Flexible Multi-modal Graph-Based Segmentation

Willem P. Sanberg[1], Luat Do[1], and Peter H.N. de With[1]

Eindhoven University of Technology, The Netherlands,
w.p.sanberg@tue.nl

Abstract. This paper aims at improving the well-known local variance segmentation method by adding extra signal modi and specific processing steps. As a key contribution, we extend the uni-modal segmentation method to perform multi-modal analysis, such that any number of signal modi available can be incorporated in a very flexible way. We have found that the use of a combined weight of luminance and depth values improves the segmentation score by 6.8%, for a large and challenging multi-modal dataset. Furthermore, we have developed an improved uni-modal texture-segmentation algorithm. This improvement relies on a clever choice of the color space and additional pre- and post-processing steps, by which we have increased the segmentation score on a challenging texture dataset by 2.1%. This gain is mainly preserved when using a different dataset with worse lighting conditions and different scene types.

Keywords: Multi-modal Signal Analysis, RGBD Segmentation, Graphs

1 Introduction

Segmentation of images has been a topic of research for many years in the field of computer vision. Over the years, increasingly complicated features or texture descriptors have been developed to improve the segmentation of interesting objects in two-dimensional (2D) texture images. This segmentation often serves as a basis for semantic labeling in applications such as object recognition, object tracking and event detection in security or sports video, cancer detection, etc.

However, texture segmentation has two fundamental limitations. First, a key issue in texture segmentation is that not all objects have clear, distinctive texture borders, leading to under-segmentation. For example, its is hard to a isolate a white cabinet in front of a white wall. Second, not all texture borders correspond to interesting object contours, depending on the application. For instance, shadows or texture with high variance can lead to over-segmentation.

These problems can be partially addressed by using multi-modal signal analysis. Recently, there has been a growth in multi-modal visual sensor systems that capture depth signals alongside 2D texture images, providing information on the geometry or shape of a scene. This facilitates, inter alia, detecting the white cabinet and neglecting the shadows, based on the change and invariability of the local geometry, respectively. To realize these benefits, new multi-modal

segmentation algorithms are required. For this purpose, we need a segmentation structure in which we can flexibly incorporate texture and shape information. Segmentation graphs meet these demands. A graph consists of vertices and edges that connect these vertices. Vertices can contain RGB or grayscale values [4, 5], feature values [3], or shape information such as depth or orientation with respect to a viewpoint [9]. The edges have one or more weights, reflecting (dis)similarity between the connected vertices. Weights can be customized in many ways [3–5, 9]. Graphs can be defined over multiple scales [5], or provide hierarchical segmentation [5]. This makes graphs flexible to incorporate information from multiple signal modi. Moreover, graph structures can be implemented and segmented efficiently [4]. In the survey of Peng *et al.* [6], several graph-based texture segmentation methods are discussed and compared. From their quantitative analysis, the authors conclude that the method of Felzenszwalb and Huttenlocher [4] (Local Variance Segmentation, LVS) performs best in extracting the key structures in an image, especially when there are many objects present, and has the best average segmentation quality. Furthermore, the complexity of LVS is low, making it an attractive method to build upon.

By extending the LVS method, this paper addresses two related contributions. First, we extend the uni-modal segmentation method (LVS) to perform multi-modal analysis, such that any number of signal modi available can be incorporated in a very flexible way. Second, we experiment with several preprocessing steps for the texture information, which improves the results of the uni-modal LVS algorithm.

The remainder of this paper is divided as follows. The uni-modal baseline method is explained in Section 2 and we present our improvements in Section 3. We evaluate our contributions with several quantitative experiments, described in Section 4, the results of which are presented in Section 5. Our conclusions, a discussion and recommendations for future work are presented in Section 6.

