Classifier Based Graph Construction for Video Segmentation

Anna Khoreva[1], Fabio Galasso[2], Matthias Hein[3], Bernt Schiele[1]
[1] Max Planck Institute for Informatics, Germany
[2] OSRAM Corporate Technology, Germany
[3] Saarland University, Germany

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, consist of three essential components: 1. powerful features account for object appearance and motion similarities; 2. spatio-temporal neighborhoods of pixels or superpixels (the graph edges) are modeled using a combination of those features; 3. video segmentation is formulated as a graph partitioning problem. While a wide variety of features have been explored and various graph partition algorithms have been proposed, there is surprisingly little research on how to construct a graph to obtain the best video segmentation performance. This is the focus of our paper. We propose to combine features by means of a classifier, use calibrated classifier outputs as edge weights and define the graph topology by edge selection. By learning the graph (without changes to the graph partitioning method), we improve the results of the best performing video segmentation algorithm by 6% on the challenging VSB100 benchmark, while reducing its runtime by 55%, as the learnt graph is much sparser.

1. Introduction

Video segmentation recently witnesses growing interest [4, 5, 6, 9, 17, 26, 33, 35, 38, 39, 44, 49, 55, 59]. On the one hand, this is motivated by its usefulness for applications such as semantic scene understanding [26], activity recognition [50], or geometric context classification [41]. In these cases, organizing a video into spatio-temporal tubes allows the joint consideration of appearance and motion, while reducing the search space for the solution. On the other hand, video segmentation poses interesting research questions. In addition to the scene and scale ambiguities of image segmentation [3, 16, 25, 42], various parts of the scene will change over time as well as appear or disappear.

Graph-based approaches are among the top-performing methods for video segmentation [21, 17, 32, 7, 38, 44, 56, 19]. The use of graphs is long established in segmentation [45, 3, 25, 11, 21, 35, 7, 38, 47]. Graphs provide a natural representation of image/video sequences, where edges encode the spatio-temporal structure, and allow long-term reasoning due to their transitivity property. Graph-based video segmentation techniques: 1. compute features among pairs of pixels or superpixels; 2. design a graph according to the spatio-temporal neighborhood of the pixels or superpixels and manually combine features to weight its edges; 3. partition the graphs with spatio-temporal clustering.

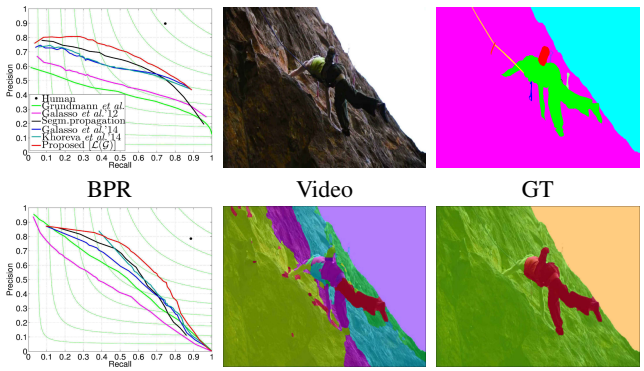


Figure 1: Climbing up! We consider state-of-the-art features for the computation of superpixel similarities and the graph partitioning theory of recent work [19]. We contribute theory and best-practices for *graph construction* and set a new state-of-the-art performance on the challenging VSB100 [20] (BPR and VPR reported here, more details in Section 4.)

Previous work has used a variety of features and has proposed various graph partitioning algorithms. However, we argue in this paper that constructing the underlying graph is a crucial step for best performance of such graph-based methods that has received little attention in the literature. This paper therefore explicitly addresses the problem of graph construction. We propose and empirically evaluate procedures and validation-based best practices to learn both the edge topology and weights. Our contribution includes 1. using a classifier for learning the pairwise similarities between superpixels, leveraging the recent availability of

a larger training set for video segmentation [20]; 2. using different classifiers for differently-neighboring superpixels (within the same frame or across time) and further considering the neighboring topology (superpixels directly neighboring or connected by longer-term links); 3. calibrating the confidence of the various classifiers with their classification accuracy; 4. selecting edges based on the classifier confidence which, while further improving the quality, also reduces the graph size and thus the computational load. These topics are respectively treated in Section 3.

In Section 2 we present the features and the graph partitioning model we use. The proposed approach based on learning allows the seamless integration of multiple features from recent literature [7, 3, 38, 19]. We build upon the graph partitioning model of [19] based on spectral clustering and show that addressing the graph construction *explicitly* helps to achieve better performance (cf. Fig. 1) without altering the graph partitioning or the underlying features.

Related Work: Meaningful features are necessary for good video segmentation. Much literature [7, 21, 38, 18] has proposed features for appearance, motion or shape similarities among the graph nodes. Most works are currently limited in the number of features they can leverage, as often the researchers hand-design the feature combination to measure similarity between pixels or superpixels. In this work we learn classifiers to combine features and seamlessly integrate them.

Much research has been devoted to graph partitioning models [12, 56, 35, 2, 10, 26, 19]. While measurable differences have been observed we intentionally focus on the graph construction problem instead. Therefore, we adopt the recent and successful graph partitioning model [19], which is based on spectral clustering [36, 45, 7, 17, 47]. However, our proposed graph construction is directly applicable to other graph-based techniques (see Sec. 4).

Constructing the graph is a vital step for ensuring the performance of clustering methods [34, 27] Although graph-based methods have been extensively studied, there have been limited efforts for building effective graphs. The most popular method for constructing a sparse graph is the nearest neighbor (NN) approach, including different variants such as k-nearest neighbor and ϵ-nearest neighbor methods. Another contender approach is the b-matching procedure [28], which prunes graph edges such that the degree of each node is b, producing a more balanced variant of k-nearest neighbor. Several works explored semi-supervised learning of the graph [27, 1], i.e. learning the graph from its partial labelling. By contrast our method is applied to unlabeled test-set videos.

To the best of our knowledge, graph construction based on classifier-learnt combination of features is novel in video segmentation. While learning the edge weights of the graph has been exploited in image segmentation [43, 51, 30], our work addresses the topology of the graph, raising novel issues, such as weight-calibration and edge-selection, which we discuss in Section 3. Learning the topology provides larger performance gains and benefits efficiency due to a sparser structure of the constructed graph.

