The 2018 DAVIS Challenge on Video Object Segmentation - CVPR 2018 Workshops

Video Object Segmentation via Tracking Edges and Classifying Segments

Vahan Petrosyan, Oscar Örnsberg, Alexandre Proutiere
KTH Royal Institute of Technology
{vahanp,oornberg,alepro}@kth.se

Abstract

One of the major difficulties in semi-supervised video segmentation is to correctly track every pixel within consecutive frames. Optical flow algorithms are designed to solve this pixel-wise tracking problem, but they perform poorly when large object displacement, occlusion or drifting occurs. Tracking large group of pixels (segments) significantly reduces the tracking complexity. However, accurate segmentation of regions of interest constitutes an additional challenge to the problem. To address this issue, we propose a post-processing framework which consists (i) in refining the edges detected in the current frame, (ii) in using these edges as input to a novel segmentation algorithm (referred to as PALC), and (iii) for each of the obtained segment in extracting features from the optical flow and onAVOS algorithms, and feeding these features to a random forest predictor. The proposed framework yields performance improvements, and further provides an interactive tool for fast and detailed refinements of the given frame where the algorithm fails to produce high-quality segmentation.

1. Introduction

Pixel-level object tracking has recently received a lot of attention due to increasingly many applications in video editing, augmented reality, etc., [1, 2, 3]. Obtaining a pixel-accurate mask of each frame without human input is very challenging and current state-of-the-art unsupervised/semi-supervised techniques fail to achieve near human accuracy [1, 4]. The difficulty in pixel-level tracking is mainly due to effects such as large object displacement, occlusion, and drifting. In video editing applications, achieving near human-level accuracy is often necessary. In addition, tracking the objects is often case specific. For example, it is up to the video editing professional to decide how to track reappearing or splitting objects. Hence, having an interactive tool for performing fast and accurate refinements is essential in many applications. It is very time-consuming for a user to provide a pixel-accurate segmentation. According to [5], it takes 79s to annotate a single object in a frame using the polygon annotation tool. Instead of costly pixel-level masks, [2, 6, 7, 8] propose to employ point clicks, scribbles or text input to specify the target object in the refinement frames. This improves the annotation speed several times and it only takes 5-10 sec on average to label the object of interest. When the user refines the annotation of the particular frame, it is essential to use that information in the next frames to further improve the segmentation/annotation quality.

Once the editing is done in the current frame, it is important to have fast refinements in the next frames. Current state-of-the-art methods are slow and do not satisfy the speed criteria of the video editing applications. [9] takes 3 sec per frame on a modern Titan Xp GPU. More recently, [1] manages to achieve a performance of 2.4 frame per sec. However, for a modern high-resolution video, the proposed methods are still inapplicable.

In order to make the video segmentation more applicable in video editing, we propose a three step solution which does improve the object segmentation accuracy. Our solution consists of:

- An edge refinement network that uses the RCF edge detector [10] to refine the edges of the current frame;
- A modification of Penalized Average Linkage Cuts (PALC) [11] segmentation algorithm which uses the refined edges and automatically adjusts the number of segments;
- A Random Forest [12] algorithm which predicts the class of every segment based on the features extracted from the output of optical flow [13] and onAVOS [4] algorithms.

Our solution can be viewed as a post-processing step, where the RCF edge detector refines the segments of the given semi-supervised video segmentation algorithm. Given the output of the initial video segmentation algorithm, the post-processing runs in 15 frames per sec on 480p DAVIS segmentation images using a Titan Xp GPU. Our solution further provides a fast refinement tool for future

frames: For every frame, the user would receive the segmentation output, and the refinement of the segments could be done with a simple click or brush tool.

In our experiments, the proposed framework was tested on the output of the previous state-of-the-art algorithm onAVOS [4] and resulted in 1.9% and 0.6% improvement on average $\mathcal{J}$ and $\mathcal{F}$ scores, respectively. While we are not able to conclude that this step gives an improvement for all the other algorithms, we can imply that the proposed framework can provide fast refinements for all the video segmentation/annotation tasks.

2. Proposed Framework

The pipeline of the proposed framework is shown in Figure 1.

2.1. Edge Refinement

Traditional edge detection methods [14, 15] extract edges based on color, gradient, texture, or other manually designed features on the local neighborhood of each pixel. In recent years, deep learning based edge detectors significantly improved the quality of edge detection. HED [16] was one of the first real-time edge detection algorithms that significantly improved the performance of the previous state-of-the-art methods. More recently, [10] (RCF) achieved super-human performance on edge detection tasks. The edges extracted from RCF are class agnostic. In video segmentation, however, edge information extracted from the previous frames can be beneficial to perform edge refinement.