2 LVS for Uni-modal Segmentation

A graph $G = (V, E)$ consists of vertices ($v_i \in V$) that are connected by edges ($e = (v_i, v_j) \in E$) that all have a weight $w(e)$. Segmenting a graph is the problem of finding disjoint subsets S_i such that $\bigcup_i S_i = V$. Ideally, the segments S_i represent areas of interest. In general, the first step of graph-based segmentation is to initialize V and E. The second step consists of selecting one or more seed points. The last step is the merging and labeling process. In the context of 2D texture images, the vertices are pixels that have edges to their neighbors. The weights are typically a measure of dissimilarity of the connected vertices, such as, e.g., the absolute difference in pixel intensity.

Felzenszwalb and Huttenlocher [4] designed their algorithm to result in a segmentation that is neither too coarse nor too fine, using the following definitions: (1) a segmentation is too fine when, among S_i, there are neighboring segments without evidence of a border between them; (2) a segmentation is too coarse if there is evidence of a boundary in at least one S_i. To generate this segmenta-

tion, Felzenszwalb *et al.* define the difference between segments, $\mathrm{Ext_T}(S_1, S_2)$, as the minimal edge weight connecting the segments. Furthermore, they define the variation within a segment, $\mathrm{Int_T}(S_1)$, as the maximum edge weight in the Minimal Spanning Tree (MST) of the segment. Note that we add the subscript T of 'Texture' here for clearer notation later in the paper.

Let us now describe the principal steps of their segmentation algorithm in more detail. First, the graph is initialized with one vertex per pixel, where each vertex has an edge to each of its 8 neighbors. The edges are weighted with the intensity difference of the connected pixels. Furthermore, the segmentation is initialized with one segment per vertex, all having an initial threshold of K_T, which is a user parameter. In the next step, which is the *seeding step*, the edges are sorted to their weight w_T. Sorting is a crucial step, since it guarantees two requirements for proper execution. First, the edge under evaluation is always the connection with the lowest weight between two segments, which is equal to $\mathrm{Ext_T}(S_1, S_2)$. Second, the edge under evaluation is always the connection with the highest weight within the new segment, which is equal to $\mathrm{Int_T}(S_1)$.

The last step, i.e. the *merging step*, executes a boundary check for all edges in order of increasing weight. The boundary check $B_T(S_1, S_2)$ is false when the following inequality holds:

$$\mathrm{Ext_T}(S_1, S_2) \leq \min\left(\mathrm{Int_T}(S_1) + \frac{K_T}{|S_1|}, \mathrm{Int_T}(S_2) + \frac{K_T}{|S_2|}\right), \tag{1}$$

where $|S_i|$ denotes the size of a segment in pixels. If the boundary check is false, the segments will be merged and the threshold of the new segment will be set accordingly. From Eq. (1) we can see that parameter K_T enables small segments to grow. For instance, when a segment contains just one pixel, its internal difference is zero, so that merging is only allowed along edges with a weight of zero. Choosing a large K_T makes it easier for segments to merge. Using an appropriate K_T, regions with high detail can be segmented in small segments, and regions with low detail can be segmented in large segments.

2.1 Shortcomings of LVS

The LVS algorithm suffers from two shortcomings. Firstly, LVS is a region-growing algorithm that merges segments when a single low-weight edge connects them. This can lead to under-segmentation, due to merging of adjacent pixels with similar intensity, but belonging to different objects (leakage). This is particularly difficult to prevent when two objects share a smooth border. Secondly, LVS allows small segments around noise and long thin segments around edges to exist, which leads to over-segmentation. This can happen due to noise in pixel intensities and blurry edges in combination with the seeding strategy of LVS.

