A Simplified Texture Gradient Method for Improved Image Segmentation

Qi Wang & M. W. Spratling

Department of Informatics, King's College London, London, WC2R 2LS

qi.1.wang@kcl.ac.uk Michael.Spratling@kcl.ac.uk

Abstract Inspired by the Probability of boundary (Pb) algorithm, a simplified texture gradient method has been developed to locate texture boundaries within gray-scale images. Despite considerable simplification, the proposed algorithm's ability to locate texture boundaries is comparable with Pb's texture-boundary method. The proposed texture gradient method is also integrated with a biologically-inspired model, to enable boundaries defined by discontinuities in both intensity and texture to be located. The combined algorithm outperforms the current state-of-art image segmentation method (Pb) when this method is also restricted to using only local cues of intensity and texture at a single-scale.

Keywords *Cue Integration, Edge Detection, Image Segmentation, Texture Segmentation*

1. Introduction

Image segmentation is a fundamental step in many image analysis or high-level computer vision applications. Image segmentation is typically achieved by locating discontinuities in the properties of adjacent image regions. For example, discontinuities in intensity, colour, or texture. This article is concerned with locating texture discontinuities.

Texture has been utilized by many previously proposed methods to locate boundaries within images. Jain et al. [1] filtered images using a bank of Gabor filters, utilized a Gaussian window to measure texture features in the filtered image, and adopted a square-error clustering algorithm to integrate texture features and make segmentations. Ojala et al. [2] utilized distributions of local binary patterns and pattern contrast to evaluate the similarity of neighboring image regions using G statistics to compare distributions. Will et al. [3] defined textures through the statistical distribution of Gabor filters responses, and the edges of textures were determined as pixels with equal posteriori probability between two classes of textures. Scarpa et al. [4] devised the Hierarchical Multiple Markov Chain algorithm (H-MMC), which employed a Markov chain to characterize texture information through their spatial interaction with neighboring regions, and also created the Texture Fragmentation and Reconstruction algorithm to perform unsupervised texture segmentation on the H-MMC.

Methods based on local discontinuities in texture alone are unlikely to detect all boundaries in all categories of image. Therefore, in recent years, several methods tried to rely on local discontinuities in multiple properties to locate boundaries within an image. For instance, Martin et al. [5] developed the Probability of Boundary algorithm (Pb), which is able to combine local discontinuities in brightness, texture and color within an image to perform image segmentation. Each cue is processed in a different way. Texture features are processed by first applying the K-Means algorithm to cluster filtered image responses to yield cluster centers called textons. Each pixel is then assigned a label corresponding to its closest texton. The difference in the number of pixels with the same texton label between adjacent regions summed over all possible texton labels is then used to localize texture boundaries. Ren [6] and Maire et al. [7] found that the combination of some cues across different scales can improve performance of the Pb algorithm. Maire et al. [7] also proposed the globalization of probability of boundary algorithm (gPb), which integrates multiple local cues using the Normalized-cuts [8] technique.

In this article, a simplified version of the Pb algorithm's texture gradient method [5] is developed to locate boundaries within images. The performance of this algorithm at locating texture boundaries is compared to the Pb algorithm using the BSDS300 benchmark [9]. Because the proposed method for locating boundaries is based on local discontinuities in texture alone, it cannot detect boundaries defined by intensity discontinuities. Spratling [10] has developed the predictive coding/bias competition (PC/BC) model of V1 that can be

used to locate boundaries defined by intensity discontinuities within an image. It has been shown that the PC/BC model of V1 outperforms the Probability of boundary algorithm [5] when this algorithm is also restricted to using local discontinuities in intensity (brightness) at single-scale in gray-scale images. Hence, to better evaluate the effectiveness of the proposed method for locating texture boundaries within gray-scale images, boundaries defined by intensity discontinuities derived from the PC/BC model of V1 are integrated with boundaries defined by discontinuities in texture derived from the proposed method. This combined image segmentation result enables boundaries defined by both of these local discontinuities to be located. Evaluation results indicate that the performance of the proposed, considerably simplified, texture gradient method is comparable with the texture gradient method used in Probability of Boundary algorithm and the performance of the combined algorithm even slightly outperforms the Pb algorithm, when this method is also restricted to using same local cues (brightness+texture) within gray-scale images. Combing the proposed method with the brightness gradient method used in the Pb algorithm, results in performance that is the same as the original Pb algorithm. Hence, the proposed simplifications do not affect performance and the outperformance of the combined algorithm is due to the better local intensity boundaries derived from the PC/BC model of V1.

