A 3D Convolutional Approach to Spectral Object Segmentation in Space and Time

Elena Burceanu[1,2*] and **Marius Leordeanu**[3,4]
[1]Bitdefender
[2]University of Bucharest, Romania
[3]Institute of Mathematics of the Romanian Academy
[4]University Politehnica of Bucharest, Romania
eburceanu@bitdefender.com, marius.leordeanu@cs.pub.ro

Abstract

We formulate object segmentation in video as a spectral graph clustering problem in space and time, in which nodes are pixels and their relations form local neighbourhoods. We claim that the strongest cluster in this pixel-level graph represents the salient object segmentation. We compute the main cluster using a novel and fast 3D filtering technique that finds the spectral clustering solution, namely the principal eigenvector of the graph's adjacency matrix, without building the matrix explicitly - which would be intractable. Our method is based on the power iteration which we prove is equivalent to performing a specific set of 3D convolutions in the space-time feature volume. This allows to avoid creating the matrix and have a fast parallel implementation on GPU. We show that our method is much faster than classical power iteration applied directly on the adjacency matrix. Different from other works, ours is dedicated to preserving object consistency in space and time at the level of pixels. In experiments, we obtain consistent improvement over the top state of the art methods on DAVIS-2016 dataset. We also achieve top results on the well-known SegTrackv2 dataset.

1 Introduction

Elements from a video are interconnected in space and time and have an intrinsic graph structure (Fig. 1). Most existing approaches use higher-level components, such as objects, super-pixels or features, at a significantly lower resolution. Considering this graph structure in space-time, explicitly at the dense pixel-level, is an extremely expensive problem. Our proposed solution to video object segmentation, Spectral Filtering Segmentation (**SFSeg**), is based on transforming an expensive eigenvalue problem inspired from spectral clustering, into 3D convolutions on the space-time volume. This makes it fast, while keeping the properties of spectral clustering. We are the first, to our best knowledge, to propose a practical spectral clustering approach to video object segmentation at the pixel level, in space and time.

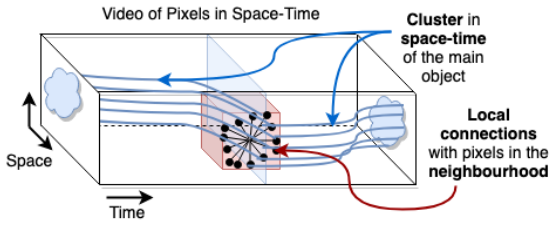


Figure 1: We see the video as a locally connected graph of pixels in space-time. The strength and the number of connections are enforcing the pixel membership to the salient video object.

Most state of the art algorithms for this task do not use the time constraint, and when they do, they take little advantage of it. Time plays a fundamental factor in how objects move and change in the world, but computer vision does not yet exploit it sufficiently. Consequently, the segmentation outputs of current state of the art algorithms is not always consistent over time. Our work comes to address precisely this aspect and our contribution is demonstrated through solid experiments on DAVIS-2016 and SegTrackv2 datasets on which we improve over state of the art methods.

We demonstrate in experiments that the eigenvector of the graph's adjacency matrix is a good solution for salient object segmentation. Once our filtering-based optimization converges, the segmentation map is spatio-temporally consistent, with a smooth transition between frames: noise coming from other objects is removed and missing parts of the object are added back. Through multiple iterations, the relevant information is propagated step by step to farther away neighbourhoods in space and time, acting like a diffusion.

Our contribution is two-fold. Besides formulating the segmentation problem in video as an eigenvalue problem on the adjacency matrix of the graph in space-time, we also provide a very fast optimization algorithm that computes the required eigenvector (which represents the desired segmentation) without explicitly creating or using the huge adjacency matrix. We prove theoretically and in practice that our algorithm reaches the same solution as a standard routine for eigenvector computation. We also show in experiments that the values in the final eigenvector, with one element per video pixel, confirm the spectral clustering assumption and provide an improved soft-segmentation of the main object.

*Contact Author

2 Related Work

Most state of the art methods for video object segmentation are using CNNs architectures pre-trained for object segmentation on large image datasets. They have a strong image-based backbone and are not designed from scratch with both space and time dimensions in mind. Many solutions [Khoreva *et al.*, 2017] adapt image segmentation methods by adding an additional branch to the architecture for incorporating the time axis: motion branch (previous frames or optical flow as) or previous masks branch (for mask propagation). Other methods are based on one-shot learning strategies and fine tune the model on the first video frame, followed by some post-processing refinement [Maninis *et al.*, 2018]. Approaches derived from OSVOS [2017] do not take the time axis into account. Our method comes to better address the natural space-time relationship, which is why it is effective when combined with frame-based segmentation algorithms.