3 Extending LVS to Multi-modal Segmentation (MLVS)

Let us now look at the integration of information from multiple signal modi into the LVS segmentation algorithm. Since in multi-modal systems multiple signals

need to be processed simultaneously, it is necessary to adapt the initialization, seeding and merging and labeling steps accordingly. First of all, in the initialization stage of LVS, each vertex $v \in V$ of graph $G(V, E)$ is assigned an extra value per mode in addition to its texture image pixel intensity. As a consequence, each edge obtains one extra weight per mode. For example, an extra weight could be the Euclidian distance between vertex normal vectors or depth values. Then, for the seeding and merging steps of multi-modal signals, we identify two key strategies: (1) defining partial boundary functions for each signal mode, and (2) combining the different weights into a single weight and using a single boundary function. In the following subsections, we will analyze these two strategies in more detail.

3.1 Strategy 1: Partial Boundary Functions

In this approach, we first define a separate partial boundary function B_m for each of the available M modes such that each partial boundary function should reflect the nature of the signal. For example, when a depth signal is available, a boundary check similar to Eq. (1) cannot be applied, since the depth value of a single object varies, in contrast to its texture value. A better alternative is to define a boundary check based on a constant threshold, instead of an adaptively growing approach.

Second, we integrate these separate partial boundary functions in the framework by defining a general boundary function $B(B_1, ..., B_M)$. This boundary function is application-specific and should be designed carefully by the user. For example, if a high boundary detection rate is required and all modi are equally important, B can be implemented with a logical OR-function:

$$B(B_1, ..., B_M) = B_1 \vee B_2 \vee \ldots \vee B_M. \tag{2}$$

However, we are aware of the unequal nature of the partial boundary functions for different modi. Due to this inequality, combining different signal modi into a single boundary function will always be a compromise.

Since edges now have multiple weights, the sorting procedure in the *seeding step* is no longer straight-forward. We have explored two methods for sorting the weights: (1) sorting to the weights of a single mode, neglecting the other weights, or (2) hierarchical sorting by ordering to the weights of one mode, and within that list, sorting to the second mode. Both methods do not guarantee that the edges are handled in order of non-decreasing edge weight Ext(.) in all modi simultaneously, since the modi contain fundamentally different data. However, the strength of combining multi-modal signals is in exploiting partly conflicting information in different modi. The consequence of multi-modal sorting is that the segmentation is no longer unique: each sorting strategy can result in a different segmentation.

After a merge of two segments, the internal differences of the new segment are updated using the weights of the edge under evaluation. This is independent of the sorting method. Therefore, Int(.) only corresponds to the maximum value in the MST for the mode that is sorted.

3.2 Strategy 2: Combining Weights

Our second approach to incorporate multiple weights is to combine them into a single weight w_C:

$$w_C = \sum_{m=1}^{M} \alpha_m \frac{w_m}{A_m}, \tag{3}$$

where M is the number of modi and A is a factor to normalize the weights of a mode to unity. For example, $A = 255$ for the luminance weight. The factor α ($\sum \alpha_m = 1$) expresses the importance of a mode. In our experiments, we have adopted $\alpha_m = 1/M$ for all m.

In the *seeding step*, the edges are sorted to the value of w_C. In the *merging step*, a boundary check analogous to Eq. (1) is applied, specified by:

$$\mathrm{Ext_C}(S_1, S_2) \leq \min\left(\mathrm{Int_C}(S_1) + \frac{K_C}{|S_1|}, \mathrm{Int_C}(S_2) + \frac{K_C}{|S_2|}\right). \tag{4}$$

An advantage of this approach is that adding additional signals only requires normalizing signal values and a difference metric that matches the nature of the signal. In our experiments, we have obtained good results when we use the absolute distance for scalar values and the Euclidian distance for vector values.

3.3 Pre- and Post-processing

Color Space In the LVS algorithm, graph edges are weighted by the difference between pixel intensity values. A commonly used multi-modal capturing device is the Kinect camera. The Kinect RGB sensor uses a Bayer mosaic filter, resulting in a lower quality of the blue and the red signal component. This leads to over-segmentation in the LVS algorithm. To reduce the effect of the Bayer filter, we have adopted the luminance values (Y) of the YUV color space as the signal mode used for texture.