2. Graph-based video segmentation

Let us represent a video sequence as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Nodes $i \in \mathcal{V}$ are superpixels, extracted at each frame from a specific hierarchical level of an image segmentation algorithm [3]. Following [19], we do not consider the lowest level (finest superpixels), but rather extract them by thresholding the ultrametric contour map (*ucm*) at a higher value (0.12), whereby fewer of them (*larger superpixels*) provide a comparable model error.

Edges of the graph $e_{ij} \in \mathcal{E}$ connect pairs of superpixels i and j with non-negative weight w_{ij}, which expresses their similarity. Following [21, 32, 18, 38], edges may connect neighbors:

within frame: i and j are neighbors if they share a common part of their superpixel 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;

across $>$ 2 frames: linked if overlapping with common long-term point trajectories.

Graph based video segmentation proceeds in three main steps:

1. Feature computation. Depending on the edge type, a number of features are available to compute the similarity between superpixels. For example, superpixels on the same frame may be related by the strength of the image segmentation boundary between them (*aba*) and by the χ^2 distance between their color histograms (sta_{χ^2}); if neighboring across frames, just sta_{χ^2} applies (see Sec. 3.1 and Tab. 1 for more details).

2. Graph construction. State-of-the-art approaches use edges e_{ij} if the two superpixels are neighbors, either within or across frames. Then, they compute edge weights w_{ij} by combining the similarities from the applicable features linearly. Current video segmentation literature [7, 21, 18, 38] sets the combinations manually on a validation set.

3. Graph partitioning. Video segmentation S is defined as a partition $S = \{S_1, S_2, \ldots, S_K\}$ of the vertex set $\mathcal{V}$, i.e. $\cup_k S_k = \mathcal{V}$, $S_k \cap S_m = \varnothing \;\; \forall \;\; k \neq m$. Given $\mathcal{S}$ the set of all partitions, graph partitioning looks for the optimal video segmentation $S^* = \{S_1^*, S_2^*, \ldots, S_N^*\} \in \mathcal{S}$ (where N is the number of visual objects) which minimizes an objective function, implicitly [21, 57, 40] or explicitly [45, 36, 53, 9].

Different tot state-of-the-art literature, our work focuses on the *graph construction*. Since we use discriminatively

trained classifiers to combine features, we name ours a *learnt graph* $\mathcal{L}(\mathcal{G})$. Furthermore, we investigate graph topology, classifier output confidence mapping and edge selection in detail in Section 3.

In the rest of this Section, we present the *features* which we use, from state-of-the-art video segmentation, and the *graph partitioning* model which we adopt, based on spectral clustering and the graph-equivalent reweighting from [19]. We use the publicly available code of [19] for the original graph construction and partitioning method.

2.1. Superpixel features

Adopting learning allows to seamlessly integrate an arbitrary number of features into the computation of the graph edge weights, *letting the classifier work out the optimal combination*. We consider 14 features from state-of-the-art video segmentation [24, 7, 3, 18, 38, 19], which apply to superpixels. We present them by grouping appearance, motion and shape features.

Appearance Based Features

Across boundary appearance [aba]. This measures similarity in the close vicinity of the common boundary between two superpixels i_f and j_f by averaging the common boundary strength (here and in the following we explicitly indicate the frame f which the superpixel belongs to for clarity). We take $\overline{v}_f^{ij}$ the average ultrametric contour map of [3] as a measure of the boundary strength between i and j and define: $aba(i_f, j_f) = \overline{v}_f^{ij}$.

Spatio-temporal appearance [sta, sta_{χ^2}]. This uses the distance between the median brightness and color of a superpixel in *Lab*-color-space as a measure of the overall similarity among two superpixels i and j, from the same or different frames f and f': $sta(i_f, j_{f'}) = \exp\{-\lambda_{sta}\|\overline{Lab}_{i_f} - \overline{Lab}_{j_{f'}}\|\}$.

Similarly sta_{χ^2} measures the overall appearance similarity using *Lab* (8-bin) color histograms and their χ^2 distance: $sta_{\chi^2}(i_f, j_{f'}) = \exp\{-\lambda_{sta_{\chi^2}} d_{\chi^2}(h(Lab_{i_f}), h(Lab_{j_f}))\}$.

Texture [$text$, $text_{\chi^2}$]. Texture information may be encoded (cf. [24] for more details) with (a subset of) the textons designed by Leung and Malik [31]. We consider the L_2 distance between the mean absolute filter responses $text(i_f, j_{f'}) = \exp\{-\lambda_{text}\|\overline{T}_{i_f} - \overline{T}_{j_{f'}}\|\}$ and the chi-squared distance between the histograms of maximum filter responses $text_{\chi^2}(i_f, j_{f'}) = \exp\{-\lambda_{text_{\chi^2}} d_{\chi^2}(h(T_{i_f}), h(T_{j_f}))\}$.

Size ratio [$size$]. We further consider the relative size difference of superpixels as an indication of appearance similarity $size(i_f, j_{f'}) = \big||i_f| - |j_{f'}|\big| / \max\{|i_f|, |j_{f'}|\}$.

Motion Based Features

Across boundary motion [abm]. We consider an optical flow estimate [58], which we smooth spatially (preserving the across-superpixels boundaries with bilateral filtering) and temporally (median filtered ± 2 frames). The resulting $\overline{u}^f(x)$ (simply indicated as $\overline{u}^f$ in the following) allows to compute the motion similarity in the vicinity of the boundary between two superpixels by averaging their $\overline{u}^f$ distance across the common boundary ψ_f^{ij}: $abm(i_f, j_f) = \exp\{-\lambda_{abm}\big(\sum_{(x_i^m, x_j^m)\in\psi_f^{ij}} \|\overline{u}^f(x_i^m) - \overline{u}^f(x_j^m)\|^2\big)/|\psi_f^{ij}|\}$.