Graph representations. Graph methods are suitable for segmentation and can have different representations, where the **nodes** can be pixels, super-pixels, voxels or image/video regions. Graph edges are undirected, modeled as symmetric similarity functions or directed [Chung, 2005]. Pixel-level representations are computationally extremely expensive, making the problem intractable for high resolution videos. Our fast solution implicitly uses a pixel-level graph representation: we make a first-order Taylor approximation of the Gaussian kernel (usually used for pairwise affinities) and rewrite it as a sequence of 3D convolutions in the video directly. Thus, we get the desired outcome without explicitly working with the graph. We describe it in detail in Sec. 3.

Spectral clustering. Computing eigenvectors of matrices extracted from data is a classic approach for clustering. There are several choices in the literature for choosing those matrices, the most popular being the Laplacian matrix [Ng *et al.*, 2001], normalized [Shi and Malik, 2000] or unnormalized. Other methods use the random walk matrix or directly the unnormalized adjacency matrix. Most methods are based on finding the eigenvectors corresponding to the smallest eigenvalues, while others, including our approach, require the leading eigenvectors. Graph Cuts are a popular class of spectral clustering algorithms, with many variants: normalized, average, min-max, mean cut and topological cut.

CRFs. Discriminative graphical models [Kumar and Hebert, 2003] are often applied over the segmentation of images and videos (denseCRF [Krähenbühl and Koltun, 2011]). CRFs are effective as they incorporate the observed data both at the level of nodes as well as edges. But they have a strict probabilistic interpretation and use inference algorithms that are significantly more expensive than the simpler eigenvector power iteration that we use for optimizing our non-probabilistic objective score.

Image segmentation. Graph cuts have been used in image segmentation [Shi and Malik, 2000]. They are expensive in practice, as they require the computation of eigenvectors of smallest eigenvalues for very large Laplacian matrices. Fast graph-based algorithm for image segmentation exist, such as [Felzenszwalb and Huttenlocher, 2004], which is linear in the number of edges and it is based on an heuristic for building the minimum spanning tree. It is still used as staring point by current methods. Another approach [Pourian *et al.*, 2015] is to learn image regions with spectral graph partitioning and formulate segmentation as a convex optimization problem.

Video Segmentation. Many video segmentation methods adapt existing image segmentation. In [Yu *et al.*, 2015] a parametric graph partitioning model over superpixels is proposed. Hierarchical graph-based segmentation over RGBD video sequences [Hickson *et al.*, 2014] also groups pixels into regions. The problem is solved using bipartite graph matching and minimizing the spanning tree. In [Li *et al.*, 2018], an efficient graph cut method is applied on a subset of pixels. To our best knowledge, all of the efficient methods group pixels into superpixels, regions from a grid or object proposals to handle the computational and memory burden. However, the hard initial grouping of pixels comes with a risk and could carry errors into the final solution, as it misses details available only at the original pixel resolution.

Our formulation is related to [Leordeanu and Hebert, 2005; Meila and Shi, 2001]. Our solution is the leading eigenvector of the adjacency matrix, computed fast and stably with power iteration as explained in Sec. 3. Using the unnormalized adjacency matrix in combination with power iteration is the least expensive spectral approach and the only one that can be factored into simple and fast 3D convolutions. This possibility gives our algorithm efficiency and speed (Sec. 4).

3 Our Approach

We formulate salient object segmentation in video as a graph partitioning problem (foreground vs background), where the graph is both spatial and temporal. Each node i represents a pixel in the space-time volume, which has $N = N_f \times H \times W$ pixels. N_f is the number of frames and (H, W) the frame size. Each edge captures the similarity between two pixels and is defined by the pairwise function $\mathbf{M}_{i,j}$. The pairwise connections between pixels i and j, in space and time are symmetric defining a $N \times N$ adjacency matrix $\mathbf{M}$. We take into account only the local connections in space-time, so $\mathbf{M}$ is sparse.

Let $\mathbf{s}$ and $\mathbf{f}$ be feature vectors of size $N \times 1$ with a feature value for each node. They will be used in defining the similarity function $\mathbf{M}_{ij}$ (Eq. 1). For now we consider the simplest case when $(\mathbf{s}_i, \mathbf{f}_i)$ represent single channel features (e.g. they could be soft masks, grey level values, edge or motion cues, or any pre-trained features). Later on we show how we can easily adapt the formulation to the multi-channel feature case. We define the edge similarity $\mathbf{M_{i,j}}$ using a Gaussian kernel:

$$\begin{aligned} \mathbf{M}_{i,j} &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2 - \beta \mathbf{dist}_{i,j}^2} \\ &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2} \mathbf{G}_{i,j} \end{aligned} \tag{1}$$

$$\mathbf{M}_{i,j} \approx \underbrace{\mathbf{s}_i^p \mathbf{s}_j^p}_{\text{unary terms}} \underbrace{[1 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2] \mathbf{G}_{i,j}}_{\text{pairwise terms}}. \tag{2}$$

In graph methods, it is common to use two types of terms for representing the model over the graph. Unary terms are about

individual node properties, while pairwise terms describe relations between pairs of nodes. In our case, $\mathbf{s}_i$, $\mathbf{s}_j$ describe individual node properties, whereas $\mathbf{f}_i$, $\mathbf{f}_j$ are used to define the pairwise similarity kernel between the two nodes. Note that in Eq. 2 we approximate the Gaussian kernel with its first-order Taylor expansion. The approximation is crucial in making our filtering approach possible, as shown next. Hyperparameters p and α control the importance of those terms.