Filtering The pre-processing step should reduce the noise within a segment but not smoothen object borders, which is referred to as *edge-preserving smoothing*. We have compared and evaluated three filtering approaches with an experiment in Section 5.

The first approach is the well-known median filter. It is a nonlinear smoothing filter that is often used to remove speckle noise, but it can also introduce false contours. There are two key parameters: the size of the neighborhood and the number of iterations that should be performed.

The second approach is Bilateral Filtering, which is a filtering technique that convolves an adaptive kernel with the input image. For each pixel under evaluation, the kernel weights decrease with the spatial distance and with the intensity difference (both compared to the kernel center), providing the possibility to smoothen texture variation in areas on both sides of an edge separately.

This enhances the homogeneity of the region and at the same time preserves the edge.

Third, Nonlinear diffusion filtering (NLDF) evolves an image L through increasing scale levels. For edge-preserving smoothing, the authors of [7] and [10] propose to make the diffusion a function of the local image gradient magnitude. We test three of their NLDF approaches, using the implementation of [1].

Post-processing The results of LVS may contain segments of only a few pixels, which are often not of interest. Therefore, if a segment is smaller than $minSegSize$, it is merged with its neighboring segment with the smallest $\mathrm{Ext_T}$.

4 Segmentation Experiments

We have performed several experiments to show that we improve (1) the results of LVS on texture with our adaptations and (2) the segmentation degree further by using our extension to multi-modal signal analysis.

Texture Segmentation We analyze our texture segmentation performance on the challenging Berkely Segmentation Data Set 500 (BSDS)[1], presented in [2] and consisting of RGB images with a wide variation of subjects (animals, landscapes, buildings, people, etc.). The dataset contains 200 training and 200 test images with multiple, manually annotated ground truths. The final score is the average score over all ground truths of an image. We will use this dataset to evaluate our texture segmentation and to train the texture pre-processing settings for the multi-modal signal analysis.

Multi-modal Segmentation NYU Depth dataset V2 (NYU)[2], presented in [8], contains aligned texture/depth/normal-frames from a variety of indoor scenes (kitchens, bedrooms, office spaces, stores, etc.), split into 795 training and 654 test frames. The texture and depth images are captured with a Kinect camera. To create dense depth images, a hole-filling/inpainting algorithm is employed. The normal images are estimated from the individual depth images, by back-projecting depth points to 3D space and performing local plane fitting. For quantitative analysis, this dataset also includes a ground-truth labeling.

Segmentation Metric To compare our results to the ground truth, Arbeláez *et al.* provide several boundary-based and segment-based metrics [2]. We will focus on the score on boundary detection but provide our final scores on all measures for completeness. We measure Recall (R) and Precision (P) scores on edge pixels,

[1] BSDS dataset at `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`

[2] NYU dataset at `http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`

where R is the ratio of true boundary pixels that are detected by the segmentation and P is the ratio of detected boundary pixels that match true boundary pixels. We adopt the evaluation technique from [2], in which boundary pixels match when they are within a distance of 2 pixels of each other.

However, in the NYU dataset, the ground-truth annotation is inaccurate at object boundaries. The influence of inaccurate boundaries is low when the recall of ground-truth regions is measured, as is performed by Silberman *et al.* [8]. In addition, over-segmentation in boundary areas will not considerably degrade the precision score. However, it is our opinion that inaccurate boundaries and over-segmentation in boundary regions should not be ignored. Therefore, we pre-process the ground-truth annotations by thinning borders that are not annotated. Hence, since we have modified the dataset in this aspect, we cannot compare ourselves anymore to experiments of others with the original dataset. Instead, we can compare the mutual results of our different segmentation strategies with the modified dataset.

