A Spectral Approach to Unsupervised Object Segmentation in Video

Elena Burceanu
elena.burceanu@gmail.com

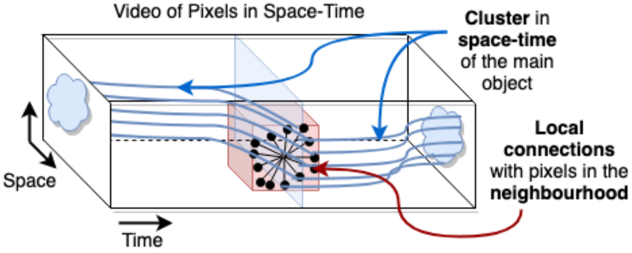
PhD - 3rd year, 1st semester report

April 23, 2019

Outline

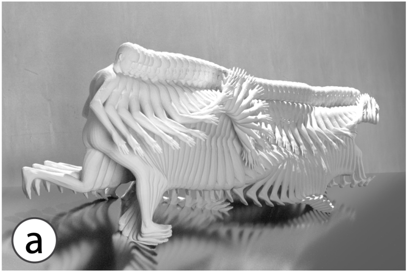
Introduction - Idea

- Video Object Segmentation - **partitioning Space-time Graph**
- Nodes - pixels, relations in local neighborhoods
- The strongest cluster = the salient object segmentation
- **Spectral clustering solution** = the principal eigenvector of the graph's adjacency matrix
- Based on power iteration (**without the explicit matrix** - intractable)
- New and fast 3D filtering technique in the space-time feature volume



Introduction - Results

- **Fast** parallel implementation on GPU
- Suitable for **online processing** of video streams
- Features: the output of existing segmentation algorithms, without any other supervision
- **Consistent improvement over top SoTA** methods on DAVIS-2016 dataset
- Both in **unsupervised and semi-supervised tasks**
- Same set of hyper-parameters



Outline

DAVIS and CNN Architectures

- Strong image-based backbone
- **Pre-trained for object segmentation** on other larger image datasets
- Adapt image segmentation solutions on videos (NOT designed for space-time)
- Approaches
  - **Temporal/motion branch** (previous frames/optical flow)
  - **Previous masks branch** (for mask propagation)
  - One-shot learning (fine-tune on the first video frame)
  - Approaches derived from OSVOS [1] do not take the time axis into account at all
- Heavily supervised **post-processing refinement** [7, 5]

Graph-based methods

- Graph representation
  - **nodes**: pixels, super-pixels, voxels or image/video regions
  - **edges**: undirected, modeled as symmetric similarity function
  - Representation influences both **accuracy** and **runtime**
- Problem
  - Partition a graph in 2 large components
  - Elements are inter-connected through high affinities inside each component
- Algos
  - Spectral clustering algorithms (find smallest or leading eigenvectors)
  - Laplacian: $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{M}\mathbf{D}^{-1/2}$, normalized $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{M}\mathbf{D}^{-1/2}$ or unnormalized $\mathbf{L} = \mathbf{D} - \mathbf{M}$
  - Random walk matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{M}$, the unnormalized adjacency matrix $\mathbf{M}$
  - Graph cut, [normalized, average, min-max, mean, topological] cut

Outline

Mathematical formulation

- VOS - graph partitioning problem (**foreground vs. background**)
- Spatial-temporal graph ($N = N_f \times H \times W$ pixels)
- Node i represents a pixel in the space-time volume
- N_f = number of frames; (H, W) = frame size
- Edge = **similarity between 2 pixels** $\mathbf{M}_{i,j}$ ($N \times N$ adjacency matrix $\mathbf{M}$ - symmetric and always non-negative, sparse - local connections)
- $\mathbf{s}$ and $\mathbf{f}$ = feature vectors of size $N \times 1$, one value for each pixel

$$\begin{aligned} \mathbf{M}_{i,j} &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2 - \beta \mathbf{dist}_{i,j}^2} = \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2} \mathbf{G}_{i,j} \\ &\approx \mathbf{s}_i^p \mathbf{s}_j^p [e^0 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2 e^0] \mathbf{G}_{i,j} \\ &\approx \underbrace{\mathbf{s}_i^p \mathbf{s}_j^p}_{\text{unary terms}} \underbrace{[1 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2] \mathbf{G}_{i,j}}_{\text{pairwise terms}}. \end{aligned} \tag{1}$$