In this paper, we define an edge refinement network using the RCF edge detector. The network takes a 3-channel input consisting of the agnostic edges of RCF (Figure 1 (a)), the edges of a given video segmentation prediction (onAVOS in this case, Figure 1 (b)) and the optical flow [13] of the edges generated from the previous frame (Figure 1 (c)). Then, we train the RCF network to perform an edge refinement. Specifically, the network learns to get rid of unnecessary edges provided by the agnostic RCF and falsely detected edges provided by the given video segmentation algorithm. Figure 1 shows an example of such an input (a,b,c) where the corresponding output of refined edges is shown in (d). Note that the network correctly eliminated the agnostic edges on the ceiling (Figure 1 (a)), since it did not observe any object of interest there. In addition, it correctly eliminated the onAVOS edges generated on the frontman's body, while adding extra edges on his right leg.

2.2. Segmentation via Penalized Average Linkage Cut (PALC)

In this subsection, we present the Penalized Average Linkage Cut (PALC), an algorithm that takes as input the image and its adjusted edge information and outputs the segments by successively merging the two superpixels exhibiting the highest PALC similarity.

Superpixel Similarity: Consider an image $\mathcal{X} = \{x_\ell\}_{\ell=1}^N$ consisting of N pixels[1]. Let $C = \{c_\ell\}_{\ell=1}^N$[2] represent the contour information of image $\mathcal{X}$, given by an arbitrary contour detection algorithm. Let $\mathcal{S} = \{S_i\}_{i=1}^K$ represent the decomposition of $\mathcal{X}$ into K superpixels ($S_i \subset \{1, \dots, N\}$).

From the image $\mathcal{X}$, we construct a weighted graph of degree 4, $G = (\mathcal{V}, \mathcal{E}, W)$, whose vertices $\mathcal{V}$ are the pixels of the image, and whose edges $\mathcal{E}$ form a grid connecting neighbouring pixels (each pixel is connected to its up, down, left and right neighbour pixel). The weight of an edge represents the similarity of the corresponding two pixels: for pixels ℓ, ℓ', we define $w_{\ell,\ell'} = e^{-\frac{c_\ell + c_{\ell'}}{\sigma_1}}$ if $(\ell, \ell') \in \mathcal{E}$ and 0 otherwise. When the pixels ℓ, ℓ' actually belong to object boundaries, the contour values c_ℓ, $c_{\ell'}$ tend to be high and hence the weight $w_{\ell,\ell'}$ tend to be very small. The cut value of two superpixels S_i and S_j is then defined by:

$$\text{Cut}(S_i, S_j) = \sum_{\ell \in S_i} \sum_{\ell' \in S_j} w_{\ell,\ell'}.$$

Let μ_i represent the average intensity color in superpixel S_i: $\mu_i = \frac{1}{|S_i|} \sum_{\ell \in S_i} x_\ell$. We define the average linkage distance of two superpixels as $||\mu_i - \mu_j||_2$ and the average linkage similarity as

$$\text{Link}(S_i, S_j) = e^{-\frac{||\mu_i - \mu_j||_2}{\sigma_2}}.$$

One desirable feature of superpixel algorithms [17, 18] is to create superpixels of similar sizes. To this aim, we account for the sizes of superpixels to define their similarity. Specifically, the latter is chosen to be inversely proportional to the product of the superpixel sizes. Our definition of superpixel similarity is obtained by combining this feature to the cut value and the average linkage. More precisely, the similarity $A(S_i, S_j)$ of superpixels S_i and S_j is:

$$A(S_i, S_j) = \frac{\text{Cut}(S_i, S_j) \times \text{Link}(S_i, S_j)}{|S_i||S_j|}. \qquad (1)$$

We refer to this similarity measure as Penalized Average Linkage Cut (PALC). Compared to the well known Normalized Cut $\left(\frac{\text{Cut}(S_i,S_j)}{\text{Cut}(S_i,\mathcal{V})} + \frac{\text{Cut}(S_i,S_j)}{\text{Cut}(S_j,\mathcal{V})}\right)$ similarity ([19]), PALC has the following advantages:

– In the denominator, PALC metric penalizes stronger to the pair of large superpixels, and hence the algorithm tends to combine pairs of superpixels with both having small sizes.