To partition the space-time graph of video pixels, we want to find the strongest cluster in this graph. We first represent a segmentation solution (i.e., cluster in the space-time graph) with an indicator vector $\mathbf{x}$, that has one element for each node in the 3D space-time volume, such that $x_i = 1$ if node (pixel) i is in the video segmentation cluster (foreground) and $x_i = 0$ otherwise (background). We define the clustering score to be the sum over all pairwise similarity terms $\mathbf{M}_{ij}$ between the nodes inside the cluster. The higher this score, the stronger the sum of connections and the cluster. The segmentation score can be written compactly in matrix form as $S(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x}$. Similar to other spectral approaches in graph matching [Leordeanu and Hebert, 2005], we find the segmentation solution $\mathbf{x}_s$ that maximizes $S(\mathbf{x})$ under the relaxed constraints $\|\mathbf{x}\|_2 = 1$. Fixing the L2 norm of $\mathbf{x}$ is needed since only relative soft segmentation values matter. Thus, our optimization problem become one of maximizing the Raleigh quotient:

$$\mathbf{x}_s = \underset{\mathbf{x}}{\operatorname{argmax}}(\mathbf{x}^T\mathbf{M}\mathbf{x}/\|\mathbf{x}\|_2). \tag{3}$$

The global optimum solution is the principal eigenvector of $\mathbf{M}$. $\mathbf{M}$ is symmetric and has non-negative values, so the solution will also have non-negative elements, by Perron-Frobenius theorem [Frobenius, 1907]. The final segmentation could be simply obtained by thresholding. However, matrix $\mathbf{M}$, even for a small video has 20 million nodes (50 frames of 480×854), making the problem of finding the leading eigenvector with standard procedures intractable (Sec 4.2).

Next we show how to take advantage of the first-order expansion of the pairwise terms defining $\mathbf{M}$ and break power iteration into several very fast 3D convolutions in space and time, directly on the feature maps, without explicitly using the very big adjacency matrix. Our method receives as input pixel level feature maps and returns a final segmentation, as the solution $\mathbf{x}_s$ to problem 3.

3.1 Power Iteration with Pixel-wise Iterations

We apply power iteration algorithm to compute the eigenvector. At iteration $k+1$, we have Eq. 4:

$$\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j}\mathbf{x}_j^k, \tag{4}$$

where, after each iteration, the solution is normalized to unit norm and $\mathcal{N}(i)$ is the set of neighbors pixels with i, in space and time. Expanding $\mathbf{M}_{i,j}$ (Eq. 2), Eq. 4 becomes:

$$\mathbf{x}_i^{k+1} \leftarrow \alpha\mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p[\alpha^{-1} - \mathbf{f}_i^2 - \mathbf{f}_j^2 + 2\mathbf{f}_i\mathbf{f}_j]\mathbf{G}_{i,j}\mathbf{x}_j^k, \tag{5}$$

$$\begin{aligned}\mathbf{x}_i^{k+1} \leftarrow \alpha\mathbf{s}_i^p(\alpha^{-1} - \mathbf{f}_i^2) \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p\mathbf{G}_{i,j}\mathbf{x}_j^k - \\ \alpha\mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p\mathbf{f}_j^2\mathbf{G}_{i,j}\mathbf{x}_j^k + \\ 2\alpha\mathbf{s}_i^p\mathbf{f}_i \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p\mathbf{f}_j\mathbf{G}_{i,j}\mathbf{x}_j^k.\end{aligned} \tag{6}$$

3.2 Power Iteration Using 3D Convolutions

In Eq. 6 we observe that the links between the nodes are local ($\mathbf{M}$ is sparse) and we can replace the sums over neighbours with local 3D convolutions in space and time. Thus, we rewrite Eq. 6 as a sum of convolutions in 3D:

$$\begin{aligned}\mathbf{X}_{crt} \leftarrow \mathbf{S}^p \cdot (\alpha^{-1}\mathbf{1} - \mathbf{F}^2) \cdot G_{3D} * (\mathbf{S}^p \cdot \mathbf{X}^k) - \\ \mathbf{S}^p \cdot G_{3D} * (\mathbf{F}^2 \cdot \mathbf{S}^p \cdot \mathbf{X}^k) + \\ 2\mathbf{S}^p \cdot \mathbf{F} \cdot G_{3D} * (\mathbf{F} \cdot \mathbf{S}^p \cdot \mathbf{X}^k),\end{aligned} \tag{7}$$

$$\mathbf{X}^{k+1} \leftarrow \mathbf{X}_{crt}/\|\mathbf{X}_{crt}\|_2, \tag{8}$$

where $*$ is a convolution over a 3D space-time volume with a 3D Gaussian filter (G_{3D}), $\cdot$ is an element-wise multiplication, 3D matrices $\mathbf{X}^k, \mathbf{S}, \mathbf{F}$ have the original video shape ($N_f \times H \times W$) and $\mathbf{1}$ is a 3D matrix with all values 1. We transformed the standard form of power iteration in Eq. 4 in several very fast matrix operations: 3 convolutions and 13 element-wise matrix operations (multiplications and additions), which are local operations that can be parallelized.