Spatio-temporal motion [stm, stm_{χ^2}]. This measures the overall motion similarity between two superpixels i_f and $j_{f'}$ based on their median optical flow $\overline{u}$: $stm(i_f, j_{f'}) = \exp\{-\lambda_{stm}\|\overline{u}_{i_f} - \overline{u}_{j_{f'}}\|^2\}$.

Similarly, we may compute the similarity with the χ^2 distance between the superpixel optical flow (22 bin) histograms: $stm_{\chi^2}(i_f, j_{f'}) = \exp\{-\lambda_{stm_{\chi^2}} d_{\chi^2}\big(h(u_{i_f}), h(u_{j_{f'}})\big)\}$.

Spatial distance [sd]. As a measure of motion-displacement, we additionally consider the spatial distance between centroids of superpixels c_{i_f} and $c_{j_{f'}}$ across frames: $sd(i_f, j_{f'}) = \|c_{i_f} - c_{j_{f'}}\|$.

Shape Based Features

Short term temporal [stt]. We measure the shape similarity by comparing $m_{j_{f'}}$ the *shape* (its binary mask m) of a superpixel j at frame f' with the shape of i_f propagated with optical flow to frame f' (its projected mask $m_{i_f}^{f'}$). stt is given by the Dice coefficient between the true $m_{j_{f'}}$ and optical-flow-projected $m_{i_f}^{f'}$ binary mask: $stt(i_f, j_{f'}) = 2\big|m_{i_f}^{f'} \bigcap m_{j_{f'}}\big| / \big(\big|m_{i_f}^{f'}\big| + |m_{j_{f'}}|\big)$.

Long term temporal [ltt, cit, td]. In a similar spirit to stt, ltt measures the similarity between superpixels i_f and $j_{f'}$ which belong to frames potentially further in time from each other ($f' = f + m$, $m \in (0, F]$ where F scales up to the whole length of the video sequence). We consider the dense point trajectories of [46] as a measure of the shape (binary mask) projection. Let Φ_{i_f} be the subset of trajectories intersecting superpixel i_f. The similarity is the Dice measure between the intersection sets Φ_{i_f} and $\Phi_{j_{f'}}$: $ltt(i_f, j_{f'}) = 2\big|\Phi_{i_f} \bigcap \Phi_{j_{f'}}\big| / \big(\big|\Phi_{i_f}\big| + \big|\Phi_{j_{f'}}\big|\big)$.

We additionally provide the classifier with the number of common intersecting trajectories (the fewer dense tracks are available, the less it should rely on ltt as a reliable shape similarity): $cit(i_f, j_{f'}) = \big|\Phi_{i_f} \bigcap \Phi_{j_{f'}}\big|$ and temporal distance between superpixels $td(i_f, j_{f'}) = |f - f'|$.

2.2. Graph partitioning

We consider the graph partitioning model of [19], currently performing best on VSB100 [20]. The approach seeks to determine the graph partition $S = \{S_1, S_2, \ldots, S_N\}$ (complete and disjoint $\cup_k S_k = \mathcal{V}$, $S_k \cap S_m = \varnothing \quad \forall \quad k \neq m$) which is optimal according to the

normalized cut (NCut) objective:

$$\mathrm{NCut}(S_1,\dots,S_N)=\sum_{k=1}^{N}\frac{\mathrm{cut}(S_k,\mathcal{V}\backslash S_k)}{\mathrm{vol}(S_k)}, \qquad (1)$$

where $\mathrm{cut}(S_k,\mathcal{V}\backslash S_k)=\sum_{i\in S_k,j\in\mathcal{V}\backslash S_k} w_{ij}$ and $\mathrm{vol}(S_k)=\sum_{i\in S_k,j\in\mathcal{V}} w_{ij}$.

Following established literature [45, 36, 54, 11, 7, 3, 48, 47, 14, 18, 17, 35, 19], we consider the spectral relaxation of the NCut problem (otherwise NP-Hard):

$$\min_{T}\mathrm{Tr}(T'L_{\mathrm{sym}}T) \quad \text{subject to} \quad TT'=I,\ T=D^{\frac{1}{2}}H, \qquad (2)$$

where H is the matrix containing indicator vectors h_i, $L_{\mathrm{sym}}=I-D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ is the normalized graph Laplacian, W is the matrix containing the pairwise affinities w_{ij} and D is the diagonal degree matrix with $d_{ii}=\sum_{j\in\mathcal{V}} w_{ij}$. The solution of (2) is provided by matrix T which contains the first k eigenvectors L_{sym} as columns.

As theoretically and empirically relevant to good performance, we reweight the affinities w_{ij}, as [19] suggests, by the number of fine superpixels to w^{Q}_{IJ} (cf. [19] for more details):

$$w^{Q}_{IJ}=\begin{cases}\sum_{i\in I}\sum_{j\in J} w_{ij} & \text{if } I\neq J,\\ \frac{1}{|I|}\sum_{i\in I}\sum_{j\in J} w_{ij}-\frac{(|I|-1)}{|I|}\sum_{i\in I}\sum_{j\in\mathcal{V}\backslash I} w_{ij} & \text{if } I=J.\end{cases} \qquad (3)$$

2.3. VSB100: Learning, Validating and Testing

[20] has recently introduced VSB100: a challenging video segmentation benchmark based on the HD quality videos from [48], the boundary precision-recall (BPR) metric from [3] and a volume precision-recall metric (VPR) that reflects the properties of a good video segmentation, such as temporal consistency. Besides the PR curves, we report aggregate performance for BPR and VPR: optimal dataset scale [ODS], optimal segmentation scale [OSS], average precision [AP]. (We additionally report the length and number of clusters (NCL) statistics.)

The 100 videos are arranged into train (40) and test (60) set. We further split the training set into a *training* and *validation* sets, where 24 video sequences are used for learning the classifier and 16 videos are used for validation of the parameters. We compare with state-of-the-art on the whole test set.

3. Graph construction

Here we discuss the proposed graph construction $\mathcal{L}(\mathcal{G})$. First we consider learning for estimating the edge weights and the importance of topology in the setup of different classifiers. Then we consider calibration of classifier scores based on their reliability. Finally, we discuss edge selection, for a sparse efficient graph. We conduct these experiments on the training and validation sets.