Mathematical formulation

- VOS - graph partitioning problem (**foreground vs. background**)
- Spatial-temporal graph ($N = N_f \times H \times W$ pixels)
- Node i represents a pixel in the space-time volume
- N_f = number of frames; (H, W) = frame size
- Edge = **similarity between 2 pixels $\mathbf{M}_{i,j}$** ($N \times N$ adjacency matrix $\mathbf{M}$ - symmetric and always non-negative, sparse - local connections)
- $\mathbf{s}$ and $\mathbf{f}$ = feature vectors of size $N \times 1$, one value for each pixel

$$\begin{aligned}\mathbf{M}_{i,j} &= \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2 - \beta \mathbf{dist}_{i,j}^2} = \mathbf{s}_i^p \mathbf{s}_j^p e^{-\alpha(\mathbf{f}_i - \mathbf{f}_j)^2} \mathbf{G}_{i,j} \\ &\approx \mathbf{s}_i^p \mathbf{s}_j^p [e^0 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2 e^0] \mathbf{G}_{i,j} \\ &\approx \underbrace{\mathbf{s}_i^p \mathbf{s}_j^p}_{\text{unary terms}} \underbrace{[1 - \alpha(\mathbf{f}_i - \mathbf{f}_j)^2] \mathbf{G}_{i,j}}_{\text{pairwise terms}}.\end{aligned} \tag{1}$$

$$\mathbf{x}_s = \underset{\mathbf{x}}{\operatorname{argmax}} \frac{\mathbf{x}^T \mathbf{M} \mathbf{x}}{\|\mathbf{x}\|_2}. \tag{2}$$

Power iteration with pixel-wise iterations

$$\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j} \mathbf{x}_j^{k}, \tag{3}$$

Power iteration with pixel-wise iterations

$$\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j} \mathbf{x}_j^k, \tag{3}$$

$$\mathbf{x}_i^{k+1} \leftarrow \alpha \mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p [\alpha^{-1} - \mathbf{f}_i^2 - \mathbf{f}_j^2 + 2\mathbf{f}_i \mathbf{f}_j] \mathbf{G}_{i,j} \mathbf{x}_j^k, \tag{4}$$

Power iteration with pixel-wise iterations

$$\mathbf{x}_i^{k+1} \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{M}_{i,j} \mathbf{x}_j^k, \tag{3}$$

$$\mathbf{x}_i^{k+1} \leftarrow \alpha \mathbf{s}_i^p \sum_{j \in \mathcal{N}(i)} \mathbf{s}_j^p [\alpha^{-1} - \mathbf{f}_i^2 - \mathbf{f}_j^2 + 2\mathbf{f}_i \mathbf{f}_j] \mathbf{G}_{i,j} \mathbf{x}_j^k, \tag{4}$$

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

- matrix size for a small video: 20 millions nodes
- replace sum over neighbourhood with 3D convolution

Power iteration using 3D convolutions

$$\begin{aligned}\mathbf{X}_{crt} \leftarrow \mathbf{S}^p \odot (\alpha^{-1}\mathbf{1} - \mathbf{F}^2) \odot G_{3D} * (\mathbf{S}^p \odot \mathbf{X}^k) - \\ \mathbf{S}^p \odot G_{3D} * (\mathbf{F}^2 \odot \mathbf{S}^p \odot \mathbf{X}^k) + \\ 2\mathbf{S}^p \odot \mathbf{F} \odot G_{3D} * (\mathbf{F} \odot \mathbf{S}^p \odot \mathbf{X}^k),\end{aligned} \tag{6}$$

Power iteration using 3D convolutions

$$\begin{aligned}\mathbf{X}_{crt} \leftarrow \mathbf{S}^p \odot (\alpha^{-1}\mathbf{1} - \mathbf{F}^2) \odot G_{3D} * (\mathbf{S}^p \odot \mathbf{X}^k) - \\ \mathbf{S}^p \odot G_{3D} * (\mathbf{F}^2 \odot \mathbf{S}^p \odot \mathbf{X}^k) + \\ 2\mathbf{S}^p \odot \mathbf{F} \odot G_{3D} * (\mathbf{F} \odot \mathbf{S}^p \odot \mathbf{X}^k),\end{aligned} \tag{6}$$

$$\mathbf{X}^{k+1} \leftarrow \frac{\mathbf{X}_{crt}}{\|\mathbf{X}_{crt}\|_2}, \tag{7}$$