– The weights $w_{\ell,\ell'}$ in the Cut function are calculated on the image contour which is more robust to noise than

[1] $x_\ell \in \mathbb{R}^3$ for RGB or CIELAB images, $x_\ell \in \mathbb{R}$ for gray scale images

[2] $c_\ell \in [0, 1]$, where $c_\ell = 1$ indicates boundary pixels

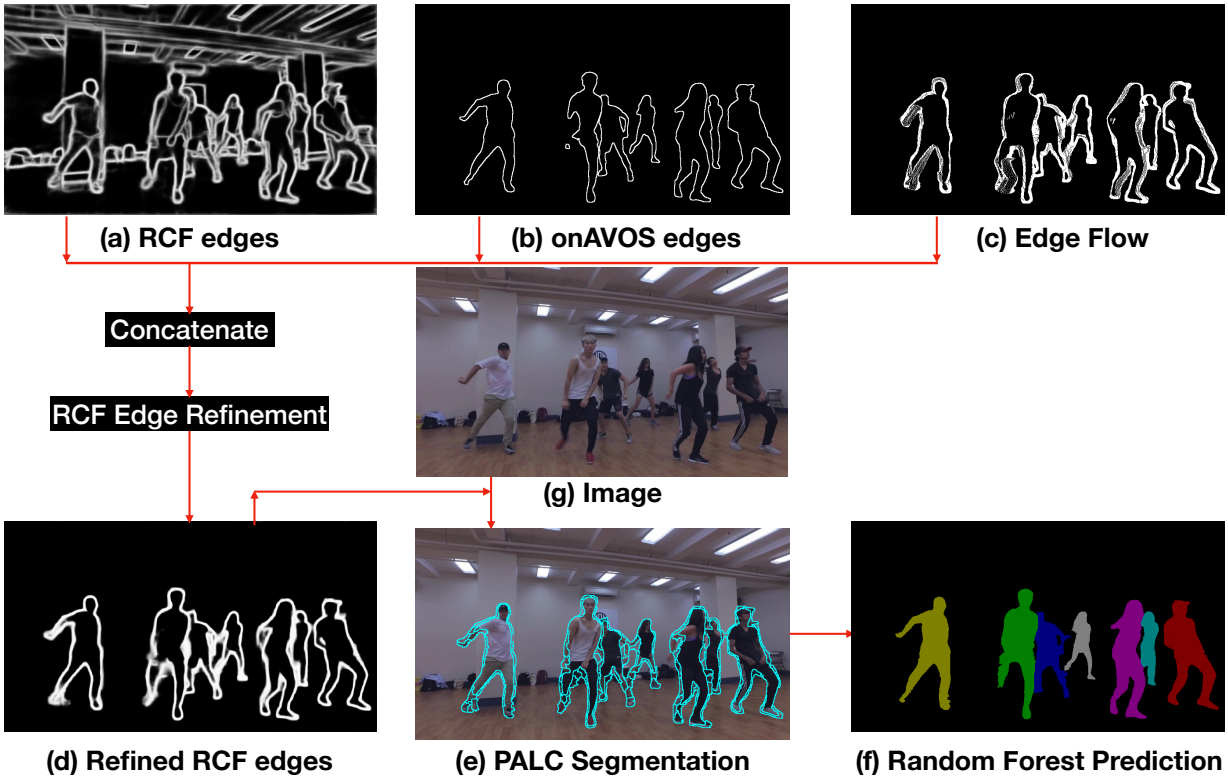

Figure 1. The pipeline of the proposed framework (better seen in color).

computing the weights based on the pixel intensity values of the original image.

– PALC has an additional term Link(S_i, S_j), promoting superpixels composed of pixels of similar colors.

PALC merging procedure: we compute the PALC similarity matrix $A = (A(S_i, S_j))_{i,j\in\{1,\dots,K\}}$ between each pair of neighboring pixels. The two pixels (later superpixels) with the highest PALC similarity are merged, and the PALC similarity matrix A is updated accordingly. We proceed with sequentially merging superpixels with highest PALC similarity and updating A until the total cut (T_{cut}) is smaller than a threshold value k. For the given K superpixels, we define the total cut by the following:

$$\mathrm{T}_{cut} = \sum_{i=1}^{K} \sum_{j\neq i} \mathrm{Cut}(S_i, S_j).$$

This sequential procedure is presented in Algorithm 1. Note that we merge superpixels with the highest PALC similarity only if the sum of their sizes is smaller than h times that of the smallest superpixel, denoted by $S_{(1)}$. Otherwise, we merge $S_{(1)}$ with its most similar superpixel. This feature helps to delete too small and too distinct superpixels. In our experiments, we take $k = h = 1000, \sigma_1 = 0.1, \sigma_2 = 50$. Figure 1 (e) shows an output of the PALC segmentation for the given refined edge (Figure 1 (d)) the and corresponding image inputs (Figure 1 (g))