3.3 Multiple Feature Channels

Our approach in Eq. 7 can easily accommodate multiple feature channels if we rewrite $\mathbf{M}_{i,j}$ from Eq. 2 and propagate it through Eq. 7, the final multi-channel solution is obtained by summing over the final solution for each channel:

$$\mathbf{X}_{crt}^{multi} = \sum_{m=1}^{N_{feat}} \mathbf{X}_{crt}(\mathbf{F}_c), \tag{9}$$

where $\mathbf{F}_c$ is one (3D) channel feature matrix.

4 Algorithm

We present the version of our algorithm (Alg. 1) that has a single channel feature map, but can be easily adapted to the multi-channel version, using Eq. 9. We first initialize the solution $\mathbf{X}$ with a uniform vector or with a soft-segmentation provided by another method, if it is available. We also initialize feature maps $\mathbf{S}$ and $\mathbf{F}$, which could be of any kind: lower-level (optical flow, edges, gray-level values) or higher-level pre-trained semantic features (deep features or initial soft/hard segmentation maps). At each iteration, we select a time frame around the current one. In Step 2, we multiply the corresponding matrices, apply the convolutions, compose the results and obtain the new segmentation mask for pixels in current frame, using the space-time operations (as in Eq. 7). Since the solution needs to be binary at the end (for evaluation), after each iteration (Step 3, line 14 in Alg. 1), we project the solution in a more discrete space (see Sec. 4.1).

Algorithm 1 Power iteration with 3D convolutions algorithm. At each iteration we pass through the whole video and compute the updated soft-segmentation X. First, we select a time window around current frame $[i-w, i+w]$. Secondly we compute the eigenvector with convolutions. Then, after each iteration, we binarize the solution (see Sec. 4.1).

$\mathbf{S}$ - unary feature maps for video
$\mathbf{F}$ - defines pairwise feature maps for video
$\mathbf{X}$ - salient object segmentation

$\mathbf{X} \leftarrow \mathbf{S}$
for $iter$ **in** $[1..N_i]$ **do**
  for i **in** $[1..N_f]$ **do**
    ▷ STEP 1. Take a temporal window around frame i:
    $\mathbf{S}_w, \mathbf{X}_w, \mathbf{F}_w \leftarrow T_{OF}(\mathbf{S}, \mathbf{X}, \mathbf{F})[i-w : i+w]$
    ▷ STEP 2. Compute new mask:
    $\mathbf{T1} \leftarrow (\alpha^{-1}\mathbf{1} - \mathbf{F}_w^2) \cdot G_{3D} * (\mathbf{S}_w^p \cdot \mathbf{X}_w)$
    $\mathbf{T2} \leftarrow -G_{3D} * (\mathbf{F}_w^2 \cdot \mathbf{S}_w^p \cdot \mathbf{X}_w)$
    $\mathbf{T3} \leftarrow 2\mathbf{F}_w \cdot G_{3D} * (\mathbf{F}_w \cdot \mathbf{S}_w^p \cdot \mathbf{X}_w)$
    $\mathbf{X}_{new}[i] \leftarrow \mathbf{S}_w^p \cdot (\mathbf{T1} + \mathbf{T2} + \mathbf{T3})$
  end for
  $\mathbf{X} \leftarrow \text{normalize}(\mathbf{X}_{new})$
  ▷ STEP 3. binarization:
  $\mathbf{X} \leftarrow \text{project\_binary}(\mathbf{X})$
end for

4.1 Binarization - Spectral vs Discrete Space

At the end, we need to have a hard segmentation map for the object of interest. Over the iterations, a spectral method makes the solution continuous. It was previously observed that in graph matching optimization, where the solution is relaxed [Leordeanu *et al.*, 2009], keeping it close to the initial discrete domain comes with a better final performance, even though the optimum in the spectral space is affected. So we integrated the binarization in the iterative optimization. After several iterations in the continuous space, we start projecting the solution on an almost discrete space through a sigmoid (which continuously approximates a step function) and initialize the next iteration with this projection. After the last iteration, we apply a hard threshold on a solution much closer to the discrete space than before. This way, the transition is smoother compared with a simple sharp thresholding.