3.1. Learning superpixel affinities

Let us consider the graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$, as introduced in Section 2, composed of superpixel nodes, connected over their spatio-temporal neighborhoods.

We propose the use of a classifier to learn the edge weights. To this end, we harvest from the training set pairs of superpixels connected by an affinity and provide them to a classifier along with their ground truth labelling (the indication whether two superpixels belong to the same video segment or not). Random Forest is used for learning.

There are four superpixel edge types: within, across 1, across 2 and across >2 frames. While a single classifier should suffice for all, in our first experiments it turned out that its performance is extremely poor. By contrast, the use of multiple classifiers is beneficial. We attribute this to data unbalance (the edges within frames are the vast majority) and to scarcity of training samples (esp. compared to the large image and video variability).

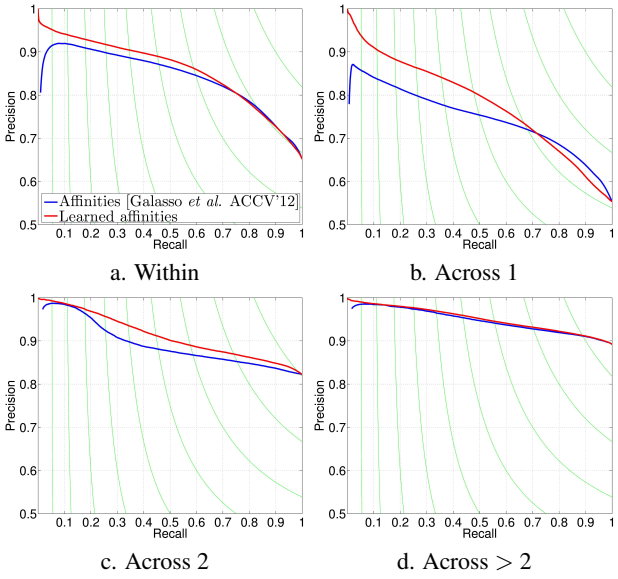
We set therefore to consider four classifiers for the four edge types. The corresponding available features are:
within: sta, sta_{χ^2}, stm, stm_{χ^2}, aba, abm, sd, $text$, $text_{\chi^2}$, $size$;
across 1: sta, sta_{χ^2}, stm, stm_{χ^2}, stt, sd, $text$, $text_{\chi^2}$, $size$;
across 2: ltt, cit, stt, sd, $text$, $text_{\chi^2}$, $size$;
across >2: ltt, cit, sd, td, $size$.

In our experience the Random Forest classifier profits from removing redundant or irrelevant features. Therefore for each affinity type we validate the subset of features to improve the model. The maximum set which we consider consists of 10 features (within frame), therefore we can test each possible combination finding the one which maximizes the average precision of the classifier. This is an exhaustive search of the feature space, however in this particular setting it is computationally tractable as the feature set is quite small. We train a new classifier for each subset of features and validate the performance on a subset of the validation set. The best performing feature sets for each affinity type are reported in Table 1. Our findings on the importance of each feature for each affinity type are in agreement with [18] (the most contributive are highlighted in bold in the table).

Affinity type	Set of features
i. within frame	{**sta**, sta_{χ^2}, **stm**, stm_{χ^2}, **aba**, abm, sd, $text$, $text_{\chi^2}$, $size$}
ii. across 1 frame	{**sta**, sta_{χ^2}, stm, **stt**, **sd**, $text$, $text_{\chi^2}$, $size$}
iii. across 2 frames	{**ltt**, **cit**, **stt**, sd, $size$}
iv. across >2 frames	{**ltt**, **cit**, sd, td}

Table 1: Set of features for learning.

Our experiments confirm that only considering pairs of



a. Within b. Across 1 c. Across 2 d. Across > 2

Figure 2: Affinity scores designed by [18] vs learned affinities.

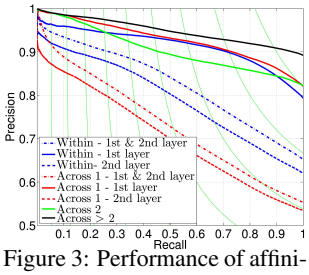


Figure 3: Performance of affinities, defined by the original graph topology [19].

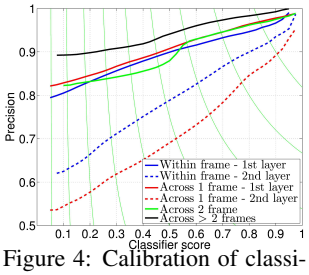


Figure 4: Calibration of classifier scores.

superpixels in the training set which have at least 60% overlap with ground truth objects improves results, as also noted in [37, 41]. Further stricter thresholds do not benefit the performance and also reduce the number of training samples.

In Figure 2, we plot precision-recall curves comparing our learnt affinities against the original ones of [18], for which weighted-product combinations of motion, appearance and shape features were hand-tuned. Note that the improvement of our curve (red) is particularly prominent at the high-precision regimes. High precision scores are important as they correspond to decisions taken with most confidence, thus most detrimental to the graph partitioning when wrong.

Implementation details. We use the Random Forest implementation of [15]. The number of features to sample for each node split is set to $\sqrt{\mathrm{F}}$, where F is the dimensionality of the feature space. As weak learners we use binary split functions, and the maximum tree depth is set to 50. Split thresholds are chosen to optimize the Gini impurity. The minimum number of data points required to split a node in the tree is set to 15. Ensemble averaging is used to fuse the predictions of trees. Other parameters, such as number of trees [250, 350, 150, 300] and minimum number of data points allowed at leaves [10, 15, 5, 15] are validated on the subset of the validation data and differs for each affinity type, depending on the dimensionality of training sets.

3.2. Topology of the graph

Note from Figure 2 the arguable overall performance (red curves) of the affinities learnt for the across 1 (Fig. 2(b)) and the across 2 type (Fig. 2(c)). The across 1 type have 55% precision (we take the overall precision at 100% recall). These have therefore 55% chance of correctness compared to 82% of the across 2 type learnt affinities. The across 1 affinities should ideally be more accurate, as they connect superpixels closer in time.