Because the proposed texture segmentation method is inspired by the texture gradient method used in the Probability of Boundary algorithm [5], section 2 first reviews this algorithm. Afterwards, section 3 describes the proposed texture gradient method in detail, highlighting the differences compared with the Pb algorithm and the complexity advantage of each difference. This section also details how the proposed algorithm is combined with the PC/BC model of V1. The results are given in section 4.

2. The Texture Gradient Method Used in the Probability of Boundary Algorithm

The texture Gradient method used in the Probability of Boundary Algorithm [5] is introduced step-by-step in the following paragraphs.

Step 1: A filter bank is defined which contains both even-symmetric and odd-symmetric filters (second order derivative of Gaussians and their Hilbert transforms). These filters are defined at two scales (with a standard deviation of 1 and $\sqrt{2}$ pixels) and at six evenly spaced orientations. The sizes of these filters at the two scales are 13×13 and 19×19 pixels respectively. This filter bank is shown in figure 1.

Step 2: The original image is convolved with the filter bank to obtain a set of 24 response maps. These response maps are clustered by applying the K-Means algorithm. Each cluster center is called a texton. Therefore, each pixel within an image can be assigned its closest texton. In other words, each pixel within an image is represented by the label of its closest cluster center for further computation. In fact, in order to derive better image segmentation results, the Pb algorithm does not apply the K-Means clustering on each new image, but firstly records all filter bank responses from all training images. The K-Means clustering is applied to all responses from all images in the training set to generate 64 universal textons. When processing each new image, each pixel is assigned the label of the closest universal texton. The texton labels are represented using 64 binary maps.

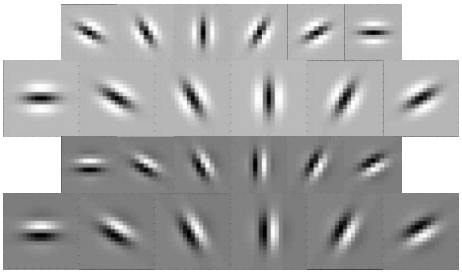


Figure 1 Filter Bank for the Texture Gradient Method Used in the Pb Algorithm

Step 3: A second set of filters are defined which consist of two uniform half discs at eight evenly spaced orientations. All the values in the two half discs are the same, and the sum of values have been normalized to one. The size of each half disc is 23×23 pixels. These half disc filters are shown in figure 2.

Figure 2 Half Disc Filters Defined in the Pb Algorithm

Step 4: The half disc filters are convolved with the 64 binary texton label maps in order to count the number of pixels with the same label within each semi-circular region. The difference in the number of pixels with the same texton label between each half of one disc at one of eight orientations is then computed. 64 binary label maps yield 64 differences at each orientation. The sum of these 64 differences is

defined as the texture gradient at one orientation. There are thus eight texture gradients, one at each orientation generated in this step. The Pb algorithm utilizes Savitzky-Golay filter to smooth these eight texture gradients, which is a kind of polynomial regression on these texture gradients.

Step 5: Texture gradients have been computed for eight orientations at each pixel. In the final step, only the maximum texture gradient is kept at each pixel across the eight orientations. Four examples of the maximum texture gradient images are shown in figure 5. Afterwards, non-maximum suppression [11] is applied to thin the texture boundaries. These thin texture boundaries derived from the maximum texture gradient are the final texture boundary result.

3. Method

3.1 Proposed Texture Gradient Method

Each step of the proposed texture gradient method is described in the following paragraphs, and the differences with the Pb algorithm's texture gradient method are highlighted. To quantitatively evaluate the effect of these differences on the computational complexity of the proposed method, the computation time of each step is compared using a computer with an Intel® Core™ i7-3770 CPU@3.40GHZ CPU. The computation time for these two different methods is the one when tested on 100 test images from the BSDS300 dataset [9], which is shown in table 1.