4.2 Numerical Analysis

We compare the standard power iteration eigenvector computation with our filtering formulation, both from qualitative and quantitative (speedup) points of view.

Computational Complexity. Lanczos [1950] method for sparse matrices has $\mathcal{O}(kN_fN_pN_i)$ complexity for computing the leading eigenvector, where k is the number of neighbours for each node, N_f the number of frames in video, N_p the number of pixels per frame and N_i the number of iterations. Our full iteration algorithm has also $\mathcal{O}(kN_fN_pN_i)$ complexity, but with highly parallelizable operations, comparing to Lanczos. The Gaussian filters are separable, so the 3D convolutions can be broken into a sequence of three vector-wise convolutions, reducing the complexity $\mathcal{O}(k)$ for filtering to $3\mathcal{O}(k^{\frac{1}{3}})$: 3*7*7=147 vs 3+7+7=17 for a 3x7x7 kernel.

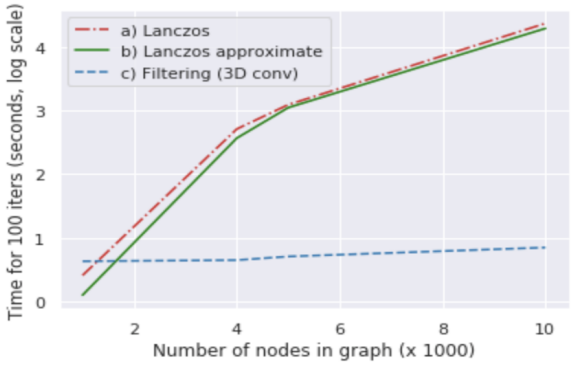


Figure 2: Total runtime in logarithmic scale for 100 iterations, including the time for building the adjacency matrix for power iteration. Our filtering formulation scales with the number of nodes, in contrast to power iteration, having an exponentially better time.

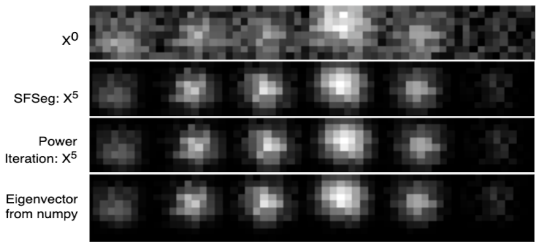


Figure 3: A toy examples for qualitative comparisons with soft masks for a six frames video. Starting with a very noisy input segmentation mask and showing: SFSeg segmentations after 5 iters; Power Iteration after 5 iters; the real main eigenvector. The results are almost identical, proving that SFSeg is a good approximation. More, for other methods, this is tractable only on toy examples.

We compare three solutions: **a)** Lanczos for the principal eigenvector for Eq. 1 **b)** Lanczos for the approximate adjacency matrix as in Eq. 2 **c)** our 3D convolutions approach. For a small graph of 4000 nodes (a video with 10 frames of 20×20 pixels), **a)** and **b)** have 0.15 sec/iter and our 3D filtering formulation has 0.02 sec/iter (Fig. 2). Our approach scales better, having a huge advantage when working with videos with millions of nodes because we do not explicitly build the adjacency matrix and filtering is parallelized on GPU.

Qualitative analysis. We perform tests on synthetic data, in order to study the differences between the original spectral solution using the exponential pairwise scores (1) and the one obtained after our first-order Taylor approximation trick (2). In Fig. 3 we see qualitative comparisons between the solutions obtained by three implementations: our SFSeg, power iteration with original pairwise scores and *numpy* eigenvector with original pairwise scores. The output is almost identical. In the synthetic experiments, the input is noisy, but all spectral solutions manage to reconstruct the initial segmentation.

Quantitative analysis. We analyze the numerical differences between the original eigenvector and our approximation (SFSeg). We plot the angle (in degrees) and the IoU (Jaccard) between SFSeg (first-order approximation of pairwise

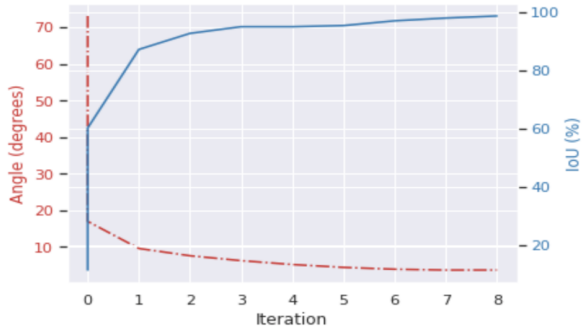


Figure 4: The angle and the IoU between real eigenvector and our SFSeg solution. The evolution of those metrics is monitored over multiple SFSeg iterations.

functions, optimized with 3D convolutions) and the original eigenvector (exponential pairwise functions in the adjacency matrix), over multiple SFSeg iterations in Fig. 4. Note that in these experiments we intentionally start from a far away solution (70 degrees difference between the SFSeg initial segmentation vector and the original eigenvector) to better show that SFSeg indeed converges to practically the same eigenvector. Such comparisons can be performed only on synthetic data with relatively small videos, for which the computation of the adjacency matrix needed for the original eigenvector is tractable. The results clearly show that SFSeg, with first order approximations of the pairwise functions on edges and optimization based on 3D filters, reaches the same theoretical solution, while being orders of magnitude faster.