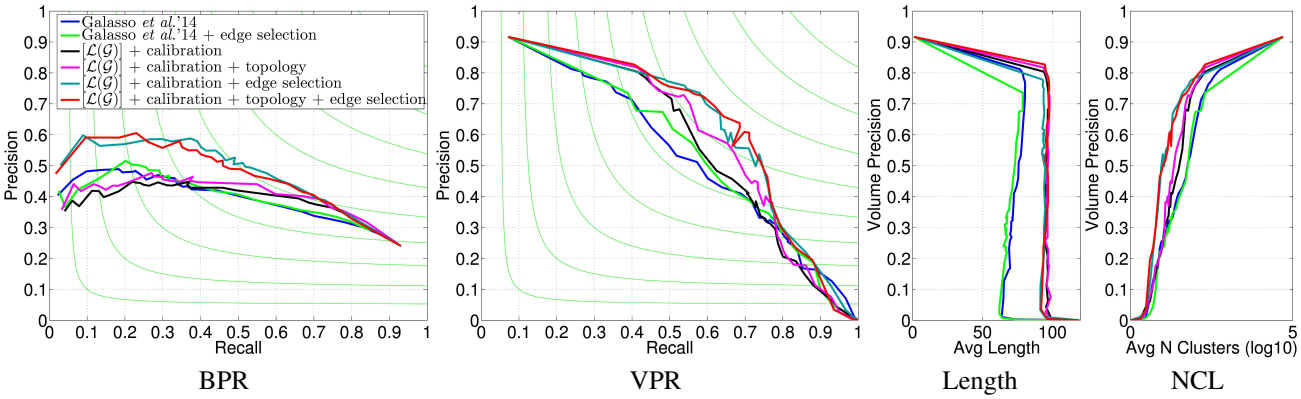
Let us take a closer look into the graph topology of [19], i.e. the edge connectivity $\mathcal{E}$. In the case of connectivity between superpixels within or across 1 frame, the graph is densified by using edges among neighboring superpixels (we call these *layer-1 neighbors*) and among more distant superpixels which share the same neighbor (we name these *layer-2 neighbors*). While the across 1 type affinities consider both direct temporal neighbors (best temporally-matching superpixel edges, according to optical flow propagation) and layer-2 neighbors, the across 2 type affinities only consider layer-1 neighbors.

We propose to treat the topologically different neighbors separately, which we illustrate in Figure 3, whereby we plot precision-recall curves for all types of our learnt affinities. We separate the two topologies both for the within and the across 1 type and re-learn separate classifiers. The results in Figure 3 show that now the layer-1 across 1 affinities reaches the overall performance (85%) of the across 2 affinity, and the corresponding performance of the within type also raises to 80%. As for the across 2 type, also the across $>$ 2 type only has layer-1 neighbors and is therefore not affected by the topological procedure.

Taking into account the topology of the graph increases performance and improves the edge-selection procedure (cf. Sec. 3.4). Treating separately the two neighbor layers, video segmentation performance increases (on the validation set) by 2% on the BPR and 3% on the VPR measures of VSB100 [20] (cf. Fig. 5). (These experiments are conducted by changing the topology of the graph and selecting edges with precision higher than 97% for all affinity types.)

3.3. Calibration of classifier outputs

An ideal subsequent processing of the graph would be the selection of the most likely edges (assuming that these be correct) and the deletion of wrong ones. This is desirable because it sparsifies the graph and reduces the chance of segmentation errors. For this purpose the classifier scores should correspond to the confidence measure of two super-



	BPR			VPR			Length	NCL
Algorithm	ODS	OSS	AP	ODS	OSS	AP	$\mu(\delta)$	μ
Galasso *et al.* CVPR'14 [19]	0.46	0.49	0.37	0.54	0.61	0.56	75.94(46.39)	60
Galasso *et al.* CVPR'14 [19] + edge selection	0.47	0.51	0.37	0.57	0.62	0.57	74.14(46.26)	70
Proposed $[\mathcal{L}(\mathcal{G})]$ + calibration	0.49	0.53	0.36	0.59	0.65	0.59	96.83(34.42)	50
Proposed $[\mathcal{L}(\mathcal{G})]$ + calibration + topology	0.51	0.54	0.38	0.62	0.67	0.62	97.85(33.55)	60
Proposed $[\mathcal{L}(\mathcal{G})]$ + calibration + edge selection	**0.52**	0.57	**0.44**	0.65	**0.70**	0.64	94.23(37.59)	20
Proposed $[\mathcal{L}(\mathcal{G})]$ + calibration + topology + edge selection	**0.52**	**0.58**	**0.44**	**0.66**	**0.70**	**0.65**	96.34(37.24)	20

Figure 5: Comparison of the proposed graph learning method with the baseline algorithm of [19], on the validation set of VSB100 [20]. The plots and table show BPR and VPR measures, aggregate performances ODS, OSS and AP, and length statistics (mean μ, std. δ, no. clusters NCL) (cf. Sec. 3.4 for details).

pixels being merged together. However, the classifier outputs for different affinity types have different ranges and provide different confidence levels.

We propose a probabilistic interpretation of the learnt scores and to calibrate the classifier outputs based on their performance on the validation set. We define a linear mapping $\Pi : S \mapsto P$, such that the classifier score s is approximated by its precision value p. We mean by precision p the ratio of true positive edges among all weights higher than or equal s. Precision is taken as a proxy to the true posterior probability (affinity between two nodes).

For each affinity type we estimate its own calibration function, which is illustrated in Figure 4. This calibration is an easier interpretation of the classifier outputs and serves to align the scores to their quality. This is important when combining multiple classifiers, as also noted in [22].

The calibration of classifier outputs is not dependent on the choice of the learning algorithm. The proposed procedure provides a way to encode edge weights and in our experience can help to improve the clustering performance. The calibrated classifier output scores are used as edge weights in the graph.