- $*$ = convolution over a 3D space-time volume; G_{3D} = 3D Gaussian filter; $\odot$ = element-wise multiplication; 3D matrices $\mathbf{X}^k, \mathbf{S}, \mathbf{F}$ - video shape ($N_f \times H \times W$); $\mathbf{1}$ = 3D matrix with all values 1
- very fast matrix operations: 3 convolutions and 13 element-wise matrix operations (multiplications and additions),
- local/easy to parallelize operations

Multiple feature channels

$$\mathbf{M}_{i,j} = \mathbf{s}_i^p \mathbf{s}_j^p [N_{feat} - \sum_{c=1}^{N_{feat}} \alpha_c (\mathbf{f}_{c,i} - \mathbf{f}_{c,j})^2] \mathbf{G}_{i,j}. \quad (8)$$

Multiple feature channels

$$\mathbf{M}_{i,j} = \mathbf{s}_i^p \mathbf{s}_j^p [N_{feat} - \sum_{c=1}^{N_{feat}} \alpha_c (\mathbf{f}_{c,i} - \mathbf{f}_{c,j})^2] \mathbf{G}_{i,j}. \quad (8)$$

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

- $\mathbf{F}_c$ is a (3D) channel feature matrix
- We can adapt to the case of multiple feature channels for **S**

Outline

Algorithm

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

$\mathbf{X} \leftarrow \mathbf{S}$
for *iter* ***in*** $[1..N_i]$ **do**
  for i ***in*** $[1..N_f]$ **do**
    // Step 1. Apply Optical flow warp $\mathbf{T}_{OF}$ for 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:
    $\mathbf{T1} \leftarrow (\alpha^{-1}\mathbf{1} - \mathbf{F}_w^2) \odot G_{3D} * (\mathbf{S}_w^p \odot \mathbf{X}_w)$
    $\mathbf{T2} \leftarrow -G_{3D} * (\mathbf{F}_w^2 \odot \mathbf{S}_w^p \odot \mathbf{X}_w)$
    $\mathbf{T3} \leftarrow 2\mathbf{F}_w \odot G_{3D} * (\mathbf{F}_w \odot \mathbf{S}_w^p \odot \mathbf{X}_w)$

    $\mathbf{X}_{new}[i] \leftarrow \mathbf{S}_w^p \odot (\mathbf{T1} + \mathbf{T2} + \mathbf{T3})$
  end
  $\mathbf{X} \leftarrow \text{normalize}(\mathbf{X}_{new})$
end

Figure: Power iteration with 3D convolutions algorithm. At each iteration we pass through the whole video and compute the updated soft-segmentation X. At Step 1 we warp S_w, X_w, F_w w.r.t the current frame, in a time window around it $[i-w, i+w]$, using pixel-wise displacements according to optical flow

Optical flow warping

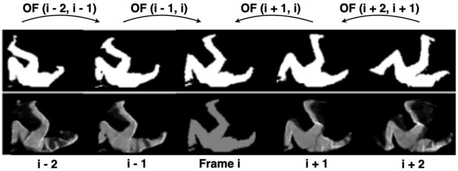


Figure: Align nearby frames using the optical flow displacement, w.r.t the center frame. The rows contain segmentation masks for five consecutive frames. The first row has the original input segmentation for **S**. The second row contains the new masks, after optical flow warping w.r.t center frame. Even though the optical flow warping is not perfect, we notice that the masks per frame after warping are more similar - thus they could form a stronger cluster in space and time.

- Remove **motion and deformations** differences between frames (alignment)

Online vs offline processing

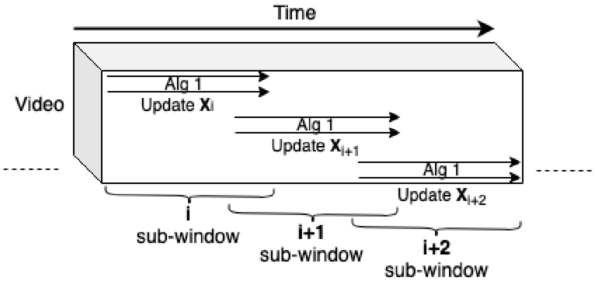


Figure: When the video is a contiguous stream or if it is very large, instead of applying power iterations on the full video, we can apply fewer iterations on smaller video sub-windows, with similar effect. To speed up convergence, we initialize the solution with the final solution over the previous sub-window (for the frames that overlap).