Step 1: A filter bank is defined which comprises both odd-symmetric and even-symmetric filters (first and second order derivative of Gaussians). These filters are defined at one scale (with a standard deviation of σ_fb=0.3 pixels) and at eight evenly spaced orientations (FBORIENT=8). The sizes of these filters are 7×7 pixels.

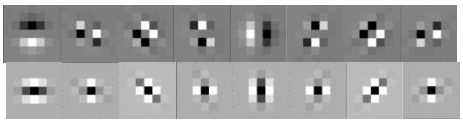


Figure 3 Filter Bank of the Proposed Texture Gradient Method

Step 2: As for the Pb algorithm, the proposed method convolves the original image with the filters in the filter bank to yield a set of 16 response maps.

$$MAP_i = |I \star F_i| \quad (1)$$

where I is the original image, F_i is the i_{th} filter in the filter bank defined in step 1, $\star$ is the convolution operation, and MAP_i is the i_{th} response map to the i_{th} filter.

The number and size of the filters determines the computation time when they are convolved with the same input image. The proposed method using 16 filters of 7×7 pixels has a significant advantage over the Pb algorithm with 24 filters of 13×13 and 19×19 pixels. The comparison of computation time is shown in table 1.

As previously mentioned, step 2 of the Pb algorithm can be implemented either by obtaining universal textons from training images, or by applying the K-Means clustering algorithm to generate textons for each input image individually. The training can be done off-line and the time taken will depend on the size of training image set. The computation time for applying the K-Means clustering algorithm off-line and on-line is shown in table 1. In contrast to the Pb algorithm, the proposed method does not apply the K-Means clustering algorithm, instead texture gradients are calculated directly from the filter response maps. Hence, there is an additional computation time for Pb compared to the proposed algorithm.

Step 3: A second set of filters are defined which consists of two half circular-symmetric filters (a normalized Gaussian filter) at eight evenly spaced orientations (TGORIENT=8). The standard deviation of Gaussian filter is σG=6 pixels. The size of each half Gaussian filter is 31×31 pixels. These filters are shown in figure 4. In contrast to the Pb algorithm, employing filters that are half Gaussians instead of half uniform discs avoids the need to use Savitzky-Golay filter smoothing that is used in the step 4 of the Pb algorithm, and also provides a better trade-off between comparing large regions of the image and localizing boundaries.

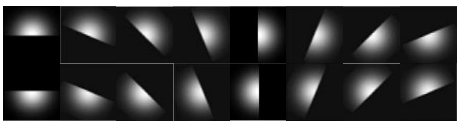


Figure 4 Half Gaussian Filters Defined in the Proposed Method

Step 4: The half Gaussian filters are directly convolved with the 16 response maps to calculate the sum of filtered responses within each Gaussian semi-circular region. The differences between sums of filtered responses within each half of one Gaussian circular region at one of eight orientations are then computed. 16 response maps generate 16 differences at each orientation. The sum of these 16 differences is defined as the texture gradient at one orientation.

$$TG(tgL_i, tgR_i) = WT \sum_{i=1}^{16} \frac{(tgL_i - tgR_i)^2}{tgL_i + tgR_i}$$

Where $\begin{cases} tgL_i = MAP_i \star gfL \\ tgR_i = MAP_i \star gfR \end{cases}$ (2)

where MAP_i is the i_{th} response map to the i_{th} filter in the filter bank, $\star$ is convolution operation, gfL or gfR is a half Gaussian filter at each orientation, tgL_i and tgR_i compute the sum of responses from the i_{th} response maps at each pixel within each half Gaussian region, WT determines the range of texture gradient values, and is used in the combined algorithm in section 3.2, its value was determined by trial and error, and $TG(tgL_i, tgR_i)$ is the texture gradient at each orientation.

Because the size of half Gaussian filter in the proposed method (31×31 pixels) is bigger than the size of half disc in the Pb algorithm (23×23 pixels), there is not a clear complexity advantage at this step for the proposed method. However, the proposed method uses a smaller number of input maps, 16, compared to the 64 label maps used by the Pb algorithm. The computation time of the texture gradient operation for the proposed method and the Pb algorithm is shown in table 1.