3.4. Edge selection

Following the argument of the previous section, we now modify the graph structure by reducing the number of edges and selecting the ones with high confidence. Each affinity type is thresholded with some confidence level, reducing the number of temporal and spatial edges in the graph. The goal is to have a connected graph with a minimal set of the most certain edges, as for maximal sparsity and the least chance of segmentation error.

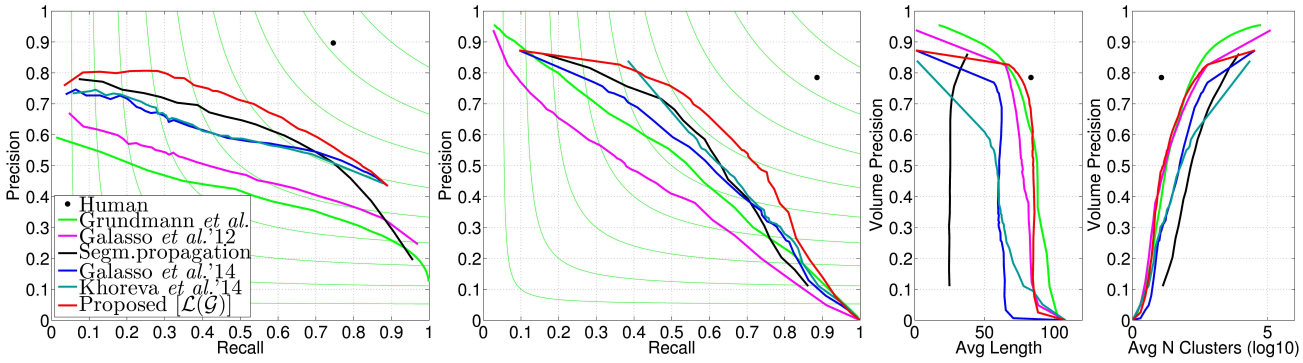
For finding the optimal thresholds for each affinity type grid search is applied. We find the confidence levels for four affinity types which provide the best performance on the validation set. We measure the performance as the sum of F-measures (ODS, OSS) and AP for BPR and VPR metric. We restrict the candidate space of the thresholds for each affinity type to [0.5;1], as the goal is to leave the most confident edges which have at least 50% precision. Edge selection turns out to be essential for best performance, cf. the next discussion.

We also explored other procedures for edge selection, such as kNN, but they all underperform by large margins. Our edge selection produces a potentially unbalanced (nodes have different number of neighbors) but better graph.

3.5. Discussion

In Figure 5, we analyze how the learnt graph $[\mathcal{L}(\mathcal{G})]$ and the proposed steps improve on the (validation) performance, with respect to the baseline algorithm of [19].

Given a learnt and calibrated graph (3rd row), topology improves 2.2% (4th row, average improvement over all six measures) while edge selection improves 5.2% (5th row). Edge selection is thus more important than topology. Adding topology on top of edge selection further contributes 0.5%. The importance of edge selection contrasts previous literature [43, 51, 30], all concerned with edge



	BPR			VPR			Length	NCL
Algorithm	ODS	OSS	AP	ODS	OSS	AP	$\mu(\delta)$	μ
Human	0.81	0.81	0.67	0.83	0.83	0.70	83.24(40.04)	11.90
Grundmann *et al.* [21]	0.47	0.54	0.41	0.52	0.55	0.52	87.69(34.02)	18.83
Galasso *et al.*'12 [18]	0.51	0.56	0.45	0.45	0.51	0.42	80.17(37.56)	8.00
Segm. propagation [20]	0.61	0.65	0.59	0.59	0.62	0.56	25.50(36.48)	258.05
Galasso *et al.*'14 [19]	0.62	0.66	0.54	0.55	0.59	0.55	61.25(40.87)	80.00
Khoreva *et al.*'14 [29]	0.61	0.66	0.52	0.58	0.61	0.58	51.72(39.90)	176.65
Proposed [$\mathcal{L}(\mathcal{G})$]	**0.64**	**0.70**	**0.61**	**0.63**	**0.66**	**0.63**	83.41(35.27)	50.00

Figure 6: Comparison of state-of-the-art video segmentation algorithms with our proposed method on the test set of VSB100 [20] (cf. Sec. 4 for details).

	BPR			VPR			Length	NCL
Algorithm	ODS	OSS	AP	ODS	OSS	AP	$\mu(\delta)$	μ
Galasso *et al.*'14 [19] - 1–SC [8, 23]	0.61	0.64	0.52	0.55	0.60	0.54	69.80(42.26)	19.00
Proposed [$\mathcal{L}(\mathcal{G})$] - 1–SC [8, 23]	**0.63**	**0.69**	**0.63**	**0.60**	**0.65**	**0.59**	78.75(38.49)	25.00
Galasso *et al.*'14 [19] - GRACLUS [13]	0.59	0.64	0.51	0.34	0.46	0.31	52.35(38.20)	15.00
Proposed [$\mathcal{L}(\mathcal{G})$] - GRACLUS [13]	**0.62**	**0.67**	**0.52**	**0.54**	**0.60**	**0.53**	94.85(28.43)	18.00
Galasso *et al.*'14 [19] - MCL [52]	0.59	0.64	**0.45**	0.40	0.46	0.37	34.80(38.30)	37.32
Proposed [$\mathcal{L}(\mathcal{G})$] - MCL [52]	**0.64**	**0.68**	0.39	**0.58**	**0.59**	**0.59**	31.44(46.83)	68.78

Table 2: General applicability of the proposed graph construction. We have tested different clustering methods with the graph of [19] and our learnt graph. In all cases the learnt graph yields better performance and thus generalizes beyond the employed spectral clustering.

weights.

To further test the importance of edge selection, we have applied this to the baseline algorithm of [19] (1st and 2nd rows). The improvement is only marginal (1.3%). We conclude therefore that a pre-requisite for successful edge selection is weight calibration plus the good performance of the classifier in the high precision regime (see Fig. 2).

4. Comparison with state-of-the-art video segmentation methods

In Figure 6 we compare the proposed method to the baseline [19] and state-of-the-art video segmentation algorithms [21, 18, 20, 29] on the test set of VSB100 [20]. We consider the graph [$\mathcal{L}(\mathcal{G})$] with the learnt topology and edge weights proposed in Section 3.

