Monocular Occlusion Based Video Segmentation

Radford Parker Ahmad Humayun

Fig. 1. Shows the segmentation results on the TennisBalls sequence. **(a)** sample frame from input video. **(b)** occlusion posterior per pixel calculated using our random forest classifier. **(c)** shows oversegmentation result by just using RGB information. **(d)** shows oversegmentation augmented by depth information from Kinect. **(e)** shows oversegmentation by using RGB and occlusions computed in (b). Note that oversegmentation using occlusion (e) successfully reduces the number of false positives, in a fashion similar to the oversegmentation with depth (d).

I. Introduction

Dense video segmentation is considered a useful first step for many vision applications. It creates spacetime superpixels which help avoid processing millions of individual voxels. Traditionally, video segmentation methods work by simply observing low-level pixel cues when combining voxels to make space-time regions. This is typically done through appearance and optical flow cues. One thing amiss from these cues is the notion of depth in a scene. Depth can play a critical role in delineating object boundaries, especially where motion and appearance cues fail.

There are two ways to include depth information in a video segmentation pipeline: (1) with the inclusion of an RGBD sensor or (2) by inferring ordinal depths through occlusion boundaries [3], [7]. Even though the former solution simplifies the problem, it is less elegant since it does not allow processing of monocular RGB videos. In this paper we will improve the state-of-the-art video segmentation algorithm by detecting occlusion regions to infer depth boundaries. We will evaluate results by comparing the oversegmentations when using (1) no depth, (2) an RGBD sensor, and (3) occlusions to infer object boundaries.

II. Related Work

Most methods for video segmentation are based on photometric consistency [8]. Apart from using color, Grundmann *et al.* [2] uses optical flow explicity to provide a hierarchy of spacetime superpixels using Felzenszwalb and Huttenlocher [1] oversegmentation. Their algorithm trades the use of high-level cues like occlusions for computational efficiency. On the other hand, some techniques demonstrate the importance of occlusions in creating space-time superpixels [6].

A variety of methods exist for computing occlusions in frame pairs, including those based on graph-cuts [5] Recently, Humayun *et al.* [4] introduced a learning framework to infer occlusions on a per pixel basis. In this paper we use latter, in combination with [2], to improve video segmentation in monocular video.

III. Approach

Our algorithm has 3 steps: (1) compute forward and backward flow for a consecutive pair of frames from the input video; (2) compute occlusion for all pixels in the first frame; (3) use flow and occlusions to make space-time superpixels. Figure 2 gives an overview of our approach.

A. Occlusion Posterior

When a foreground object in a scene moves, some portion of the background gets occluded. This information is relevant for segmentation because two regions sharing an occlusion boundary need not be merged because they belong to different objects. Humayun *et al.* [4] uses features based on appearance, texture, and flow to train a random forest for finding pixels that are occluded. Since the label here is binary ($\{$occluded, not occluded$\}$), the forest performs regression to produce a probability of occlusion $\mathcal{O}_i$ for pixel i.

Part of our research is to decide which features used by [4] can be discarded in lieu of efficiency without sacrificing a significant amount of precision. We experimented varying combination of features proposed in [4], by taking hints from the feature importance returned by the forest. Although [4] shows improvements in classifying occlusion with the use of multiple flow algorithms, we constrain ourselves to a single flow algorithm [9] in order to reduce computation time.

One set of features our occlusion classifier uses is based on temporal gradients. Since occlusions usually co-occur with flow discontinuities, measuring gradient of the flow vectors is potentially a useful measure:

$$f_{\mathrm{TG},x}(\mathbf{x},z) = \|\nabla \bar{u}_x\|, \quad f_{\mathrm{TG},y}(\mathbf{x},z) = \|\nabla \bar{u}_y\|, \tag{1}$$

where $f_{\mathrm{TG}}(\mathbf{x},z)$ gives the feature value at pixel $\mathbf{x}$ at scale-space level z.