Algorithm 1: PALC Superpixels

Input: image $\mathcal{X}$, contour $C, k, \sigma_1, \sigma_2, h$;
$\mathcal{S} \leftarrow 1 : N$;
Compute $A \leftarrow (A(S_i, S_j))_{i,j\in\{1,\dots,N\}}$ from $\mathcal{X}, C, \mathcal{S}$ using (1) ;
while $T_{cut} > k$ **do**
  Choose $(i, j) \in \mathrm{argmax}\, A(S_i, S_j)$;
  if $\frac{|S_i|+|S_j|}{|S_{(1)}|} < h$ **then**
    $S_i \leftarrow S_i \cup S_j, \mathcal{S} \leftarrow \mathcal{S} \setminus \{S_j\}$
  else
    $i' \in \mathrm{argmax}\, A(S_{(1)}, S_{i'})$;
    $S_{i'} \leftarrow S_{(1)} \cup S_{i'}, \mathcal{S} \leftarrow \mathcal{S} \setminus \{S_{(1)}\}$
  end
  Update A;
  Update T_{cut} ;
end
Output: $\mathcal{S}$;

2.3. Prediction using Random Forest

We extract some basic features such as segmentation size, average flow velocity of the segment and proportion of pixels that fall into certain segments. We train a random forest classifier to detect the class of each segment produced

by the PALC segmentation. For prediction, we add an additional class which indicates that the classifier prefers to keep the onAVOS's output since the given segment may include parts from two objects. We only change the output of the onAVOS algorithm if the prediction confidence of random forest is higher than p. In our experiments we take $p = 0.8$.

3. Conclusion and Future Work

We have presented a post-processing framework for video object segmentation. We have shown that the proposed framework is capable of improving the output of highly accurate video segmentation algorithms such as onAVOS. More importantly, the proposed framework provides the user an easy to use tool for fast and detailed video annotation and editing. In the future, we plan to replace the unsupervised PALC segmentation and the random forest classifier with more advanced deep learning based supervised segmentation algorithm. At the same time, we aim at decreasing the runtime of the refinement edge detector. As a result, we intend to provide an interactive tool for real-time detailed video annotation and editing.

References

[1] X. Li and C. C. Loy, "Video object segmentation with joint re-identification and attention-aware mask propagation," *arXiv:1803.04242*, 2018. 1

[2] A. Khoreva, A. Rohrbach, and B. Schiele, "Video object segmentation with language referring expressions," *arXiv:1803.08006*, 2018. 1

[3] Y.-T. Hu, J.-B. Huang, and A. G. Schwing, "Maskrnn: Instance level video object segmentation," *arXiv:1803.11187*, 2018. 1

[4] P. Voigtlaender and B. Leibe, "Online adaptation of convolutional neural networks for video object segmentation," *arXiv:1706.09364*, 2017. 1, 2

[5] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," *arXiv:1405.0312*, 2014. 1

[6] A. Benard and M. Gygli, "Interactive video object segmentation in the wild," *arXiv:1801.00269*, 2018. 1

[7] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. V. Gool, "Deep extreme cut: From extreme points to object segmentation," *arXiv:1711.09081*, 2017. 1

[8] Y. Chen, J. Pont-Tuset, A. Montes, and L. V. Gool, "Blazingly fast video object segmentation with pixel-wise metric learning," *arXiv:1804.03131*, 2018. 1

[9] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy, "Video object segmentation with re-identification," *arXiv:1708.00197*, 2017. 1

[10] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5872–5881, 2017. 1, 2

[11] V. Petrosyan, A. Proutiere, A. Aytekin, and Y. Liu, "Superpixel segmentation vie penalized average linkage cuts," in *To appear: European Conference on Computer Vision (ECCV)*, 2018. 1

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. 1

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," *arXiv:1612.01925*, 2016. 1, 2

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986. 2

[15] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *International Conference on Computer Vision (ICCV)*, pp. 1841–1848, 2013. 2

[16] S. He, R. Lau, W. Liu, Z. Huang, and Q. Yang, "SuperCNN: A superpixelwise convolutional neural network for salient object detection," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 330–344, 2015. 2

[17] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012. 2

[18] R. Giraud, V. T. Ta, and N. Papadakis, "SCALP: Superpixels with contour adherence using linear path," in *International Conference on Pattern Recognition (ICPR)*, pp. 2374–2379, 2016. 2

[19] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. 2