The proposed method improves the performance of [19] on both BPR and VPR by a large margin, as it appears both in the plots and the tables (average improvement of 4% in BPR and 8% in VPR, 6% on all measures). We outperform all recent video segmentation algorithms and the challenging segmentation propagation baseline [20].

The proposed graph construction however is directly applicable to other graph-based techniques. We have tested different graph partitioning methods [8, 23, 13, 52] with the graph of [19] and our learnt graph, the results are presented in Table 2. For all three tested methods our learnt graph improves significantly the performance both on BPR and VPR (up to 6–10% on average). This shows that our graph construction generalizes beyond the employed spectral clustering technique. Note that the 1-spectral clustering approach [8, 23] outperforms spectral clustering in terms of AP with respect to BPR while being worse on VPR.

Regarding runtime, the efficiency of the algorithm depends on the number of superpixels n (nodes in the graph). The (test-time) Random Forests classification runtime is negligible with respect to feature computation and graph partition. In spectral clustering, the bottleneck is the eigendecomposition: the Lanczos method has complexity $O(kE)$

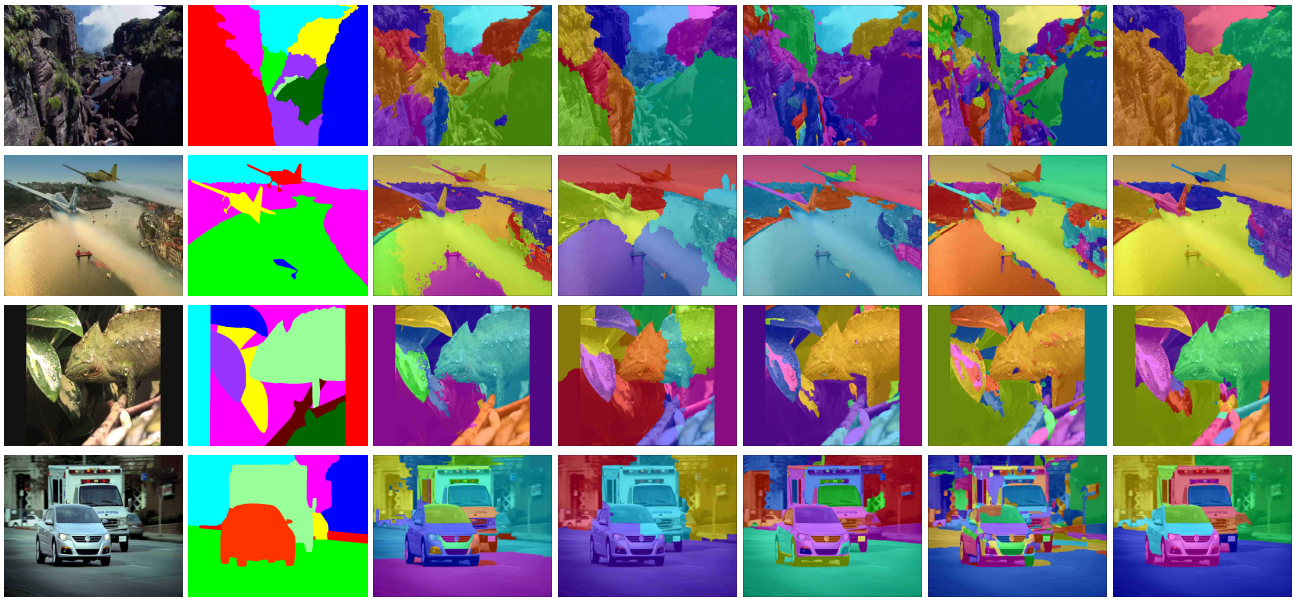


Figure 7: Comparison of video segmentation results of algorithms [21, 18, 20, 19] and our proposed method [$\mathcal{L}(\mathcal{G})$] to one of ground truths [20]. We report for each algorithm the coarse-to-fine segmentation level with best performance in VPR. Our approach qualitatively improves on the algorithm [19], better discriminating visual objects with less number of clusters (cf. Sec. 4 for details).

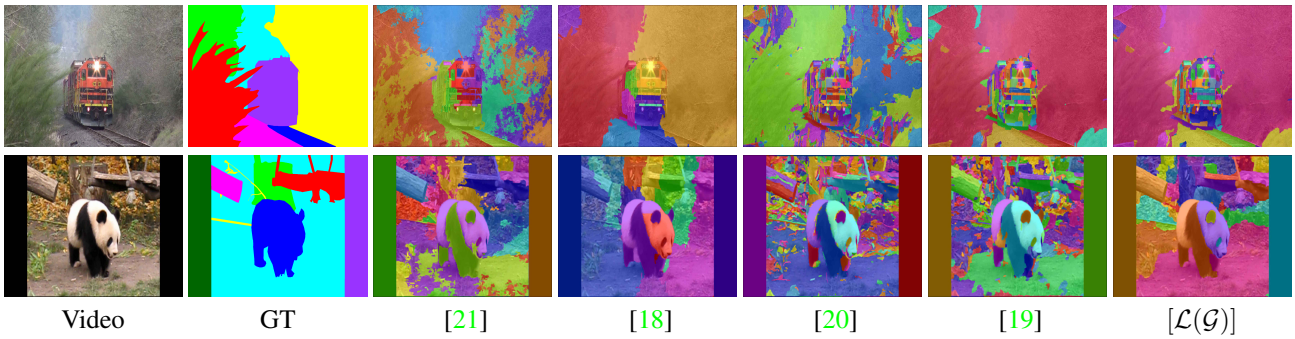


Figure 8: Failure cases for the algorithms [21, 18, 20, 19] and the proposed graph learning method [$\mathcal{L}(\mathcal{G})$]. All methods fail to correctly discern objects, oversegmenting the foreground and background due to the misleading appearance differences and textured background.

and the iteration number scales with $\sim \log E$ (k the number of eigenvectors and E the number of edges in the graph, which scales linearly with n, approx. $\sim 366n$). In our graph due to the edge selection procedure the average number of edges is reduced to 15% and the constructed graph is much sparser, hence the reduction in runtime of 55% with respect to [19]. E.g. runtime of "soccer" reduces from 4.8 min to 2.9 min, "hippo fight" from 9.3 min to 4.4 min.

