5-2017

Image Segmentation Using De-Textured Images

Yaswanth Kodavali

Image Segmentation Using De-Textured Images

By

Yaswanth Kodavali

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

In

Computer Science

Approved:

Xiaojun Qi, Ph.D. Major Professor	Curtis Dyreson, Ph.D. Committee Member

Kyumin Lee, Ph.D.
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
2017

ABSTRACT

Image segmentation is one of the fundamental problems in computer vision. The outputs of segmentation are used to extract regions of interest and carry out identification or classification tasks. For these tasks to be reliable, segmentation has to be made more reliable. Although there are exceptionally well-built algorithms available today, they perform poorly in many instances by producing over-merged (combining many unrelated objects) or under-merged (one object appeared as many) results. This leads to far fewer or more segments than expected. Such problems primarily arise due to varying textures within a single object and/or common textures near borders of adjacent objects. The main goal of this report is to pre-process the input images to nullify the effects of such textures. We introduce a pre-processing technique that prepares the input images, before the application of segmentation algorithms. This technique has demonstrated an enhancement in quality of the segments produced. This pre-processing method is called the de-texturing method. We experimented with the effect of the proposed de-texturing method on two existing segmentation methods, namely, the Statistical Region Merging (SRM) method [1] and a k-means-based method as suggested in [2].

ACKNOWLEDGEMENTS

I would like to thank my major professor Dr. Qi for her continuous guidance throughout the duration of the project.

I would like to thank my parents, brother, friends and girlfriend for their continued moral support during my Masters.

A special thanks to my Grandparents for helping me in realizing my dreams.

CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

1. Introduction

Image segmentation basically divides images into groups of pixels, based on shared attributes among the pixels. Different computer vision tasks require different levels of segmentation. While certain tasks require segmentation of one specific type/class of objects, others require segments of all possible objects that can be extracted from the image. Depending upon the requirement, a segmentation method either tries to search for certain features throughout the image and group their matching portions as foreground and the remaining as background, or, tries to cluster the pixels with matching local features into multiple segments. As each task has its own segmentation requirement, this makes image segmentation an important precursor in most of the computer vision applications.

Any object in a given image does not exhibit a uniform pattern throughout its structure. There are usually varying textures within an object that make the segmentation algorithms assume that these features belong to different objects. It is also not uncommon to see a smooth gradient, i.e., uniform texture or color patterns along the boundary of adjacent objects. As a result, the segmentation algorithms may merge these adjacent patterns as a single object. These two scenarios are termed as under-merging and over-merging, respectively.

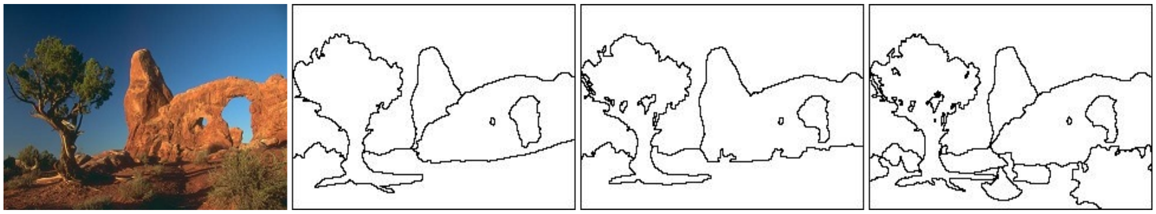

Figure 1.1: An image and its possible segmentation results. The number of segmented regions increases as we go from the left to the right.

Fig.1.1 presents the varying number of segments in the outputs, which are obtained by employing the same algorithm with different levels of threshold(s) of variations in the pixels' features. All these segmentation results are desirable. The segmentation results with a smaller number of regions are termed as coarse results. The segmentation results with a larger number of regions are termed as fine results. Certain applications that follow a coarse-to-fine hierarchy for their feature identification tasks [3] need coarser segments, while some require smaller segments [4]. The end goal of any segmentation method is, therefore, to improve the quality of these segments, irrespective of their coarseness. This quality enhancement is verified in terms of the improved boundary accuracy, and the overlapping accuracy of segments over corresponding ground truths, which contain segments manually labeled by human subjects. To visualize the segmentation results as an RGB image, we need to assign each labeled region a color corresponding to certain R, G, and B values. Fig. 1.2 shows a sample image and three ground truths labeled by different human subjects. The labeled regions were assigned by colors whose three channels are respectively the mean R, G, and B values of the corresponding regions in the input image.