Parameter Selection and Validation We have performed a parameter-range search for each pre-processing method on the training set of BSDS. After this training step, we select two sets of parameter settings: Θ_{T,F_u}, a setting with the highest unweighted harmonic mean F_u for the improved texture segmentation, and Θ_{T,F_w}, a setting with the highest weighted harmonic mean F_w. We select Θ_{T,F_w} such that it promotes P as a basis for multi-modal segmentation and aim at increasing R by including detected boundaries from other modi.

To validate our training, we run our texture segmentation with Θ_{T,F_u} and Θ_{T,F_w} on the test sets of both BSDS and NYU and compare it to the results of the LVS method, all using the luminance Y as texture input. Next, we execute our two multi-modal segmentation methods on the NYU test set. For this, we use Θ_{T,F_w} and a number of different settings for depth, normal and angle modi.

5 Segmentation Results

Uni-modal Segmentation In our training results, Nonlinear Diffusion Filtering with the Weickert Diffusivity function outperforms the other filtering methods. We have selected Θ_{T,F_u} as the tenth level of the evolution with a contrast factor of 0.5 and $K_T = 300$, and Θ_{T,F_w} as the thirteenth level of the evolution with a contrast factor of 0.5 and $K_T = 1000$.

We have executed both settings on the full test sets of BSDS and NYU to check how generic the settings are. To provide precision-recall curves instead of points, we have performed a range search over K_T while keeping the pre- and post-processing settings constant. As a baseline reference, we also show the score of the LVS method with luminance values Y as texture input. The results are shown in Fig. 1, indicating that our trained settings perform consistently on the BSDS test data. For the NYU test set, the results are less distinctive, but still a slight improvement is achieved. This shows the robustness of our method, as the parameters were not trained on the NYU images and this dataset has worse

lighting conditions and degraded image quality and also contains different types of scenes. More importantly, it has an inaccurate ground truth.

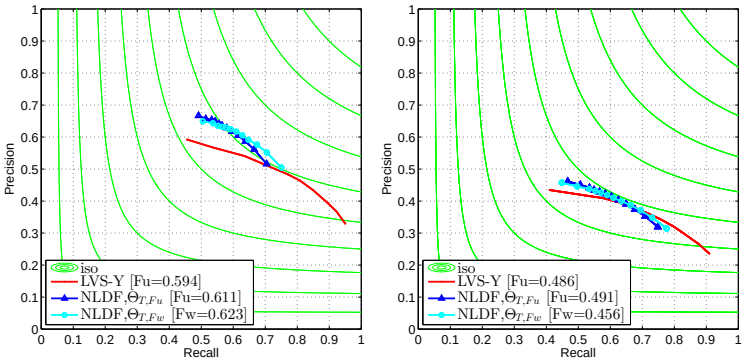


Fig. 1. Boundary detection results on the BSDS (left) and NYU (right) test sets. The settings used are Θ_{T,F_u} and Θ_{T,F_w}, with a range search over K_T to provide a curve.

Table 1. Boundary and region metrics of [2] on the BSDS test set

	Boundary	Region Covering	Region PRI	Region VI
LVS-RGB	0.55	0.46	0.79	3.27
LVS-Y	0.59	0.50	0.75	2.47
Θ_{T,F_u}	0.62	0.49	0.75	2.45

The results of all metrics provided in [2] are presented in Table 1. Using Y values instead of RGB triplets as texture input, increases the scores of LVS significantly. Our pre- and post-processing steps augment the score on the boundary metric further, which is expected since we optimize our algorithm on boundary detection. Simultaneously, our extensions do not degrade the scores on the region-based metrics significantly. This again confirms the robustness of our method.

To illustrate our contributions visually, we show the results on several BSDS test images in Fig. 2. It is clear that our approach gives a cleaner and generally more accurate segmentation than the LVS method for various scenes. The rightmost image, for which we score the lowest F_u, is difficult for this LVS-based method, as the regions have high texture variation and no clear edges.

Multi-modal Segmentation The quantitative results are summarized in Table 2. Overall, our method based on the combined weight clearly outperforms the method of using partial boundary functions. Using a combined weight of luminance and depth values provides the best results.