5 Experimental Analysis

Experiments on DAVIS-2016. DAVIS-2016 [Perazzi *et al.*, 2016] is a densely annotated video object segmentation dataset. It contains 50 high-resolution video sequences (30 train/20 valid), with a total of 3455 annotated frames of real-world scenes. The benchmark comes with two tasks: the unsupervised one, where the solutions do not have access to the first frame of the video and the semi-supervised one, where the methods use the ground-truth from the first frame. In both setups, the methods can train the model on the training set and report their performance on the validation set. Our results are reported on the validation set, but we do not use the training set. For optical flow we used the Pytorch implementation of Flownet2 [Reda *et al.*, 2017].

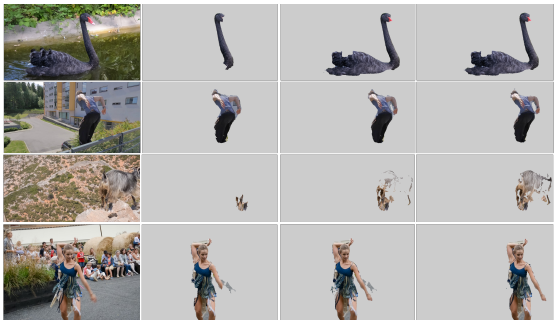
Experimental Setup. We test SFSeg with input from pre-computed segmentations of the video produced by top methods from DAVIS-2016, on both tasks. For the features maps, we initialized $\mathbf{S}$ with the pre-computed input segmentation values. For $\mathbf{F}$, we used two channels: the magnitude for the direct optical flow and for the reverse optical flow. We set: $N_i = 5$; $\alpha = 1$ and $p = 0.1$ for unsupervised task and $p = 0.2$ for the semi-supervised one. The algorithm is implemented as in Alg. 1 with the multi-channels as in Eq. 9.

In Tab. 1 we show the results of our method, SFSeg, when combined with top methods on DAVIS-2016, semi-supervised and unsupervised tasks. For a better understanding of the results, we also show the effect of applying SFSeg over other competitive, non-SOTA methods. We noted that the improvement is not related with the quality measure of the input. In some cases the improvement is stronger when input comes from stronger methods. Nevertheless, we consistently improve over the input method, whose segmentation mask we use to initialize the segmentation $\mathbf{X}_0$.

In Fig. 5 we show the iterative effect of SFSeg. Each example starts from the initial RGB frame and its initial segmentation (as produced by top DAVIS methods), and presents the segmentation at an intermediate iteration and the final one, when SFSeg reaches convergence.

	Input Method	Input Score (J)	**SFSeg** over Input (J)	Improved Videos (%)
Semi Supervised	OnAVOS	86.1	**86.3** (+0.2)	65
	OSVOS-S	85.6	**86.0** (+0.4)	90
	PReMVOS	84.9	**88.2** (+3.3)	90
	FAVOS	82.4	**83.0** (+0.6)	95
	OSMN	73.9	**75.9** (+2.0)	95
Un Supervised	COSNet	80.5	**80.9** (+0.4)	65
	MotAdapt	77.2	**77.5** (+0.3)	65
	PDB	77.2	**77.4** (+0.2)	60
	ARP	76.2	**77.7** (+1.5)	90
	LVO	75.9	**78.8** (+2.9)	90
	FSEG	70.7	**72.3** (+1.6)	95
	NLC	55.1	**55.6** (+0.5)	65
	Average Boost		+1.1%	80%

Table 1: Improvement over top segmentation methods on DAVIS-2016 tasks, validation set. SFSeg has the same hyper-parameters per task. We also included results for other competitive (non-SOTA) inputs. 2^{nd} column: Jaccard score for the input method; 3^{rd} column: score after applying SFSeg over the input method; 4^{th} column: the percentage of videos when the performance is improved after using SFSeg. The average SFSeg boost is 1.1% in Jaccard score and on average SFSeg raises performance for 80% of videos. Input methods : [Voigtlaender and Leibe, 2017; Maninis *et al.*, 2018; Luiten *et al.*, 2018; Lu *et al.*, 2019; M. Siam, 2019; Song *et al.*, 2018; Koh and Kim, 2017; Tokmakov *et al.*, 2017; Jain *et al.*, 2017; Cheng *et al.*, 2018; Yang *et al.*, 2018; Faktor and Irani, 2014].