Step 5: Eight texture gradients at eight orientations have been computed at each pixel. The proposed method keeps the maximum texture gradient at each pixel across eight orientations. Four original images from BSDS300 dataset [9] with their maximum texture gradients are shown in figure 5. As with the Pb algorithm, non-maximum suppression [11] is also applied to thin the texture boundaries. These thinned texture boundaries are the final texture boundary result derived from the proposed method.

In summary, the main differences between the proposed texture gradient method and the texture gradient method used in the probability of boundary algorithm [5] are in the following three aspects.

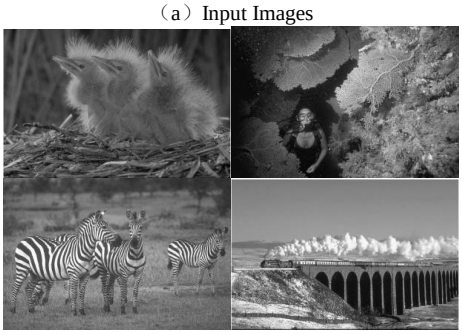
- **First, the filter bank which is convolved with the original images.** The proposed method defines odd-symmetric and even-symmetric filters at only one scale at eight evenly spaced orientations; in contrast, both even-symmetric and odd-symmetric filters at two scales at six evenly spaced orientations are defined in the Pb algorithm. The proposed method has the complexity advantage at this stage because of the significantly smaller number and size of the filters in the filter bank.
- **Second, the sources that are used to compute the texture gradient.** The proposed method directly uses the absolute values of the filter outputs to compute texture gradients instead of employing (universal) texton labels. The proposed method has a complexity advantage at this step, because of the significantly additional computation time of applying the K-Means clustering algorithm (either off-line or on-line) and assigning a texton label to each pixel.
- **Third, the second set of filters which are defined to compute the sum of sources within semi-circular regions, and hence to compute the texture gradient.** The proposed method defines filters that are half Gaussians instead of filters that are uniform half discs. Despite the half Gaussian filters being larger, the proposed method still has the complexity advantage at this step due to the significant smaller number of input maps with which the filter is convolved.

Step(s)	Method(s)	Computation time (seconds)
Convolution operation at step 2	Proposed	3.15
	Pb [5]	17.76
Obtaining (universal) texton and assigning texton label at step 2	Proposed	0
	Pb (off-line)	52.81
	Pb (on-line)	433.44
Texture gradient operation at step 4	Proposed	303.68
	Pb	415.40
Overall steps	Proposed	338.63
	Pb (off-line)	538.32
	Pb (on-line)	928.35

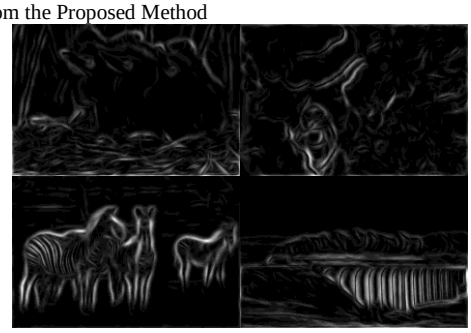
Table 1 the Computation Time of Different step for the Proposed Method and the Texture Gradient Method Used in the Pb Algorithm

From the differences listed above it can be seen that the proposed method is considerably simpler than the texture gradient method used in the Pb algorithm. The overall computation time for the proposed simplified texture gradient method and the texture gradient method used in the Pb algorithm is shown in table 1, which indicates the proposed method is approximate 2 or 6 seconds per image faster than the texture gradient method used in the Pb algorithm which is implemented by applying the K-Means clustering algorithm off-line or on-line. Although the computation time for the Pb algorithm may become shorter if the Pb algorithm is implemented by an optimized method, the reported computation time for the Pb algorithm is obtained using the code provided by the authors of that algorithm [5].