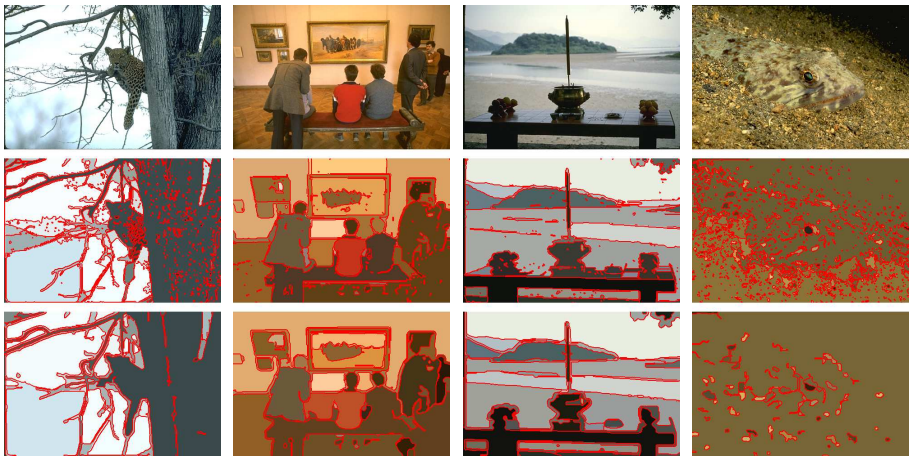


Fig. 2. Images of the BSDS test set. For each image, we show the original image (top), the result of LVS-Y (middle) and our result using Θ_{T,F_u} (bottom). The three images at the left are selected from the top 6 (ranked on F_u), right is the bottom one.

However, several improvements are small or even negative. The cause of this is analyzed with the graphs in Fig. 3 (b), where we plot the increase (or decrease) for F_u, R and P for each individual image and several different multi-modal methods, with the texture segmentation Θ_{T,F_w} as a reference. No image-index correspondences exist, since the images are sorted on their score for each graph separately. By plotting the scores in a sorted fashion, the graphs clearly show that although we generally achieve an increase in recall, we decrease the precision at the same time. This limits the increase in the overall F_u. Since we have designed the multi-modal segmentation to increase the recall of a high-precision texture segmentation, this confirms our expectations. However, it also shows that our multi-modal segmentation methods are not applicable to all images. For example, setting a $maxDist$ in the partial boundary function helps distinguishing foreground from background objects when they have a similar texture. At the same time, it introduces false boundaries on surfaces that are at an angle with respect to the viewpoint. Similarly, normals provide additional object boundaries, but introduce false detections easily, on e.g. wrinkled surfaces. This is especially problematic in the noisy NYU data. Furthermore, it is striking that using more than two signals does not increase the performance further. From the $w_C(Y, A)$ (green) and $w_C(Y, D, A)$ (cyan, dashed) curves in Fig. 3 (b), it can be seen that adding the third mode increases P but decreases R. This indicates that a more sophisticated way of combining the different modi can potentially boost the performance.

We show several practical results on NYU test images with our MLVS in Fig. 4. This figure displays four images with ground-truth overlay. For each image, we present the result of our texture segmentation and compare it to the

result of our multi-modal segmentation. Each image is marked in the graphs of Fig. 3 (b). The leftmost image of Fig. 4 shows an example on which a good increase in both F_u and P is achieved, by using the combined weight of the luminance and angle signals. The highest increase in recall is obtained with the second image, using partial boundary functions on the texture and the depth signals. The third image shows a good performance of the most stable multi-modal approach (using a combined weight of luminance and depth values). The multi-modal segmentation performs worst on the fourth image, mainly due to the normal signal that causes false boundaries on the wrinkled blanket.