Input Frames | Input Mask | SFSeg Iter2 | SFSeg Convergence

Figure 5: We present the evolution of SFSeg, over several iterations. Using the input segmentation mask (column 2) from top methods on DAVIS: ARP, FSEG and LVO, we show the intermediate value of the mask at Iter2 (column 3) and Iter4 (column 4).

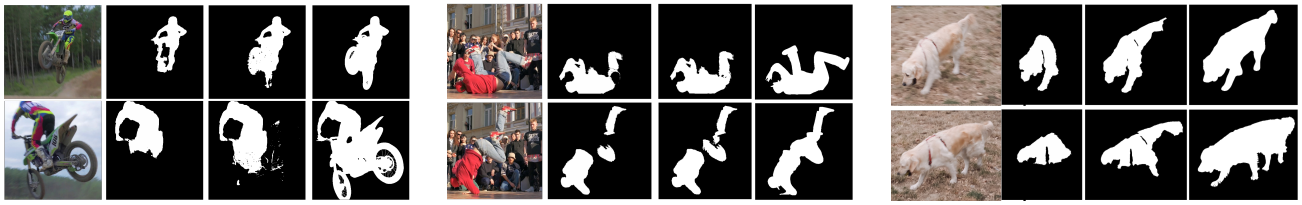


Figure 6: We show the output of SFSeg (col 3) over the input masks (col 2) received from top DAVIS-2016 solutions in various video frames (col 1). We see how the quality of the masks is increasing, bringing the input masks closer to ground truth (col 4). 1. PReMVOS - 3^{rd} place on semi-supervised (motocross-jump); 2. OnAVOS - 1^{st} place on semi-supervised (breakdance); 3. ARP - 2^{nd} place on unsupervised (dog).

Method	Score (J)
LVO	57.3
FSEG	61.4
OSVOS	65.4
NLC	67.2
MaskTrack	70.3
BB + **SFSeg** + denseCRF (ours)	**72.7**

Table 2: Comparative results on SegTrackv2. Our standalone solution, Backbone + SFSeg + denseCRF, obtains the best results among the other top methods in the literature. Input methods: [Tokmakov *et al.*, 2017; Jain *et al.*, 2017; Caelles *et al.*, 2017; Faktor and Irani, 2014; Khoreva *et al.*, 2017].

Method	DAVIS (J)	SegTrackv2 (J)
BB	67.2	72
BB + denseCRF	68.1	72
BB + SFSeg	68.7	72.1
BB + SFSeg + denseCRF	**69.2**	**72.7**

Table 3: Refinement Comparison. We compare SFSeg with denseCRF when applied to a competitive end-to-end Backbone (as detailed above). While SFSeg outperforms denseCRF when used individually, the two methods prove to be not only different, but also complementary, since combining them boosts the Jaccard score.

In Fig. 6 we show qualitative examples of our spectral method, SFSeg, applied over the initial mask, received as input from highly qualitative segmentation solutions on DAVIS-2016. The new masks show significant improvement without using other new means of supervision.

Experiments on SegTrackv2. SegTrackv2 [Li *et al.*, 2013] is a video object segmentation dataset, containing 14 videos, with multiple objects per frame. The purpose for video object segmentation task is to find the segmentation for all the objects in the frame (also split in two tasks: using the first frame or not). We use our standalone method, SFSeg, applied over the soft output of a competitive **Backbone (BB)**: UNet over ResNet34 pretrained features, fine tuned 40 epochs on salient object segmentation in images on DUTS dataset [Wang *et al.*, 2017] (with RectifiedAdam as optimizer). In Tab. 2 we show comparative results of our standalone method and other top solutions on the SegTrackv2 dataset.

SFSeg vs denseCRF. We compare SFSeg with denseCRF [Krähenbühl and Koltun, 2011], which is one of the most used refinement method in video object segmentation [Song *et al.*, 2018]. When applied over the same **Backbone** presented above, we observe that SFSeg brings a stronger improvement than denseCRF on both DAVIS-2016 and SegTrackv2 (Tab. 3). More, the two are complementary: in combination, the performance is boosted by the largest margin.

Running Time. The algorithm scales well, the running time being linear in the number of video pixels, as detailed in Sec. 4. For a frame of 480×854 pixels, it takes 0.055 sec per iteration, compared with 0.8 sec for denseCRF. Filtering takes 60% of time, the rest of 40% being used on other auxiliary operations (copying tensors). The time penalty of adding SFSeg is minor for most methods, which takes several seconds per frame (*e.g.* 4.5 sec per frame [Maninis *et al.*, 2018], 13 sec per frame [Luiten *et al.*, 2018]). We tested on a GTX Titan X Maxwell GPU, in Pytorch [Paszke *et al.*, 2017].