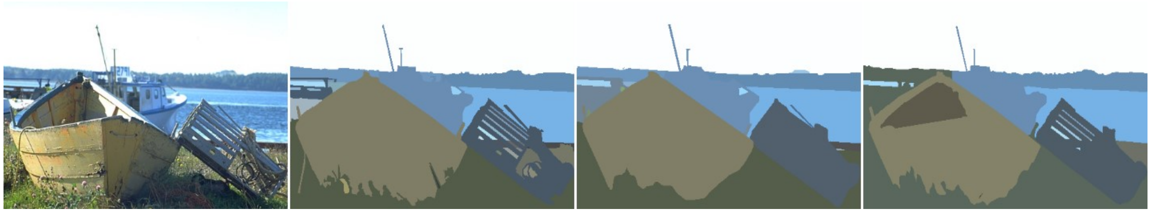

Figure 1.2: An image and its three ground-truths labeled by different human subjects.

2. Related Work

There are numerous approaches available today, which are used for solving image segmentation. Objects in an image can be segmented based on a variety of cues. Some algorithms look for homogeneity in intensity or texture. For example, Weeks et al. [5] tried segmentation in HIS color space and Sagiv et al. [6] tried segmentation using texture. Many algorithms have also attempted at finding homogeneity in multiple properties of an image. For example, Shi et al. [7] used smoothness and boundary continuity to find segments in an image.

Adams and Bischof in their breakthrough paper [8] took a different approach by assuming underlying statistical cues, in which they tried to find homogeneity. They introduced the concept of the seeded region growing (SRG) for intensity images, where the user controls the segmentation procedure by inputting seed pixels from which the regions grow. By carefully selecting the appropriate seeds and the statistics governing the grouping, it was possible to extract all objects in an image. The main drawback of this paper is the requirement of human intervention. Nock and Nielsen took a similar statistical approach in SRM [1], where they eliminated the need for any manual input of the seed pixels. In their proposed algorithm, virtually all pairs of pixels act as seeds. Each seed pair is ordered in an increasing order of their dissimilarity and merging is sequentially carried out on the ordered seed pairs. The segmentation is done hierarchically by increasing the scale value for every iteration. As the segmentation undergoes each iteration, it gradually produces bigger segments depending upon the merging criteria. The segments that do not merge remain smaller throughout all the iterations. Throughout these iterations, the order of the pixel pairs, depending upon the similarity between their respective segments, is not changed. As pointed in [9], SRM does not give importance to changing these orders in the middle of segmentation process when there is a change in the scale parameter, we bypassed this problem in our proposed approach. In other words, instead of making it a hierarchical segmentation, we ran the image through multiple segmentation runs, one run per scale value, to find a different number of segments.

While statistical approaches utilized the underlying distributions such as cues for region merging, texture based methods took the route of defining regions and pixels through feature representation. Usage of Gabor filter banks is a stable method in extracting features of pixels and their local neighborhoods. This is due to the known fact that the human visual cortex processes the images they see in a similar way as the Gabor functions [10]. Cheng et al. defined rotational invariant uniform local binary patterns (RIU-LBP) to define each pixel's texture. Their approach utilized the basic SRM algorithm, which continued to merge the regions based on the ordered seed pairs until the size of regions were over 20 pixels. Beyond this, a region merging had to satisfy another criterion to ensure the two merged regions are similar in terms of the RIU-LBP texture features. To this end, the Bhattacharya distance was utilized to compute the similarity between the regions.

One common end goal of texture-based segmentation methods is to merge or group pixels or regions using any minimization, optimization, or clustering algorithm [6] on the texture feature vector. Wang et al. suggested such a method that effectively utilizes texture features obtained from the Daubechies-4 wavelet transformation of the L-component of the input image transformed to the LUV color space[2].

3. Proposed Technique

The method we propose in this report is, de-texturing the input images prior to passing them on to any segmentation methods, specifically, the SRM or the SIMPLIcity segmentation methods. While many methods as we have discussed earlier and others in the vast available literature have strived for better representation of textures of pixels, the method offered by Mignotte in [11] simplifies the input images by denoising the interiors, while preserving the edges of regions [12]. In this report, we present a modified de-texturing method that improves upon both SRM and SIMPLIcity segmentation methods.