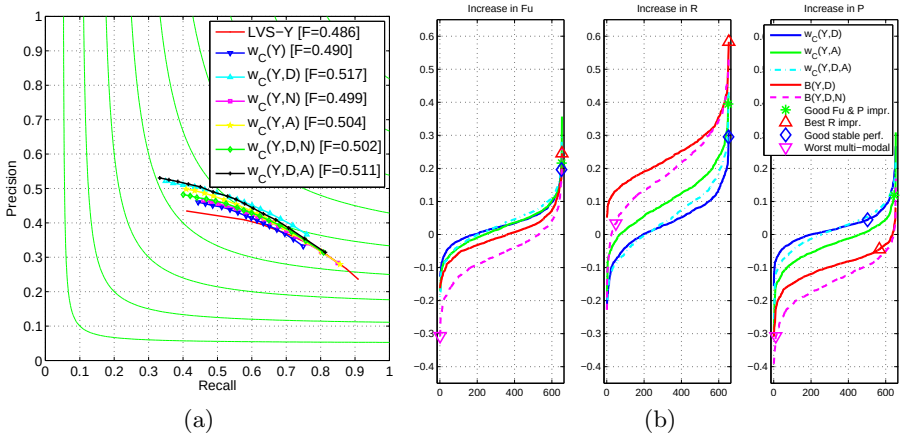
Table 2. Multi-modal Segmentation Results

Method	Key settings	F_u-score	increase
LVS-Y	max F_u on BSDS	0.480	
$B(Y)$	Θ_{T,F_u}	0.490	2.1%
	Θ_{T,F_w}	0.484	0.8%
$B(Y,D)$	$maxDist = 0.01$	0.478	-0.4%
	$maxDist = 0.15$	0.490	2.1%
$w_C(Y,D)$	$D^{-1}; K_c = 2$	0.512	6.8%
$B(Y,N)$	$K_N = 20$	0.440	-8.4%
$w_C(Y,N)$	$K_c = 2$	0.497	3.6%
$w_C(Y,A)$	$K_c = 2$	0.502	4.5%
$B(Y,D,N)$	$maxDist = 0.01;\ K_N = 20$	0.425	-11.5%
	$maxDist = 0.15;\ K_N = 20$	0.439	-8.6%
$w_C(Y,D,N)$	$D^{-1}; K_c = 2$	0.500	4.2%
$w_C(Y,D,A)$	$D^{-1}; K_c = 2$	0.510	6.3%

6 Conclusions, Discussion and Future Work

This paper aims at improving the well-known local variance segmentation method by adding extra signal modi and specific processing steps. To achieve this, we have developed an improved uni-modal texture-segmentation algorithm. With our choice of color space and additional pre- and post-processing steps, we have increased the harmonic mean of recall and precision (F_u) on the BSDS test set from 0.59 to 0.62, based on training with the BSDS training set. The same settings have improved the F_u metric on the NYU test set from 0.480 to 0.484, even though this dataset has worse lighting conditions, degraded image quality and also features different types of scenes.

Our second contribution extends this uni-modal segmentation method to perform multi-modal analysis, by introducing a method that can process any number of signal modi that are available in a flexible way. Based on the quantitative analysis on the NYU dataset, the use of a combined weight of luminance and depth values improves the F_u metric additionally from 0.484 to 0.512.



(a) (b)

Fig. 3. (a): Boundary detection results on the NYU test set, using the combined weight with different modi. (b): Individual increase in F_u (left), Recall (middle) and Precision (right) by using additional signal modi with $K_c = 2$. Values pare sorted for clarity, so no image-index correspondences between graphs exist. The four images of Fig. 4 are marked.