All other features that are in our final classifier are based on flow. Cross-checking how a pixel value changes from I_1 to I_2 when advected by flow u can reveal occlusions wherever brightness constancy breaks down:

$$f_{\mathrm{PC}}(\mathbf{x},z) = |I_1(\mathbf{x},z) - \mathrm{bicubic}(I_2(\mathbf{x}+u(\mathbf{x},z),z))|. \tag{2}$$

By computing flow in both directions, we are able to use two additional sets of features. Given perfect bidirectional flow fields, the path taken by a pixel advected from I_1 to I_2 using $u(\mathbf{x})$ and back using $u'(\mathbf{x}')$ should return the pixel to its original location. This is not true for occluded pixels, as they *flow* to undefined locations. We find this disparity in forward and reverse flow by taking the ℓ_2 distance between the original location and its position after following the pixel's flow $\mathbf{x}' = \mathrm{round}(\mathbf{x}+u(\mathbf{x},z))$,

$$f_{\mathrm{RC}}(\mathbf{x},z) = \|\mathbf{x} - (\mathbf{x}' + u'(\mathbf{x}',z))\|. \tag{3}$$

Following the same argument, the reverse flow direction at non-occluded pixels should be nearly opposite of the forward flow direction, except in cases of errant flow in regions of

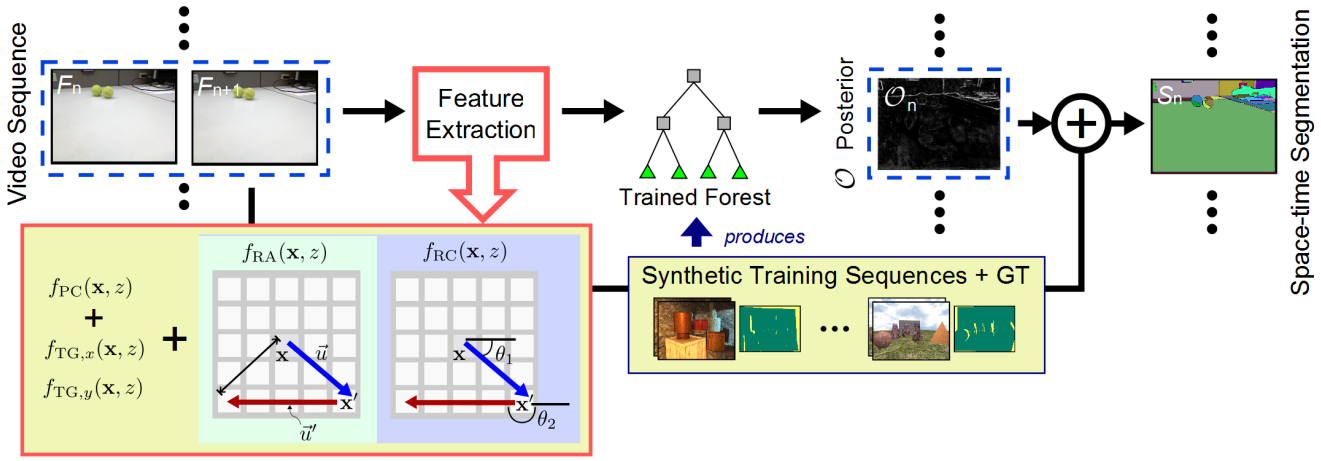


Fig. 2. Shows an overview of our approach - demonstrates how the occlusion posterior probability is computed and fed to the oversegmentation framework.

occlusion. We compute this reverse flow angle consistency as

$$f_{\mathrm{RA}}(\mathbf{x}, z) = |\pi - \arccos\left[u(\mathbf{x}, z) \cdot u'(\mathbf{x}', z)\right]|. \quad (4)$$

Apart from these 4 set of features discussed above, we also experimented with other feature combinations. We give maximum F1-scores in Table I for leave-one-out cases from our occlusion classifier's training set. The F1-score is a measure indicative of combined precision and recall performance (computed as $2PR/(P+R)$). We find our classifier built on just these 4 features gives a good trade-off between accuracy and computation time.

B. Occlusion-aware Oversegmentation

During the oversegmentation stage, Grundmann *et al.* [2] uses the ℓ_2 color distance to create edge weights between pixels both spatially and temporally. These edge weights are given as:

$$w(e_{ij}) = ((l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2)/3 \quad , \quad (5)$$

where each term represents the difference between two pixels i and j for each of the channels in the $l*a*b*$ colorspace. Some of these edge weights that lie across object boundaries are incorrectly assigned a low score due to the objects being of similar color. At these edges, the graph cut algorithm of Felzenszwalb and Huttenlocher [1] will produce false negatives causing the objects to be merged. Because this edge does not appear in the oversegmentation stage, the error will be propagated to every level of the hierarchical segmentation and cannot be recovered. In order to ensure that this edge is created, we use the per-pixel occlusion probabilities of Humayun *et al.* [4] as a distance cue in the oversegmentation. We create an edge weight for every pixel in the spatio-temporal volume according to the equation:

$$w(e_{ij}) = \alpha \cdot |I_i - I_j| \; + \; (1-\alpha) \cdot |\mathcal{O}_i - \mathcal{O}_j| \quad , \quad (6)$$

where the first term represents the ℓ_2 color distance and the second term represents the ℓ_2 occlusion probability distance. Essentially, pairs of pixels with a large occlusion probability difference will be assigned a larger weight and will not be combined while we segment the graph. The opposite is true for pairs of pixels with a small occlusion probability diference; they are assigned a smaller weight and should be combined together. This formulation is a result of the intuition that if occlusions are found between two seperate regions, they must belong to different objects and should not be merged. This information then gets propagated in order to preserve the edge in the hierarchy.

IV. Evaluation

The feature selection process involves determining the features that yield the most accurate occlusion classification of the pixels. We evaluate this data with Maya-generated sequences that have corresponding ground truth occlusion masks in Table I. By quantitatively evaluating the F1 scores from precision-recall curves for each combination of features across all test sequences, we can determine the most accurate combination of features.

The performance of the occlusion-based oversegmentation is compared to the original oversegmentation and to the oversegmentation with depth cues from an RGBD sensor. When using depth in the oversegmentation stage, the edge weight between pixels i and j is computed as

$$w(e_{ij}) = \alpha \cdot |I_i - I_j| \; + \; (1-\alpha) \cdot |D_i - D_j| \quad , \quad (7)$$

where I and D are the RGB and the Depth image. We set $\alpha = 0.5$ for our experiments. It is important to note that if either of the pixels has an unknown depth, we instead use only the ℓ_2 color distance. This function causes the oversegmentation to force edges between objects at different depths.

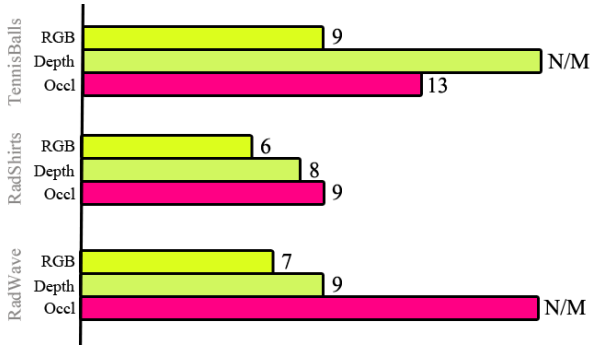


Fig. 3. Level of hierarchies that merge occurs for interesting objects across different sequences.

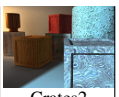
	 Crates1	 Crates2	 Robot	 Sponza1	 Sponza2	 Crates1txtr	 Brickbox1t1	 Brickbox2of	 Mayan1	 Text1
PC+RC	0.107	0.016	0.278	0.477	0.059	0.418	0.571	0.411	0.461	0.358
PC+RC+RA	0.146	0.015	0.266	0.432	0.063	0.474	0.605	0.416	0.477	0.238
PC+TG+RC+RA	0.148	0.023	0.296	0.532	0.120	0.608	0.615	0.488	0.447	0.399
PC+TG+AV+LV+CS+RC+RA	0.180	0.020	0.342	0.525	0.136	0.620	0.603	0.507	0.448	0.469
All Features	**0.189**	0.022	0.314	0.537	0.150	0.597	0.633	0.494	0.400	0.624
All Features + 4 flows	0.149	**0.034**	**0.457**	**0.640**	**0.207**	**0.807**	**0.868**	**0.741**	**0.584**	**0.754**

TABLE I. Leave-one-out F1 scores. Computed by getting the maximum F1 score over the precision-recall graph. The highlighted row is the classifier we eventually selected, whereas bold values indicates the best performing classifier for a given training image pair.

