i$ should be:

$$s_{ij} = \left(-\frac{a_{ij} + \mu b_{ij} + \rho c_{ij}}{2\beta} + \eta\right)_+ \tag{19}$$

By replacing η and ε_i according to the KKT condition, we obtain the optimal $\mathbf{s}_i$. However, in practice, we usually could achieve better performance if $\mathbf{s}_i$ is sparse, i.e., only the P nearest neighbors of $\mathbf{x}_i$ could have chance to connect to $\mathbf{x}_i$. Another benefit of learning a sparse similarity matrix $\mathbf{S}$ is that the computational burden can be largely alleviated for subsequent processing. With this motivation, we determine the parameter β.

Without loss of generality, suppose $d_{i1}, d_{i2}, ..., d_{iN}$ are ordered from small to large. If the optimal $\mathbf{s}_i$ has only P nonzero elements, then according to (19), we know $s_{iP} > 0$ and $s_{i,P+1} = 0$. Therefore, we have:

$$-\tfrac{d_{iP}}{2\beta_P} + \eta > 0, \; -\tfrac{d_{i,P+1}}{2\beta_{P+1}} + \eta \leq 0 \tag{20}$$

and

$$\begin{aligned} &\mathbf{s}_i^T 1 = \sum_{j=1}^{P} \left(-\tfrac{d_{ij}}{2\beta_i} + \eta\right) = 1 \\ &\Rightarrow \eta = \tfrac{1}{P} + \tfrac{1}{2P\beta_i} \sum_{j=1}^{P} d_{ij} \end{aligned} \tag{21}$$

By replacing η in (20) using (21), we have the following inequality for β_i:

$$\frac{P}{2}d_{ip} - \frac{1}{2}\sum_{j=1}^{P} d_{ij} < \beta_i \leq \frac{P}{2}d_{i,P+1} - \frac{1}{2}\sum_{j=1}^{P} d_{ij} \tag{22}$$

Therefore, in order to obtain an optimal solution $\mathbf{s}_i$ to the problem (16) that has exact P nonzero values, we set

$$\beta_i = \frac{P}{2}d_{i,P+1} - \frac{1}{2}\sum\nolimits_{j=1}^{P} d_{ij} \tag{23}$$

The overall β is set to the mean of $\beta_1, \beta_2, ..., \beta_n$. That is, we set β to be

$$\beta = \frac{1}{N}\sum\nolimits_{i=1}^{N}\left(\frac{P}{2}d_{i,P+1} - \frac{1}{2}\sum\nolimits_{j=1}^{P} d_{ij}\right) \tag{24}$$

The number of neighbors P is much easier to tune than the regularization parameter β since P is an integer and has explicit meaning.

3.5 Streaming video segmentation

An effective streaming algorithm can enable us to process an arbitrary long video with limited memory and computational resources, and thus is essential in video segmentation. We propose a simple yet effective clip-based segmentation method that scales well while maintaining temporal coherence, without processing the entire volume at once.

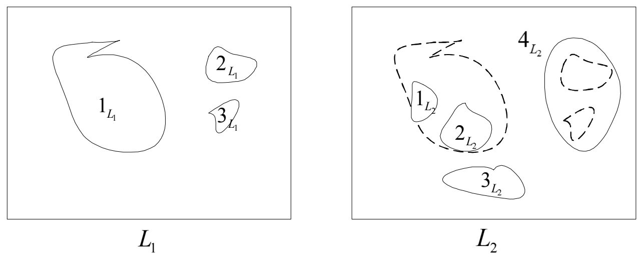


Figure 2: The segmentation labels L_1, L_2 of the overlapping frame f. L_1 denotes the segmentation of the overlapping in the previous clip, and provides some constraints to the segmentation of L_2, which is the segmentation in the current clip.