（a）Input Images



（b）Maximum Texture Gradient at each pixel Derived from the Proposed Method



（c）Maximum Texture Gradient at each pixel Derived from the Pb Algorithm

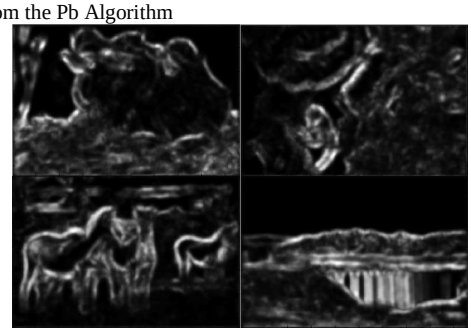


Figure 5 Four Original Images from BSDS300 dataset with Their Maximum Texture Gradients at Each Pixel Generated from the Proposed Method and the Pb Algorithm

Despite these simplifications, the proposed method is still reasonable and effective. To locate texture boundaries, an algorithm needs to firstly represent the same type of texture within an image, and then work out which pixels are located at boundaries between two different types of texture. The Pb algorithm utilizes textons, which are the cluster center of filtered responses across multiple orientations, to represent the same type of texture. In other words, pixels that have similar filtered responses across multiple orientations within an image are supposed to belong to the same type of texture. In comparison, the proposed algorithm directly uses filtered responses at one orientation within the image to represent the type of texture. This works because in circumstances where a smooth (non-textured surface) neighbours a non-smooth (textured area), the proposed method will generate weak responses at all orientations on one side of the boundary and strong responses at one orientation at least on the other side of the boundary. Second, in circumstances where two textured areas meet, different types of texture have different dominant orientations, and the proposed method will generate strong filtered responses at two different orientations to represent the two types of texture.

Both the texture gradient method used in the Pb algorithm and the proposed simplified method are consistent with the mechanism believed to operate in the brain to locate boundaries defined by texture discontinuities. This mechanism can be summarized by a three stage Linear-Nonlinear-Linear (LNL) model [12]. At the first stage, linear filtering is used to locate textures. The first two steps of the Pb algorithm and the proposed method also apply linear filters to the image. At the second stage, filtered responses are processed in a non-linear way to represent different types of texture. The Pb algorithm applies K-Means clustering to generate textons, and the proposed method directly uses the absolute value of the filtered responses. At the third stage, a linear filter is used to locate boundaries between different type of texture. The Pb algorithm utilizes uniform half discs and the proposed method uses half Gaussian filters as the linear filter to compute texture gradient.

3.2 Combination with Boundaries Defined by Intensity Discontinuities Derived from the PC/BC Model of V1

The following three modifications are required to integrate the boundaries derived from the proposed model with the boundaries derived from the PC/BC model of V1 [10].

The absolute values of the filtered responses which are derived from the step 2 of the proposed method are the intensity discontinuities in quite small regions within an input image. The proposed method utilizes these responses to represent the type of texture and then compute texture gradient to locate texture boundaries. Therefore, it is unavoidable that parts of these responses do not represent texture but the boundaries defined by local intensity discontinuities. Using these parts of the filtered responses in the texture gradient method makes these local intensity boundaries become fuzzy. In order to avoid these fuzzy local intensity boundaries located by the proposed method contaminating the boundaries derived from the PC/BC model of V1, the boundaries derived from the PC/BC model of V1 are subtracted from the 16 response maps generated in step 2.

$$MAP_i \leftarrow |MAP_i - V1R| \quad (3)$$

where MAP_i is the ith of 16 response maps, V1R is the boundary responses derived from the

PC/BC model of V1 based on local intensity discontinuities.

Second, the range of values for the texture gradient will influence the final combined algorithm's boundary result. Hence, WT in step 4 is set to a value of two, which is determined by the trials and errors.

Third, in step 5, after the maximum texture gradient is found at each pixel across the eight orientations, the maximum texture gradient is added to the boundaries derived from the PC/BC model of V1 [10] to yield a combined map. Afterwards, non-maximum suppression [11] is applied to thin the boundaries in this combined map. The combined boundaries are the final result and contain boundaries defined by local discontinuities in both texture and intensity.

4. Result

The BSDS300 dataset [9] is a standard benchmark for evaluating the performance of image segmentation algorithms. The BSDS300 dataset contains 100 test images of size 481-by-321 pixels. They are all natural images of people, animals, plants, buildings, man-made objects and some natural scenes. In figure 7, several results of image segmentation on two images from BSDS300 dataset [9] are shown. These include results derived from human observers, results derived from the PC/BC model of V1, results obtained by combining the proposed method with the PC/BC model of V1, and results obtained by the combined brightness and texture gradient method used in the Probability of Boundary algorithm.