6 Conclusions and Future Work

We formulate video object segmentation as clustering in the space-time graph of pixels. We introduce an efficient spectral algorithm, Spectral Filtering Segmentation (SFSeg), in which the standard power iteration for computing the principal eigenvector of the graph adjacency matrix is transformed into a set of 3D convolutions applied on 3D feature maps in the video volume. Our original theoretical contribution makes the initial intractable problem possible. We validate experimentally that our solution based on a first-order Taylor approximation of the original pairwise potential used in spectral clustering is practically equivalent to the original one. In experiments, SFSeg consistently improves (for 80% of videos) over top published video object segmentation methods, at a small additional computational cost. Moreover, our method also achieves top performance in combination with other backbone networks (not necessarily state of the art). The consistent improvements in practice indicate that our spectral approach brings a new and complementary dimension to clustering in space-time, which is not fully addressed by current solutions. In the immediate future we will explore ways to learn more powerful features end-to-end, within our spectral clustering formulation.

Acknowledgments

part by UEFISCDI, under Projects EEA-RO-2018-0496 and PN-III-P1-1.2-PCCDI-2017-0734.

References

[Caelles *et al.*, 2017] S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. *CVPR*, 2017.

[Cheng *et al.*, 2018] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang. Fast and accurate online video object segmentation via tracking parts. *CVPR*, 2018.

[Chung, 2005] Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 2005.

[Faktor and Irani, 2014] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. *BMVC*, 2014.

[Felzenszwalb and Huttenlocher, 2004] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[Frobenius, 1907] G. Frobenius. "About a Fundamental Theorem of Group Theory. II. *Session Reports of the Royal Prussian Academy of Sciences*, 1907.

[Hickson *et al.*, 2014] S. Hickson, S. Birchfield, I. Essa, and H. Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. *CVPR*, 2014.

[Jain *et al.*, 2017] S. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv*, 2017.

[Khoreva *et al.*, 2017] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Hornung. Learning video object segmentation from static images. *CVPR*, 2017.

[Koh and Kim, 2017] Y. Koh and C. Kim. Primary object segmentation in videos based on region augmentation and reduction. *CVPR*, 2017.

[Krähenbühl and Koltun, 2011] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *arXiv*, 2011.

[Kumar and Hebert, 2003] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. 2003.

[Lanczos, 1950] C. Lanczos. An iteration method for the solution of the eigenvalue i problem . of linear differential and integral operators. 1950.

[Leordeanu and Hebert, 2005] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *ICCV*, 2005.

[Leordeanu *et al.*, 2009] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. *NIPS*, 2009.

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

[Li *et al.*, 2018] S. Li, B. Seybold, A. Vorobyov, X. Lei, and C. Kuo. Unsupervised video object segmentation with motion-based bilateral networks. *ECCV*, 2018.

[Lu *et al.*, 2019] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. *CVPR*, 2019.

[Luiten *et al.*, 2018] J. Luiten, P. Voigtlaender, and B. Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. *ACCV*, 2018.

[M. Siam, 2019] S. Lu L. Petrich M. Gamal M Elhoseiny M Jagersand M. Siam, C. Jiang. Video object segmentation using teacher-student adaptation in a human robot interaction (hri) setting. *ICRA*, 2019.

[Maninis *et al.*, 2018] K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. Video object segmentation without temporal information. *PAMI*, 2018.

[Meila and Shi, 2001] M. Meila and J. Shi. A random walks view of spectral segmentation. *AISTATS*, 2001.

[Ng *et al.*, 2001] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *NIPS*, 2001.

[Paszke *et al.*, 2017] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Autodiff in pytorch. 2017.

[Perazzi *et al.*, 2016] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Hornung. A benchmark dataset and evaluation methodology for video object segmentation. *CVPR*, 2016.

[Pourian *et al.*, 2015] N. Pourian, S. Karthikeyan, and B. Manjunath. Weakly supervised graph based semantic segmentation by learning communities of image-parts. *ICCV*, 2015.

[Reda *et al.*, 2017] F. Reda, R. Pottorff, J. Barker, and B. Catanzaro. flownet2-pytorch: Pytorch implementation of flownet 2.0: Evolution of optical flow estimation with deep networks, 2017.

[Shi and Malik, 2000] J. Shi and J. Malik. Normalized cuts and image segmentation. Technical report, 2000.

[Song *et al.*, 2018] H. Song, W. Wang, S. Zhao, J. Shen, and K. Lam. Pyramid dilated deeper convlstm for video salient object detection. *ECCV*, 2018.

[Tokmakov *et al.*, 2017] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. *ICCV*, 2017.

[Voigtlaender and Leibe, 2017] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. *BMVC*, 2017.

[Wang *et al.*, 2017] L. Wang, H. Lu, Y. Wang, D. Feng, M. sand Wang, B. Yin, and X. Ruan. Learning to detect salient objects with image-level supervision. *CVPR*, 2017.

[Yang *et al.*, 2018] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. Katsaggelos. Efficient video object segmentation via network modulation. *CVPR*, 2018.

[Yu *et al.*, 2015] C. Yu, H. Le, G. Zelinsky, and D. Samaras. Efficient video segmentation using parametric graph partitioning. *ICCV*, 2015.