- **Partial iterations** by applying SFSeg on smaller sub-volumes of video

Numerical Complexity

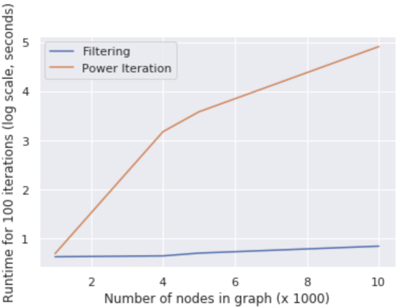


Figure: Total runtime in logarithmic scale for 100 iterations, including the time building the (big) adjacency matrix, for power iteration. For filtering algorithm, for a 4 million nodes graph, the time for 100 iterations is 74 seconds.

- Lanczos method for sparse matrices: $\mathcal{O}(kN_fN_pN_i)$
- SFSeg full iteration: $\mathcal{O}(kN_fN_pN_i)$ - with highly parallelizable operations; without explicit adjacency matrix

Synthetic Example

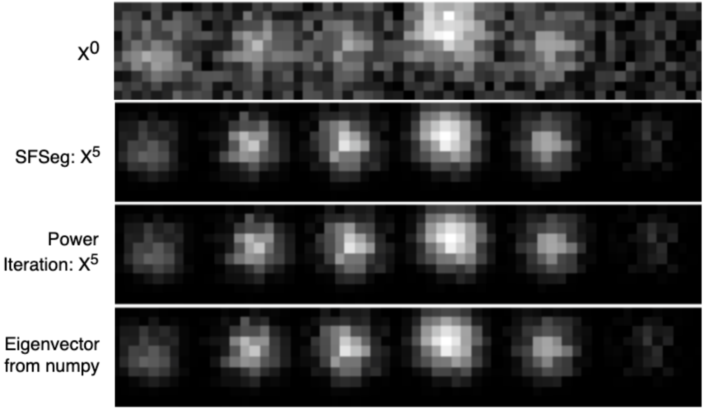


Figure: Soft masks for a 6 frame video: The first row contains the input segmentation mask, which is very noisy. The next line contains our SFSeg segmentations (iter 5). Next row corresponds to Power Iteration (iter 5). The last line contains the eigenvector computed with the numpy library.

Outline

Experimental Setup

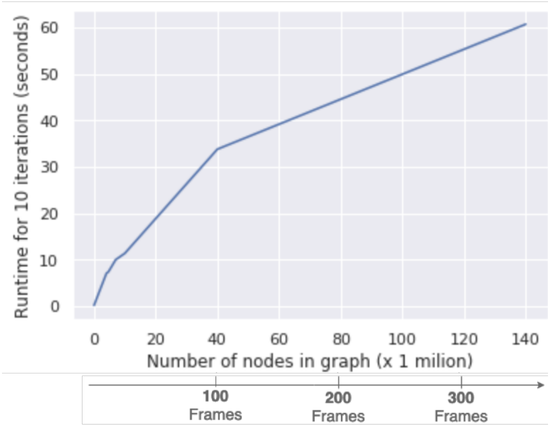
- DAVIS dataset
  - 50 video sequences, 3455 annotated frames of real-world scenes
  - Densely annotated, high-resolution videos
  - 2 tasks: semi-supervised and unsupervised (with/without access at first frame GT)
  - Train/Validation sets = 30/20 sequences
  - We don't use the training set
- SFSeg: input from pre-computed segmentations of the video produced by top methods from DAVIS

Results

Task	Input Method	Input Method (J)	SFSeg + Input Method (J)	Relative Boost (%)
Unsup DAVIS	PDB [26]	77.2	**77.3**	+0.44
	ARP [12]	76.2	**77.7**	+6.30
	LVO [27]	75.9	**78.8**	+12.03
	FSEG [10]	70.7	**71.9**	+4.10
Sup DAVIS	OnAVOS [28]	86.1	**86.3**	+1.44
	OSVOS-S [17]	85.6	**86.0**	+2.78
	PReMVOS [16]	84.9	**85.3**	+2.65

- Consistent improvement over: **all top 3** unsupervised and **all top 4** semi-supervised DAVIS-2016 methods
- We use the other methods input for: initialize the segmentation + single channel feature map
- For all input methods inside the two groups (unsupervised and semi-supervised task), the **hyper-parameters are identical**

Running time



- linear in the number of video pixels
- DAVIS-2016 experiments: +0.6 seconds per frame, on one GPU
- applied over the input segmentation from other solutions (4.5 sec per frame OSVOS-S , 13 sec per frame PReMVOS)