To quantitatively evaluate the performance of an image segmentation algorithm, the BSDS300 benchmark employs the F-score. The F-score is defined as 2PR/(P+R) where P is the precision and R is the recall obtained when comparing the algorithm's segmentation with segmentations produced by humans. In practice, the F-score varies from 0.41, which is the performance obtained by randomly defining pixels as boundaries, to 0.79, which is the performance obtained by a human observer compared with other human observers. Figure 6 shows F-scores with the precision-recall curves obtained by different algorithms. Table 2 reports F-scores obtained by state of the art image segmentation methods.

Although the proposed texture gradient method is a considerable simplification compared to the texture gradient method used in the Pb algorithm, the two methods generate the same F-score 0.58 [5]. Furthermore, the performance obtained by combining the brightness gradient method used in the Pb algorithm with the proposed method is the same as the performance obtained by combining the brightness and texture gradient methods used in the Pb algorithm (F-score 0.63). Hence, the proposed simplifications do not affect performance. The performance obtained by integrating the proposed method with the PC/BC model of V1 (F-score 0.64) is slightly better than the performance obtained by the Pb algorithm (F-score 0.63) [5]. The reason that the combined algorithm, outperforms the Pb algorithm is that the PC/BC model of V1 produces better performance in detecting boundaries defined by local intensity discontinuities (F-score 0.61) [10] than the brightness gradient method used in the Pb algorithm (F-score 0.60) [5].

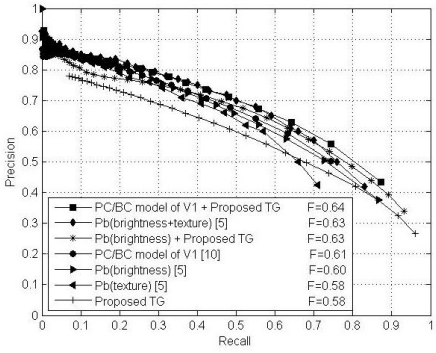


Figure 6 Performances of Different Image Segmentation Algorithms Evaluated by the precision-recall curves & F-score from the BSDS300 Benchmark. These Algorithms only Use Single-scale Local Cues Within Gray-scale Images.

The current state-of-art algorithms for image segmentation are SCG [13] and gPb_ucm [14]. The gPb_ucm [14] is an extended version of the Pb algorithm [5]. These two algorithms have the same F-score of 0.71, which is significantly higher than that of the proposed method integrated with the brightness method used in the Pb algorithm (0.63). The SCG [13] and the gPb-ucm [14] both employ multiple image features and image scales. These multiple image features include local discontinuities in color, intensity, texture and also global information of these features within an image. In contrast, the integrated algorithm only employs local intensity texture discontinuities at a single-scale. Therefore, the computation time of the proposed method integrated with the brightness gradient method used in the Pb algorithm is significantly lower than that of the current state-of-art algorithms when tested on 100 test images from the BSDS300 dataset [9], which is shown in table 2. The computation times for the current state-of-art algorithms were obtained using the code provided by the authors of those algorithms [13][14].

Experiments were conducted to assess the affects of altering the values of six key parameters in the proposed algorithm. These results are shown in table 3. It can be seen that the algorithm is robust to changes in these parameters, with the F-score remaining 0.63 or 0.64 for all combinations of parameters tested. Hence, the good performance of the combined algorithm is not the result of using parameters with a set of special values. In other words, the proposed texture gradient method combined with the PC/BC model of V1 is robust.