We start by partitioning the video into equally sized clips of n frames (n = 6 in our experiments), and one frame f is overlapped between neighboring clips. The temporal consistent constraints are introduced by properly propagating solutions from previous temporal window to the current window. Given the previous and current segmentation labels L_1, L_2 of the overlapping frame f, we first compute the similarity matrix O of different segments. The similarity of the i-th segment i_{L_1} in the previous segmentation and the j-th segment j_{L_2} in the current segmentation is defined as:

$$o(i_{L_1}, j_{L_2}) = |m_{i_{L_1}} \cap m_{j_{L_2}}|/min\left(|m_{i_{L_1}}|, |m_{j_{L_2}}|\right), \tag{25}$$

where $m_{i_{L_1}}$ and $m_{j_{L_2}}$ are masks to indicate which pixels belong to the segments i_{L_1} and j_{L_2}. After obtaining the similarity matrix O, we can assign new segmentation ids to the segments in L_2. Intuitively, two segments with the highest similarity should have the same segment id. However, there are three special cases to consider, which are illustrated in Fig. 2. The first case is that a segment (e.g., 3_{L_2}) has no overlapping in the previous segment. Then a new segment id should be assigned to 3_{L_2}. The second case is that two segments (e.g., 1_{L_2} and 2_{L_2}) are included by one previous segment (1_{L_1}). Then the one (2_{L_2}) with larger size will be assigned the id

Algorithm 2 The algorithm for generation of the temporal consistent constraints between previous and current segmentation labels L_1, L_2 of the overlapping frame f.

```
Input: Previous and current segmentation labels $L_1$ and $L_2$,
    threshold $threshold$;
Output: Refined segmentation labels $L'_2$, $mapping$;
 1: Get number of segments $num_1$ and $num_2$ in $L_1$ and $L_2$;
 2: Computer $O$ by Eq.(25);
 3: Assign $mapping$ the ids with the largest similarity to $L_2$ by
    $[value, mapping] = max(O)$;
 4: for $i = 1 : num_2$ do
 5:     if $value(i)$ is smaller than $threshold$, then
 6:         Updating $mapping(i)$ with a new id;
 7:     end if
 8:     if $mapping(i)$ is used by a previous segment, then
 9:         Assign a new id to the segment with a smaller size;
10:     end if
11:     if Another segment in $L_1$ has the same similarity to segment
        $i$ as $mapping(i)$, then
12:         Updating $mapping(i)$ with the id of a larger segment;
13:     end if
14:     Refine $L_2$ to $L'_2$ based on $mapping$;
15: end for
16: return $L'_2, mapping$;
```

(1) of overlapping segment (1_{L_1}), and the one (1_{L_2}) with smaller size will be assigned a new id. Lastly, if one segment (e.g., 4_{L_2}) includes two previous segments (2_{L_1} and 3_{L_1}), this segment will be assigned the id (2) of a larger segment (2_{L_1}). The algorithm for generating temporal consistent constraints is given in Algorithm 2. It takes previous and current segmentation labels L_1 and L_2, and similarity threshold $threshold$ as input. And it calculates which segment id in L_1 corresponds to each segment in L_2. Algorithm 2 outputs the refined segmentation labels L'_2 as well as the segment id mapping $mapping$.

4. EXPERIMENTS

In this section, we evaluate our JGLVS on the standard benchmark VSB100 [11]. First, we compare our method with other state-of-the-art methods. Then, we further analyze the effectiveness of our main components. Finally, we report the efficiency of our method.

4.1 Experimental Settings

We give the details of dataset selection, feature extraction and evaluation metrics in this subsection.

4.1.1 Dataset:

The selected VSB100 [11] is a very challenging dataset used for empirical evaluation. It is the largest video segmentation dataset with high definition frames, and consists of four difficult sub-datasets: general, motion segmentation, non-rigid motion segmentation and camera motion segmentation. Following the setting in [11, 20], we regard the general sub-dataset (60 video sequences) as our test set for all the approaches.

To make the comparison comprehensive, we set $\{\mu, \gamma\} = \{1000, 1\}$, $\{\rho\} = \{1000\}$ and $\{iteration\} = \{30\}$ in the experiment. β is automatically determined by the algorithm. In addition, the number of frames per window is set to be 6, and 1 frame is overlapped between neighboring windows.