We illustrate qualitative results, comparing in Figure 7 our proposed algorithm to state-of-the-art video segmentation methods [21, 18, 20, 19]. Figure 7 supports the positive quantitative results. The proposed approach allows to better distinguish visual objects with well-localized boundaries and limited label leakage. Segmentations provided by our method capture better motion and appearance self-contained within the objects, distinguishing the homogeneous areas of foreground and background with less number of clusters. However, a failure cases show further potential for improvement (see Figure 8).

5. Conclusions

In this paper we addressed the classifier based graph construction procedure for video segmentation. We proposed an empirical approach to learn both the edge topology and weights of the graph. While combining well-established features by means of a classifier and calibrating the classifier scores by its accuracy we alter the graph structure selecting the most confident edges. Our method of learning the graph helps to improve both performance on the challenging VSB100 benchmark as well as efficiency without changing the graph partitioning model.

References

[1] A. Alexandrescu and K. Kirchhoff. Data-driven graph construction for semi-supervised graph-based learning in NLP. In *HLT-NAACL*, 2007. 2

[2] B. Andres, J. H. Kappes, T. Beier, U. Köthe, and F. A. Hamprecht. Probabilistic image segmentation with closedness constraints. In *ICCV*, 2011. 2

[3] P. Arbeláez, M. Maire, C. C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 1, 2, 3, 4

[4] V. Badrinarayanan, I. Budvytis, and R. Cipolla. Mixture of trees probabilistic graphical model for video segmentation. *IJCV*, 2013. 1

[5] D. Banica, A. Agape, A. Ion, and C. Sminchisescu. Video object segmentation by salient segment chain composition. In *ICCV, IPGM Workshop*, 2013. 1

[6] M. V. D. Bergh, G. Roig, X. Boix, S. Manen, and L. V. Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013. 1

[7] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 1, 2, 3, 4

[8] T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. In *ICML*, 2009. 7

[9] J. Chang, D. Wei, and J. W. Fisher. A video representation using temporal superpixels. In *CVPR*, 2013. 1, 2

[10] H.-T. Cheng and N. Ahuja. Exploiting nonlocal spatiotemporal structure for video segmentation. In *CVPR*, 2012. 2

[11] J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Trans. Med. Imaging*, 2008. 1, 4

[12] C. Couprie, L. J. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *PAMI*, 2011. 2

[13] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *PAMI*, 2007. 7

[14] X. Di, H. Chang, and X. Chen. Multi-layer spectral clustering for video segmentation. In *ACCV*, 2012. 4

[15] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html. 5

[16] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 1

[17] K. Fragkiadaki and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *CVPR*, 2012. 1, 2, 4

[18] F. Galasso, R. Cipolla, and B. Schiele. Video segmentation with superpixels. In *ACCV*, 2012. 2, 3, 4, 5, 7, 8

[19] F. Galasso, M. Keuper, T. Brox, and B. Schiele. Spectral graph reduction for efficient image and streaming video segmentation. In *CVPR*, 2014. 1, 2, 3, 4, 5, 6, 7, 8

[20] F. Galasso, N. S. Nagaraja, T. Z. Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 1, 2, 3, 4, 5, 6, 7, 8

[21] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 1, 2, 7, 8

[22] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. *CoRR*, 2014. 6

[23] M. Hein and T. Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse pca. In *NIPS*, 2010. 7

[24] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007. 3

[25] P. Isola, D. Zoran, D. Krishnan, , and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014. 1

[26] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *ICCV*, 2013. 1, 2

[27] T. Jebara and S. Chang. Graph construction and bmatching for semi-supervised learning. In *ICML*, 2009. 2

[28] T. Jebara and V. Shchogolev. B-matching for spectral clustering. In *ECML*, 2006. 2

[29] A. Khoreva, F. Galasso, M. Hein, and B. Schiele. Learning must-link constraints for video segmentation based on spectral clustering. In *GCPR*, 2014. 7

[30] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Task-specific image partitioning. *TIP*, 2013. 2, 7

[31] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001. 3

[32] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 1, 2

[33] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 1

[34] M. Maier, U. von Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *NIPS*, 2009. 2

[35] M. Maire and S. X. Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. In *ICCV*, 2013. 1, 2, 4

[36] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001. 2, 4

[37] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-Temporal Object Detection Proposals. In *ECCV*, 2014. 5

[38] G. Palou and P. Salembier. Hierarchical video representation with trajectory binary partition tree. In *CVPR*, 2013. 1, 2, 3

[39] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 1

[40] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*, 2008. 2

[41] S. H. Raza, M. Grundmann, and I. Essa. Geoemetric context from video. In *CVPR*, 2013. 1, 5

[42] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 1

[43] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003. 2, 7

[44] M. Reso, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Temporally consistent superpixels. In *ICCV*, 2013. 1

[45] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000. 1, 2, 4

[46] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010. 3

[47] N. Sundaram and K. Keutzer. Long term video segmentation through pixel level spectral clustering on gpus. In *ICCV Workshops*, 2011. 1, 2, 4

[48] P. Sundberg, T.Brox, M. Maire, P. Arbelaez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, 2011. 4

[49] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei. Discriminative segment annotation in weakly labeled video. In *CVPR*, 2013. 1

[50] E. Taralova, F. D. la Torre, and M. Hebert. Motion words for videos. In *ECCV*, 2014. 1

[51] S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung. Maximin affinity learning of image segmentation. *CoRR*, 2009. 2, 7

[52] S. van Dongen. Graph clustering via a discrete uncoupling process. *SIAM J. Matrix Analysis Applications*, 2008. 7

[53] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, 2010. 2

[54] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007. 4

[55] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012. 1

[56] C. Xu, S. Whitt, and J. J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013. 1, 2

[57] C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012. 2

[58] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Pattern Recognition (Proc. DAGM)*, pages 214–223, 2007. 3

[59] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013. 1