Method	F-score	Computation time (seconds)
Proposed TG + Pb (brightness)	0.63	440.39
Pb (brightness+texture) [5]	0.63	664.72
SCG [13]	0.71	11688.64
gPb_ucm [14]	0.71	5858.34

Table 2 the Computation Time and F-score of Different Methods

5. Conclusion

In this paper, inspired by the texture gradient method used in the Pb algorithm [5], a simplified texture gradient method based on local discontinuities in textures is proposed to locate texture boundaries within gray-scale images. The proposed method is a considerable simplification compared to the Pb algorithm. First, the proposed method uses odd-symmetric and even-symmetric filters at only one scale; rather than at two scales. Second, the proposed method directly employ the filter responses to compute the texture gradient rather than using labels of universal textons generated by applying the K-Means clustering algorithm to all filter responses from all images in the training set as used in the Pb algorithm. Third, uniform half disc filters are used in the Pb algorithm to compute texture gradient; instead, the proposed method employed half Gaussian filters in order to avoid the need to use Savitzky-Golay filter smoothing on texture gradients and bring a better trade-off between using large regions to calculate the texture gradient while maintaining good localization of boundaries.

Despite the proposed simplifications, the performance of the proposed method in detecting texture boundaries is comparable with the texture gradient method used in the Pb algorithm when tested with the BSDS300 benchmark [9]. Furthermore, a combination of the proposed texture gradient method with the PC/BC model of V1 [10] enables boundaries defined by local discontinuities in both intensity and texture to be located. When tested with the BSDS300 benchmark, the combined algorithm slightly outperforms the current state-of-art image segmentation method (the Pb algorithm) when this method is also restricted to using only local discontinuities in intensity and texture at a single-scale within gray-scale images.

Reference

1. K. Jain and F. Farrokhnia, “Unsupervised Texture Segmentation Using Gabor Filters,”in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, 1990.
2. T.Ojala and M.Pietikainen, “Unsupervised Texture Segmentation Using Feature Distributions,”*Pattern Recognition*, vol.32, pp.477-486, March.1999
3. S. Will, L. Hermes, and J. M. Buhmann, “On Learning Texture Edge Detectors,”in *Proc. Int. Conf.Image Processing*, 2000
4. G. Scarpa, R. Gaetano, M. Haindl, and J. Zerubia, “Hierarchical Multiple Markov Chain Model for Unsupervised Texture Segmentations,”*IEEE Trans. Image. Process*, vol. 18, no. 8, pp. 1830-1843, May. 2009.
5. D. R. Martin, C.C.Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color and texture cues,”*IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May. 2004.
6. X.Ren, “Multi-scale improves boundary detection in natural images,”in *Proc. European Conf.Computer. Vision*, 2008.
7. M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, “Using contours to detect and localize junctions in natural image”,*Comput. Vis. Pattern. Recognition*, pp. 1-8, June 2008.
8. J. Shi, and J. Malik, ”Normalized cuts and image segmentations”, *IEEE Tran. Pattern. Analysis and Machine Intelligence*, vol. 22, pp. 888-905, Aug. 2000.
9. D.Martin, C.Fowlkes, D.Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, “in *Proc. Int. Conf. Computer. Vision*, vol. 2, pp. 416–423, Jul. 2001.
10. M. W. Spratling, “Image segmentation using a sparse coding model of cortical area V1,”*IEEE Trans. Image. Process*, vol. 22, no. 4, pp. 1631-1643, Feb. 2013.
11. J. Canny, “A computational approach to edge detection,”*IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
12. M. S. Landy, and N. Graham, ”Visual perception of texture.”In: *The visual neurosciences (Chalupa LM, Werner JS, eds),*” pp 1106–1118. Cambridge, MA: MIT, 2004. .
13. X. Ren and L. Bo, “Discriminative trained sparse code gradients for contour detection”, *In*

Advances in neural information processing system, pp. 584-592, 2012.
14. P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation", *IEEE Tran. Pattern. Analy. Mach. Intell.*, vol. 33, pp. 898-916. 2011.

Parameter	Description	Standard Value	Altered Value	F-score
σ_{fb}	Standard deviation of second order derivatives of Gaussian filters.	0.3	0.6 0.1	0.64 0.63
EL	Aspect ratio of Gaussian envelop for second order derivatives of Gaussian filters	3	5 1	0.64 0.64
FBORIENT	Number of evenly spaced orientations for first and second order derivatives of Gaussian filters	8	12 6	0.64 0.64
σ_G	Standard deviation of half Gaussian filters used to compute texture gradient	6	12 3	0.63 0.63
TGORIENT	Number of evenly spaced orientations of half Gaussian filters	8	12 6	0.64 0.64
WT	Range of value of texture gradient in the proposed method	2	4 1	0.63 0.63