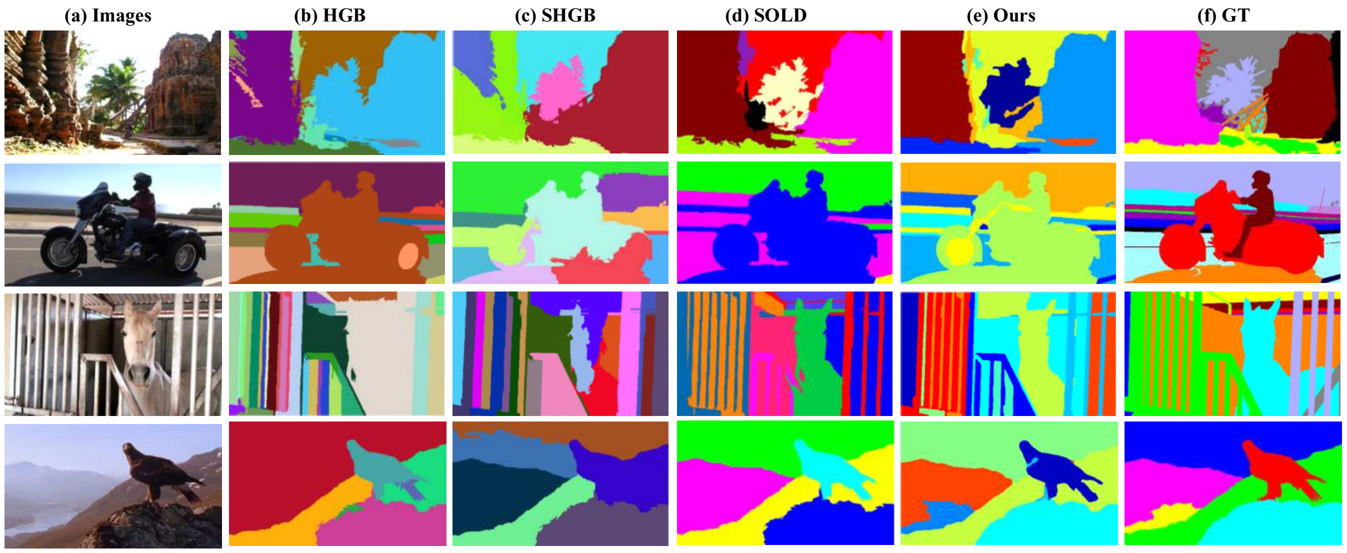


Figure 3: Qualitative comparisons with the state-of-the-art video segmentation methods HGB, SHGB and SOLD. We can see that our method substantially outperforms the algorithms of HGB, SHGB and SOLD.

Table 2: Comparison of state-of-the-art video segmentation algorithms with our proposed method on the test set of VSB100.

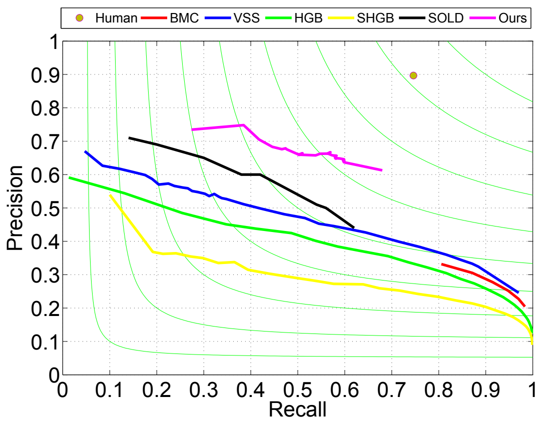
	BPR			VPR		
Algorithm	**ODS**	**OSS**	**AP**	**ODS**	**OSS**	**AP**
BMC [6]	0.47	0.48	0.32	0.51	0.52	0.38
VSS [9]	0.51	0.56	0.45	0.45	0.51	0.42
HGB [14]	0.47	0.54	0.41	0.52	0.55	**0.52**
SHGB[38]	0.38	0.46	0.32	0.45	0.48	0.44
SOLD [20]	0.54	0.58	0.40	0.53	0.60	0.46
Ours$_{-calibration}$	0.64	0.64	0.45	0.53	0.58	0.49
Ours$_{-consistency}$	0.65	0.65	**0.50**	0.51	0.53	0.47
Ours	**0.65**	**0.65**	0.48	**0.55**	**0.61**	0.51
Human	0.81	0.81	0.67	0.83	0.83	0.70