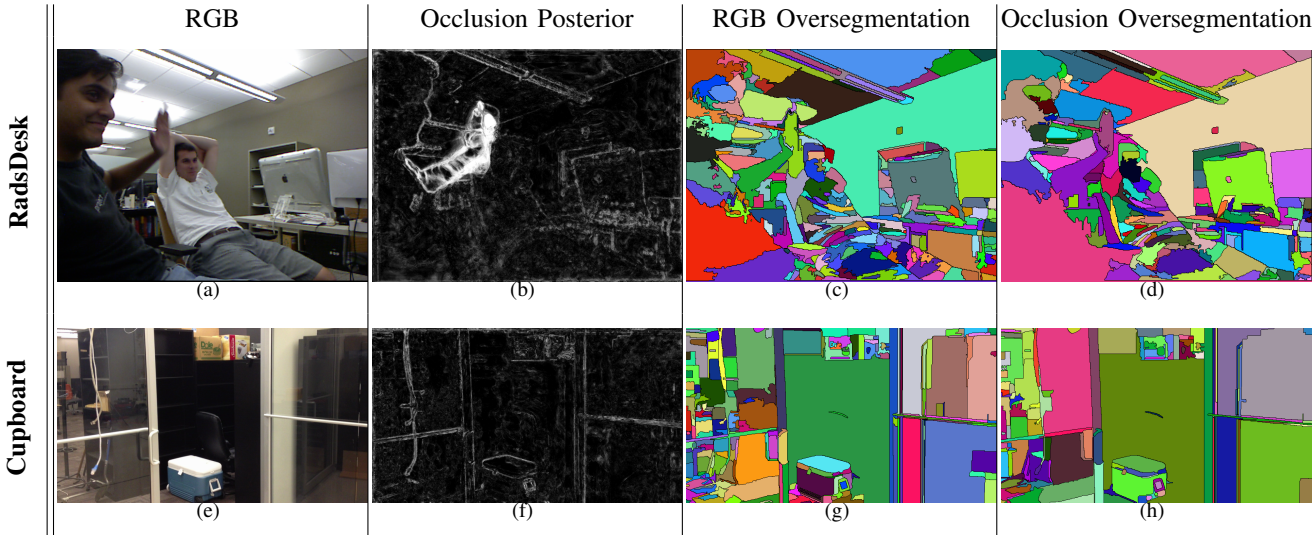


Fig. 4. Qualitative oversegmentation results for the RadsDesk and Cupboard sequence.

The algorithm is tested using real sequences where false negatives arise during the oversegmentation stage. We quantitatively compare the number of false negatives using each of the three schemes in Figure 3.

V. Discussion

In order to evaluate which features should be selected from the random forest, we created Table I. This table shows that the combination of photo-constancy, reverse flow constancy, temporal gradient, and reverse flow angle consistency perform the best at detecting occlusions. We experimented with other features suggested by Humayun *et al.* [4], but noticed that getting more accurate results required computationally expensive features.

The results of the algorithm are shown in Figure 4. A qualitative analysis can be performed by observing locations where two different objects merge during the original oversegmentation, but do not merge in the monocular occlusion based approach. In the RadsDesk sequence a false negative is made in the original oversegmentation that causes the hand to merge with a region that belongs to a different object. The Cupboard sequence clearly shows that our method reduces the number of segments produced by our algorithm. Although our approach at times helps separate different objects, the process is detrimental whenever there are no occlusion cues between two objects (for instance the chair's arm in RadsDesk). Overall, occlusions inferred from an RGB sensor can help dilineate object boundaries to some extent.

We also show that our resulting monocular oversegmentation implementation can often achieve better results in Figure 3. The bars represent the number of hierarchical stages that a true positive edge survives. A possible explanation for the monocular occlusion oversegmentation outperforming the RGBD oversegmentation is that the depth map is often noisy and contains incomplete areas.

This work demonstrates that monocular vision can provide similar cues to that of a depth sensor when it comes to video segmentation. Future work will involve using occlusion information into the hierarchical segmentation step by assigning each spatio-temporal superpixel scores based on the amount of occlusion within the region.

References

[1] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59:167–181, 2004.

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

[3] X. He and A. Yuille. Occlusion boundary detection using pseudo-depth. In *ECCV*, volume 6314 of *LNCS*, pages 539–552. 2010.

[4] A. Humayun, O. Mac Aodha, and G. J. Brostow. Learning to find occlusion regions. In *IEEE CVPR*, pages 2161 –2168, 2011.

[5] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *IEEE ICCV*, pages 508 –515 vol.2, 2001.

[6] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *IEEE CVPR*, pages 3369 –3376, 2011.

[7] A. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection andmid-level reasoning. *IJCV*, 82:325–357, 2009.

[8] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, volume 6315 of *LNCS*, pages 268–281. 2010.

[9] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, 2009.