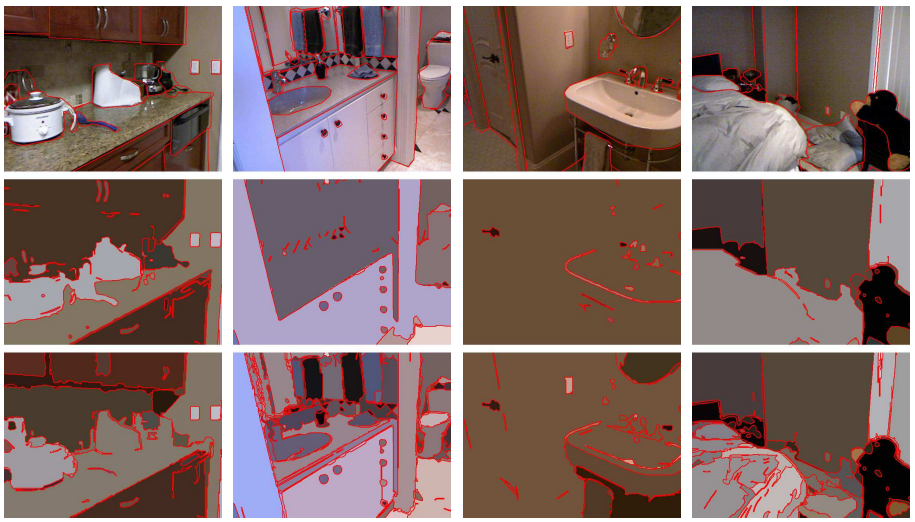


Fig. 4. Images of the NYU test set. For each image, we show the original image with ground truth (top), our texture segmentation result (Θ_{T,F_w}, middle) and a multi-modal result of interest (bottom); from left to right: $w_C(Y, A)$, $B(Y, D)$, $w_C(Y, D)$, $B(Y, D, N)$.

To assess the value of some of the aforementioned results, it should be noted that our uni-modal texture segmentation method does not outperform the state-of-the-art method of Arbeláez *et al.* [2] on the BSDS test set. We have selected an alternative graph-based method as a basis to extend it to multiple signal modi, since the full framework of Arbeláez *et al.* [2] is elaborate and complicated for multi-modal extensions.

We envision several possibilities for further improvement. First, although we have improved the results of LVS with pre- and post-processing steps, we consider that the seeding stage allows further optimization, such that all edges are evaluated with a parallel, more global approach to avoid border isolation. Second, the multi-modal algorithm should better exploit the different characteristics of signal modi. For example, the depth signal should primarily be used for separating foreground from background objects, and normals should mainly be applied to detect object surface boundaries. To this end, the boundary check or mode weight can be enhanced with e.g. the local confidence in a mode.

References

1. Alcantarilla, P., Bartoli, A., Davison, A.: KAZE Features. In: Eur. Conf. on Computer Vision (ECCV). pp. 214–227. Springer Berlin Heidelberg, Fiorenze, Italy (2012)
2. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 33(5), 898–916 (May 2011)
3. Cour, T., Benezit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. In: IEEE comp. soc. Conf. on Computer Vision and Pattern Recognition (CVPR). vol. 2, pp. 1124–1131. IEEE (2005)
4. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. Int. Journal of Computer Vision 59(2), 167–181 (Sep 2004)
5. Kropatsch, W., Haxhimusa, Y., Ion, A.: Multiresolution image segmentations in graph pyramids. Applied Graph Theory in Computer Vision and Pattern Recognition 41(2), 3–41 (2007)
6. Peng, B., Zhang, L., Zhang, D.: A survey of graph theoretical approaches to image segmentation. Pattern Recognition 46(3), 1020–1038 (Mar 2013)
7. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 12(7), 629–639 (Jul 1990)
8. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Eur. Conf. on Computer Vision (ECCV). pp. 746–760. Springer Berlin Heidelberg (2012)
9. Strom, J., Richardson, A., Olson, E.: Graph-based segmentation for colored 3D laser point clouds. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. pp. 2131–2136. IEEE, Taipei, Taiwan (Oct 2010)
10. Weickert, J.: Efficient image segmentation using partial differential equations and morphology. Pattern Recognition 34(9), 1813–1824 (1998)