4.1.2 Features and Distances:

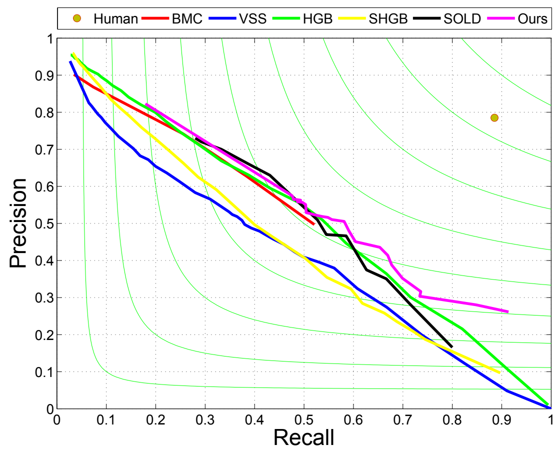
Common boundary strength [*cbs*]. This measures distance in the close vicinity of the common boundary between two superpixels i_f and j_f by averaging the common boundary strength. We take $\overline{\mathbf{v}}_f^{ij}$ the average UCM of [2] as a measure of the boundary strength between i and j and define: $cbs(i_f, j_f) = \overline{\mathbf{v}}_f^{ij}$.
Lab [*lab*]. This uses the distance between the median brightness and color of a superpixel in Lab-color-space as a measure of the overall distance among two superpixels i and j, from the same or different frames f and f': $lab(i_f, j_{f'}) = \|\overline{LAB}_{i_f} - \overline{LAB}_{j_{f'}}\|_2$.
Boundary optical flow [*bof*]. We consider an optical flow estimation [9]. The resulting $u_f(x)$ allows to compute the motion distance in the vicinity of the boundary between two superpixels by averaging their u_f across the common boundary φ_f^{ij}: $bof(i_f, j_f) = \left(\sum_{(x_i^m, x_j^m)\in\varphi_f^{ij}} \|\overline{u}^f\left(x_i^m\right) - \overline{u}^f\left(x_j^m\right)\|_2\right) / |\varphi_f^{ij}|$.
Superpixel optical flow [*sof*]. This measures the overall motion distance between two superpixels i_f and $j_{f'}$ based on their median optical flow u: $sof(i_f, j_{f'}) = \|\overline{u}_{i_f} - \overline{u}_{j_{f'}}\|_2$.
Superpixel shape distance [*ssd*]. We measure the shape distance by comparing $m_{j_{f'}}$ the shape of a superpixel j at frame f' with the shape of i_f propagated with optical flow to frame f' (its projected mask $m_{i_f}^{f'}$). *ssd* is given by the Dice coefficient between the true $m_{j_{f'}}$ and optical-flow-projected $m_{i_f}^{f'}$ binary mask: $ssd(i_f, j_{f'}) = 1 - 2|m_{i_f}^{f'} \cap m_{j_{f'}}| / \left(|m_{i_f}^{f'}| + |m_{j_{f'}}|\right)$.
Superpixel trajectories intersection [*sti*]. It measures the distance between superpixels i_f and $j_{f'}$ which belongs to frames potentially further in time from each other $f' = f + m, m > 2$. We consider the dense point trajectories of [36] as a measure of the shape (binary mask) projection. Let $\phi(i_f)$ be the subset of trajectories intersecting superpixel i_f. The distance is the Dice measure between the intersection sets $\phi(i_f)$ and $\phi(j'_f)$: $sti(i_f, j_{f'}) = 1 - 2|\phi(i_f) \cap \phi(j'_f)| / \left(|\phi(i_f)| + |\phi(j'_f)|\right)$.

4.1.3 Evaluation Metrics:

Following [11, 20], we use two evaluation metrics: 1) Boundary Precision-Recall (BPR), which casts the boundary detection problem as one of classifying boundary from nonboundary pixels and measures the quality of a segmentation boundary map in the precision-recall framework; and 2) Volume Precision-Recall (VPR), which optimally assigns spatio-temporal volumes between the computer generated segmentation and the human annotated segmentations and then measures their overlap. For both BPR and VPR, we report average precision (AP), optimal dataset scale (ODS), and optimal segmentation scale (OSS).



(a) Bourndary Global PR Curve



(b) Volume Global PR Curve

Figure 4: Comparison curves of our framework with the state-of-the-art video segmentation approaches BMC [6], VSS [9], HGB [14], SHGB [38] and SOLD [20].

