Enhancing Generic Segmentation with Learned Region Representations - Appendices

A. Parametrizing the hierarchy

Our hierarchical generic segmentation algorithm uses an iterative merging process that creates a segmentation hierarchy, starting from an initial oversegmentation. This hierarchy can be represented efficiently by a UCM [1]. The UCM values should be monotonic with the iteration number. This is indeed the case in [2, 8], for example, where the edge confidence, averaged on the contour segments, is used. However, in our algorithm, the dissimilarity values are not necessarily monotonically increasing with the iteration because they are based on a complex set of features that additionally relies on random samples; see Fig. 1.

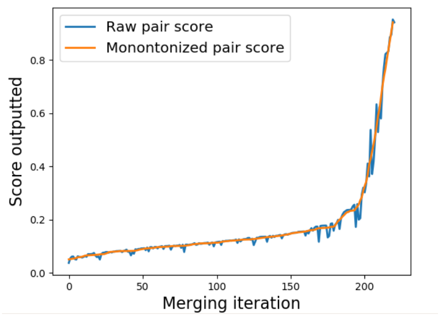

Figure 1: Example of original pair dissimilarities and their monotonized version used to parameterize the hierarchy.

In principle, we could have chosen any monotonic sequence that increases with the iterative process, and even the iteration number itself [4]. The value of the iteration number, however, is not related to the segmentation quality uniformly with respect to the set of diverse images. Therefore, uniform thresholding cannot be applied. The *Pair_Dissim*, on the other hand, has such a meaning regarding the resulting segmentation quality: the probability that the segments belong to different objects. Therefore, we therefore choose a monotonic function that approximates it, and assign a monotonized value for every segmentation. To that end, after performing all the merge iterations and saving all the *Pair_Dissim* values, we smooth the sequence of these values, and then iteratively remove every value that is smaller than the previous unremoved value. This creates a monotonically non-decreasing, but partial sequence, which we complete through linear interpolation. The initial smoothing helps to prevent bias to high values. See an example in Fig. 1. Note that we approximate the *Pair_Dissim* and not the full augmented pair dissimilarity because the latter is not used in all iterations and because, as a combination with *Sil_Score*, it loses its probabilistic interpretation.

B. Segment filtering and sampling

During the pixel-wise representation process used in our work, the output is not completely uniform over each pixel in an image segment, and outliers exist due to small structures and proximity to different objects. Therefore, before using the representation vectors we filter them using the Isolation Forest algorithm [7]. Isolation Forest works by creating several trees from the training data. On each tree, it iteratively splits the training data over a random feature using a random value until a single data point remains on each leaf. Intuitively, compared to inliers, outliers require, on average, less splits to reach a leaf and thus will be reached earlier in the forest. Remarkably, the Isolation Forest algorithm detects outliers even with significant undersampling and yet is efficient both in training and prediction.

This filtering is applied to all segments, whether coming from the initial oversegmentation or from a merging step. For large segments (300 pixels or more), we also sample the remaining pixels to reduce the cost of calculating the pixel pair distances. We show in Appendix C that both the filtering and sampling improve our results. Interestingly, filtering achieves better results when done for merged segment pairs rather than only for the initial oversegmentation.

C. Comparing different choices for the merging function

We experimented with our hierarchical generic segmentation algorithm over several versions of the merging function components. The comparisons are done using the (OIS) F-score for objects and parts on BSDS500 (validation) [9] with [8] for edge detection.

C.1. Re-ranking

Recall that re-ranking starts when the number of segments is $\#s$ or lower, and is carried out for T candidates (in each iteration). Table 1 describes the results with and without the re-ranking, and with different $\#s$ and T values.

Re-ranking noticeably improves F score results. We can also see that starting the process earlier or using more segment pairs did not improve the results.

C.2. Filtering and sampling

We examine the effect of filtering and sampling. We test sampling (denoted as $\#P$) of 300 and 3000 pixels per segment (larger sampling sizes are unfeasible) as well as filtering vs. no filtering. Table 2 describes the results.

Both fewer pixel samples and filtering improve the accuracy, by reducing the chance of selecting pixels that are not typical to the segment.

C.3. Choice of classifiers

We have trained three alternate classifiers that determine the pair dissimilarity: the first is logistic regression with an L1 penalty, without altering the class weights. Training was performed under liblinear [5] with a C value of 1. The second is a multi-layered perceptron with 5 hidden layers of size 96 each. We used the Adam solver [6] with an initial learning rate of 0.01 and logistic activation. The third is a Random Forest Regressor [3] with 20 estimators, a max depth of 15, minimum of 400 samples per leaf and a minimum split of 0.05.

We optimized the parameters of each classifier and used its probability prediction output as the pair dissimilarity (for the Random Forest, we used the average of samples on a leaf). Table 3 describes the results. The best results were obtained with logistic regression and a multi-layer perceptron with the outlined parameters.

	No re-ranking	$\#s = 120$ $T = 4$	$\#s = 240$ $T = 4$	$\#s = 120$ $T = 20$
F_{op}	0.509	0.512	0.511	0.511

Table 1: Comparison of different re-ranking parameters.

	$\#P = 300$ & Filter	$\#P = 3000$ & Filter	$\#P = 300$ & No filter	$\#P = 3000$ & No Filter
F_{op}	0.512	0.508	0.510	0.506

Table 2: Comparison of different filtering and sampling parameters.

	LR	MLP	RF
F_{op}	0.511	0.512	0.48

Table 3: Comparison of different choice of classifiers.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 182–182. IEEE, 2006.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] J. Cousty, L. Najman, Y. Kenmochi, and S. Guimarães. Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. *Journal of Mathematical Imaging and Vision*, 60(4):479–502, 2018.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[8] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[9] J. Pont-Tuset and F. Marques. Supervised evaluation of image segmentation and object proposal techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(7):1465–1478, 2016.