Table 3 Effect of Altering the Value of Six Key Parameters Used in the Proposed Algorithm on the F-score from the BSDS300 Benchmark

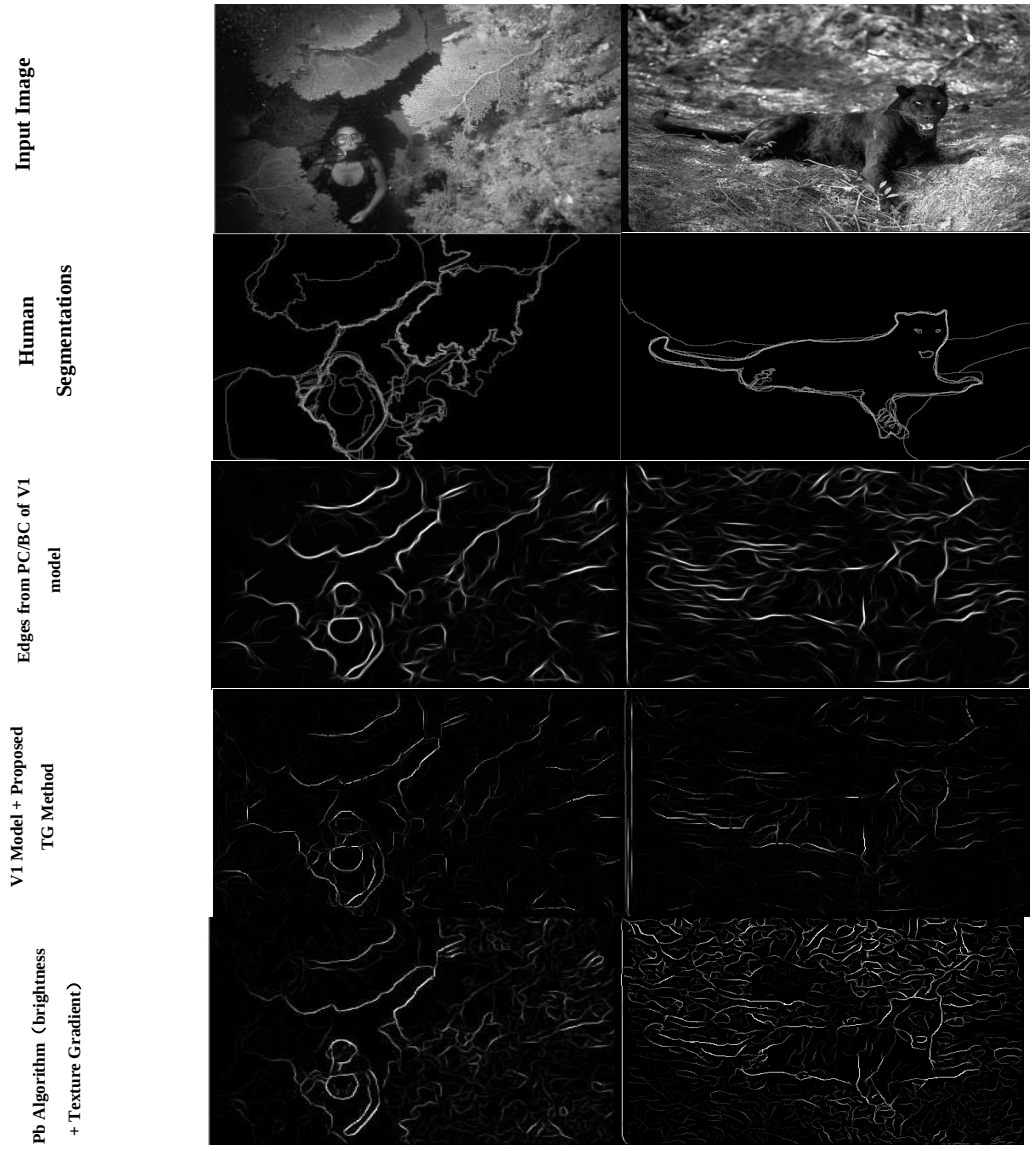


Figure 7 Two Images from BSDS300 Dataset [9] and Their Image Segmentation Results Obtained by Different Algorithm