4.2 Comparison with state-of-the-art video segmentation methods

We compare our approach with the following five state-of-the-art video segmentation algorithms: BMC [6], VSS [9], HGB [14], SHGB [38] and SOLD [20]. We also report our method without calibration (Ours$_{-calibration}$) and our method without temporal consistency (Ours$_{-consistency}$). Table 2 illustrates a summary of the aggregate evaluation performance, including ODS, OSS and AP of both BPR and VPR. Fig.4 shows the BPR and VPR curves of the comparisons on the VSB100 dataset. From Table 2 and Fig.4, we have the following observations:

- Our approach outperforms the state-of-the-art methods (BMC, VSS, HGB, SHGB and SOLD) in both BPR and VPR on the VSB100 dataset. Specifically, our proposed method outperforms the currently best performance (SOLD [20]) on both BPR and VPR by a large margin, as it appears both in the Table 2 and Fig.4 (11%, 7% and 8% in BPR, 2%, 1% and 5% in VPR). Our AP in VPR is slightly lower than HGB. But we can alleviate it by simply increasing the superpixels number, as shown in Table 3.

- Though VSS [9] and SOLD [20] exploits multiple cues as well, our method performs better. This probably owes to the proposed joint graph learning and video segmentation framework, and the automatically learned weights for different cues.

- SOLD [20] is a strong competitor. The superior performances over SOLD in both BPR and VPR demonstrate that our approach can not only effectively infer the spatial similarity between superpixels within a frame, but also preserve the longer-range temporal consistency in a streaming mode.

- Temporal consistency processing plays an important role for VPR, as indicated in Table 2. An example is given in Fig.5 to illustrate the effect of temporal consistency processing. If we do not constrain that the same object in the close frames to have the same label, the performance on VPR metric will decrease.

- Topology calibration improves the performance, especially for VPR. This is due to that without calibration, the distances of different topological structures (especially for cross frame) are not comparable.

We illustrate qualitative results in Fig.3, comparing our proposed method to the state-of-the-art video segmentation algorithms including HGB, SHGB and SOLD. Fig.3 shows consistent results to the quantitative results. Our method is able to provide better distinguished visual objects with well-localized boundaries and limited label leakage.

4.3 Component analysis

In this subsection, we study the effect of *level* and the different types of *distances* on our proposed method.

As described in Section 3, our proposed algorithm is imposed on the superpixels which are extracted from [2]. The number of superpixels is determined by the value of *level*. From the Table 3, we can see that the *level* is important to the performance. In general, when $level = 75$, the overall best performance is achieved for both BPR (65%, 65% and 48%) and VPR (55%, 61% and 51%). In addition, when $level = 25$, the AP for both BPR and VPR reaches the peak values: 54% and 56% respectively. This is due to that when $level = 25$, more superpixels are generated and over-segmentation improves the precision but decreases the recall [2].

The effect of different distances is analyzed and the results are shown in Table 4. As described in Section 3, our method uses different types of pairwise distance between superpxiels for video segmentation. From Table 4, we can see that *distances* have different impact on the performance, hence it is important to learn the weights for different distances. Using a combined distances with learned weight achieves better performance than using a single distance.

Table 3: The effect of *level* on our proposed method.

	BPR			VPR		
Algorithm	**ODS**	**OSS**	**AP**	**ODS**	**OSS**	**AP**
$Level = 25$	0.59	0.59	**0.54**	0.54	0.58	**0.56**
$Level = 50$	0.64	0.65	0.53	**0.55**	0.57	0.51
$Level = 75$	**0.65**	**0.65**	0.48	**0.55**	**0.61**	0.51
$Level = 95$	0.64	0.64	0.47	**0.55**	0.59	0.46
$Level = 125$	0.61	0.61	0.41	0.53	0.58	0.40
Human	0.81	0.81	0.67	0.83	0.83	0.70

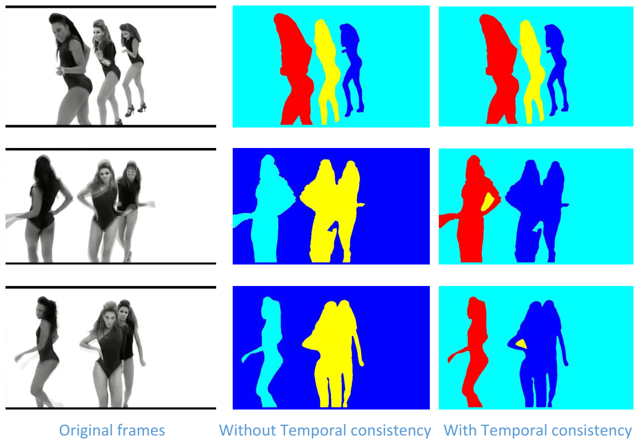


Figure 5: Segmentations of a given sequence of video frames. The left are the original frames, the middle lines are the segmentations without temporal consistency, and the right lines are the segmentations with temporal consistency processing. It is clear that if no temporal consistency processing is applied, the same object in the successive frames cannot be guaranteed to have the same segment id (i.e., label). In this case, the VPR will degrade.