Video Difficulty Attributes

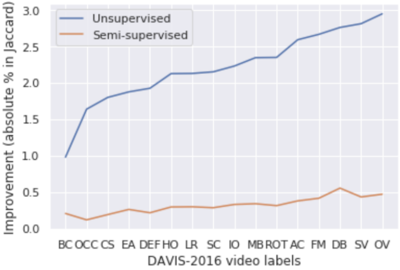


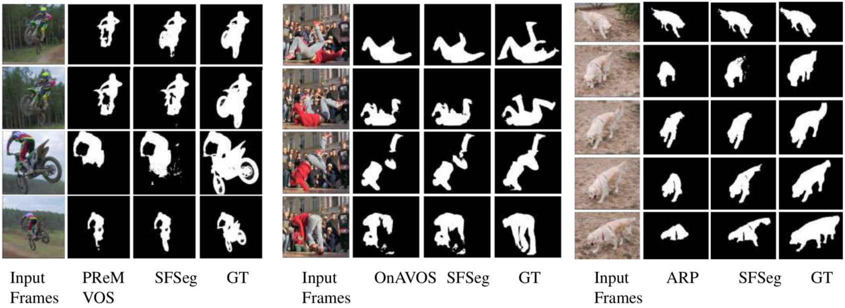
Figure: Improvement in Jaccard score, per video attribute - average over all videos, over all methods, per attribute, per task

- Consistent behaviour over tasks for SFSeg
- **Biggest gain**: attributes related to natural object shape variations

  (Out-of-view, Scale-Variation, Dynamic Background, Fast-Motion, Appearance Change)
- **Small gain**: depend less on the object and more on external factors

  (Background Clutter, Occlusion, Camera-Shake, Edge Ambiguity)

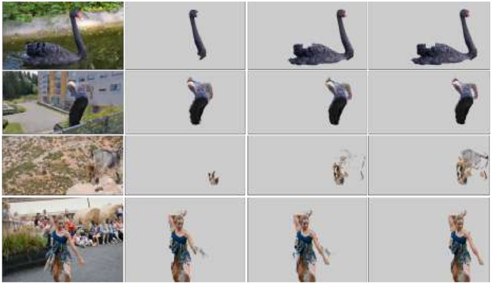
Qualitative Results I



1. PReMVOS [4] - 3rd place on semi-supervised, motocross-jump sequence
2. OnAVOS [7] - 1st place on semi-supervised, breakdance sequence
3. ARP [3] - 2nd place on unsupervised, dog sequence

- Input masks (col 2) received from top DAVIS-2016 solutions
- We see how the quality of the masks is increasing after applying SFSeg (col 3), bringing the input masks closer to GT (col 4)

Qualitative Results II



Input Frames | Input Mask | SFSeg Iter2 | SFSeg Convergence

- Input segmentation masks (col 2) from top methods on DAVIS-2016 ARP [3], FSEG [2], LVO [6]
- Mask evolution over SFSeg iterations
- We show the intermediate value of the mask at Iteration 2 (col 3) and at the last Iteration 4 (col 4)

Outline

Conclusions

- Segmentation in video as clustering in the **Space-time Graph of pixels**
- Efficient spectral algorithm: **Spectral Filtering Segmentation**
- Transformed the standard power iteration for computing the principal eigenvector of the graph adjacency matrix into a set of 3D convolutions directly on 3D feature maps in the video volume
- Theoretical contribution **makes the initial intractable problem possible**
- Consistently **improves over top** published VOS methods in both **unsupervised and semi-supervised** scenarios at a relatively minor additional computational cost

Thank you!

References I

Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool.
One-shot video object segmentation.
2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5320–5329, 2017.

Suyog Jain, Bo Xiong, and Kristen Grauman.
Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos.
arXiv preprint arXiv:1701.05384, 2017.

Yeong Jun Koh and Chang-Su Kim.
Primary object segmentation in videos based on region augmentation and reduction.
2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7417–7425, 2017.

References II

Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe.
Premvos: Proposal-generation, refinement and merging for video object segmentation.
Asian Conference on Computer Vision, 2018.

K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool.
Video object segmentation without temporal information.
IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018.

P. Tokmakov, K. Alahari, and C. Schmid.
Learning video object segmentation with visual memory.
ICCV, 2017.

References III

Paul Voigtlaender and Bastian Leibe.
Online adaptation of convolutional neural networks for video object segmentation.
BMVC, 2017.