Table 4: The effect of *distances* on our proposed method.

	BPR			VPR		
Algorithm	**ODS**	**OSS**	**AP**	**ODS**	**OSS**	**AP**
CBS	0.64	0.64	0.46	0.43	0.46	0.38
BOF	0.64	0.64	0.44	0.43	0.44	0.37
LAB	0.64	0.65	0.45	0.51	0.53	0.43
SOF	0.64	0.64	0.47	0.42	0.45	0.37
SSD	0.64	0.64	0.44	0.51	0.56	0.45
STI	0.64	0.64	0.45	0.46	0.49	0.42
All	0.65	0.65	0.48	0.55	0.61	0.51
Human	0.81	0.81	0.67	0.83	0.83	0.70

4.4 Efficiency Analysis

In this subsection, we evaluate the time efficiency of our proposed method. These experiments are carried out on a desktop with an Intel(R) Core (TM)2 Duo CPU and 8GB RAM. Our method is conducted on the features of the generated superpixels, which are obtained in the preprocessing step. The computational cost for our algorithm is $O(N^2) + O(N^3)$. However, we do segmentation on the N superpixels, which is usually 10-100s. Therefore, our method can segment 1 frame within 0.2s on average, as indicated in Fig. 6. Fig. 6 reports the running time corresponding to the number of segmentations. In average, it costs 0.1895 seconds per frame for our proposed method. Large number of segmentations does not necessarily consume more time, because the optimization may terminate in less iterations.

5. CONCLUSIONS

This paper proposes the JGLVS framework that simultaneously learns the graph similarity matrix and video segmentation, instead of first organizing the superpixels into graphs and then cutting the generated graph for segmentation. Based on the spatial and visual information, each vertex is assigned with adaptive and optimal neighbors for graph similarity learning. By imposing rank constraints on the Laplacian matrix, the number of connected components in the generated similarity graph are equal to the number of segmentations, sidestepping the need for separate steps in similarity computation and graph cutting. Experimental results show that the proposed unsupervised JGLVS outperforms state-of-the-art video segmentation algorithms by a large margin on the VSB100 dataset.

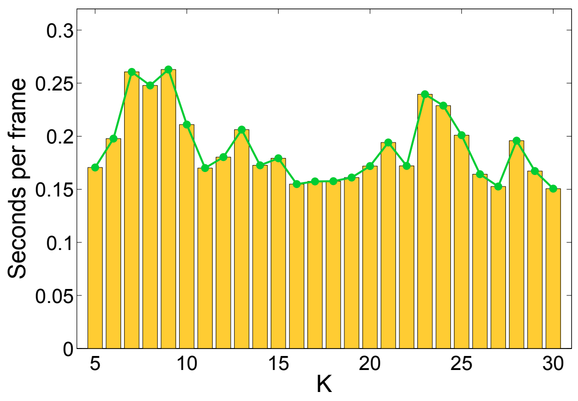


Figure 6: The running time (seconds per frame) *vs.* number of segmentations (K) of our proposed method.

6. ACKNOWLEDGMENTS

The work of Jingkuan Song and Nicu Sebe has been partially supported by the FP7 EC project xLiMe. The work of Lianli Gao is partially supported by the Fundamental Research Funds for the Central Universities (Grant No.ZYGX2014J063) and the National Natural Science Foundation of China (Grant No. 61502080).

7. REFERENCES

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, pages 2294–2301, 2009.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

[3] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *ICCV*, pages 833–840, 2009.

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, pages 282–295, 2010.

[5] L. Chen, J. Shen, W. Wang, and B. Ni. Video object segmentation via dense trajectories. *IEEE Trans. Multimedia*, 17(12):2225–2234, 2015.

[6] J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629–640, 2008.

[7] K. Fan. On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations. I. *Proceedings of the National Academy of Science*, 35:652–655, Nov. 1949.

[8] K. Fragkiadaki and J. Shi. Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In *CVPR*, pages 2073–2080, 2011.

[9] F. Galasso, R. Cipolla, and B. Schiele. Video segmentation with superpixels. In *ACCV*, 2012.

[10] F. Galasso, M. Keuper, T. Brox, and B. Schiele. Spectral graph reduction for efficient image and streaming video segmentation. In *CVPR*, 2014.

[11] F. Galasso, N. S. Nagaraja, T. J. Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013.

[12] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. T. Shen. Optimal graph learning with partial tags and multiple features for image and video annotation. In *CVPR*, pages 4371–4379, 2015.

[13] L. Gao, J. Song, F. Nie, F. Zou, N. Sebe, and H. T. Shen. Graph-without-cut: An ideal graph learning for image segmentation. In *AAAI*, pages 1188–1194, 2016.

[14] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, pages 2141–2148, 2010.

[15] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *ICCV*, pages 1865–1872, 2013.

[16] H. Jiang, G. Zhang, H. Wang, and H. Bao. Spatio-temporal video segmentation of static scenes and its applications. *IEEE Trans. Multimedia*, 17(1):3–15, 2015.

[17] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, 2015.

[18] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, pages 3271–3279, 2015.

[19] A. Khoreva, F. Galasso, M. Hein, and B. Schiele. Classifier based graph construction for video segmentation. In *CVPR*, 2015.

[20] C. Li, L. Lin, W. Zuo, S. Yan, and J. Tang. Sold: Sub-optimal low-rank decomposition for efficient video segmentation. In *CVPR*, 2015.

[21] B. Liu and X. He. Multiclass semantic video segmentation with object-level active inference. In *CVPR*, pages 4286–4294, 2015.

[22] B. Luo, H. Li, T. Song, and C. Huang. Object segmentation from long video sequences. In *ACM Multimedia*, pages 1187–1190, 2015.

[23] T. Ma and L. J. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, pages 670–677, 2012.

[24] N. S. Nagaraja, F. R. Schmidt, and T. Brox. Video segmentation with just a few strokes. In *ICCV*, pages 3235–3243, 2015.

[25] F. Nie, X. Wang, and H. Huang. Clustering and projected clustering with adaptive neighbors. In *SIGKDD*, pages 977–986, 2014.

[26] F. Nie, X. Wang, M. I. Jordan, and H. Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI*, pages 1969–1976, 2016.

[27] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, pages 1583–1590, 2011.

[28] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, pages 614–621, 2012.

[29] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1187–1200, 2014.

[30] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*, pages 460–473, 2008.

[31] S. H. Raza, M. Grundmann, and I. A. Essa. Geometric context from videos. In *CVPR*, pages 3081–3088, 2013.

[32] A. V. Reina, S. Avidan, H. Pfister, and E. L. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, pages 268–281, 2010.

[33] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen. Hashing on nonlinear manifolds. *IEEE Trans. Image Processing*, 24(6):1839–1851, 2015.

[34] J. Son, I. Jung, K. Park, and B. Han. Tracking-by-segmentation with online gradient boosting decision tree. In *ICCV*, 2015.

[35] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Trans. Multimedia*, 15(8):1997–2008, 2013.

[36] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[37] Y. Wang, J. Liu, Y. Li, and H. Lu. Semi- and weakly-supervised semantic segmentation with deep convolutional neural networks. In *ACM Multimedia*, pages 1223–1226, 2015.

[38] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, pages 626–639, 2012.

[39] X. Yao, J. Han, G. Cheng, and L. Guo. Semantic segmentation based on stacked discriminative autoencoders and context-constrained weakly supervised learning. In *ACM Multimedia*, pages 1211–1214, 2015.

[40] S. Yi and V. Pavlovic. Multi-cue structure preserving MRF for unconstrained video segmentation. In *ICCV*, 2015.

[41] C.-P. Yu, H. Le, G. Zelinsky, and D. Samaras. Efficient video segmentation using parametric graph partitioning. In *ICCV*, 2015.

[42] V. Zografos, R. Lenz, E. Ringaby, M. Felsberg, and K. Nordberg. Fast segmentation of sparse 3d point trajectories using group theoretical invariants. In *ACCV*, pages 675–691, 2014.