3.1. Brief Overview of the SRM segmentation method

Before introducing our improvement, let us have a brief look at the SRM segmentation algorithm. The algorithmic overview of SRM is summarized as follows:

1. All pixels of the image are grouped into pairs of 2, p and p’, where p is the pixel under consideration and p’ is any one of the border pixels based on 4-connectivity as shown in Figure 3.1.

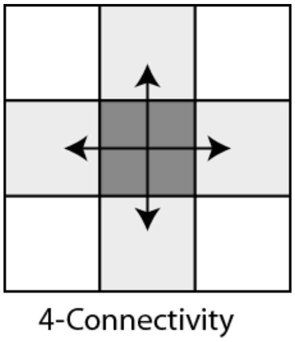

Figure 3.1: 4- connectivity of a pixel consists of its top, left, bottom, and right border pixels.

2. Calculate the similarity function f (p, p’) of all these pixel pairs. The choice of the similarity function ‘f’ here is an absolute difference in pixel intensities.

3. All pixel pairs are sorted in the ascending order of their similarity function value.
4. For each ascendingly ordered pair of pixels:
    - 4.1 If the pair has been selected, process the next pixel pair.
    - 4.2 Otherwise, another check is performed to see if the pixels in the pair belong to the same region. If they belong to the same region, process the next pixel pair.
    - 4.3 If both are untrue (i.e., the pair has not been selected and the pixels in the pair do not belong to the same region), then a merging predicate is evaluated to check if the pixels, or the regions to which they belong to, can be merged or not. The predicate P is given as

$$P(R,R') = \begin{cases} true, & if\ |\bar{R} - \overline{R'}| \leq \sqrt{b^2(R) + b^2(R')} \\ false, & otherwise \end{cases}$$

where R and R’ are regions to which the pixel pair, p and p’, belong to, and

$$b(R) = g\sqrt{\left(\frac{1}{2Q|R|}\right)\ln\left(\frac{min(|R|,g)^{(|R|+1)}}{\delta}\right)}$$

Where$\delta = \frac{1}{6|I|^2}$, |R| is the number of pixels in region R, $|I|$ is the number of pixels in the entire image, Q is the scale factor with the values of 1, 2, 4, 8, 16, 32, 54, 128, and 256, respectively.

    - 4.4 If the predicate is true, then the pixel pair, or the regions they belong to, are merged and this pixel pair is marked as selected.
5. Step 4 is repeated until the order has been exhausted (i.e., all the pairs have been selected).

3.2. Brief overview of SIMPLIcity segmentation

The SIMPLIcity paper [2] suggests a segmentation algorithm that intends to create as many segments as possible even if it means under-merging. They build this segmentation system for their image retrieval system as a way of creating as many meaningful portions as possible to understand the semantics of an image. We plugged in our de-texturing to this algorithm to improve the quality of these segments and avoid overlap of a segment over multiple objects.

The algorithmic view of SIMPLIcity segmentation is summarized as follows, where the input is an RGB image and a value *max-K* is passed for multiple k-means clustering results.

1. Apply padding to the image so that its rows and columns are a multiple of 4's.
2. Transform the image to the LUV color space.
3. Apply a one level Daubechies-4 wavelet transform to L component.
4. The decomposition produces 4 subbands, namely, A, H, V, and D representing average intensities, horizontal features, vertical features, and diagonal features, respectively, of the original image.
5. Divide each band into non-overlapping 2x2 blocks as shown in Figure 3.2. It should be noted that each 2x2 block corresponds to a 4x4 block in the original image as illustrated in Figure 3.3.

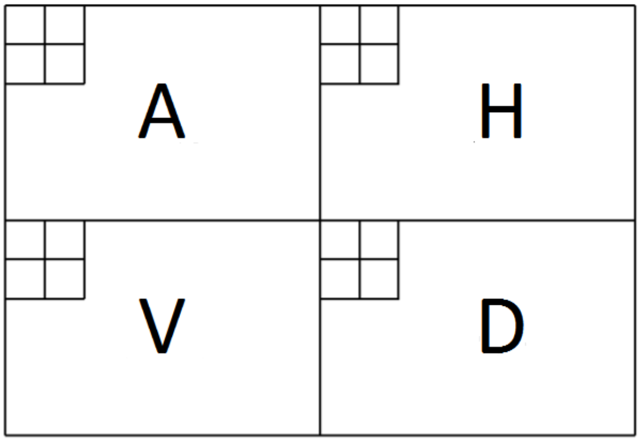

Figure 3.2: Each sub-band of the decomposed image is divided into non-overlapping 2x2 blocks. The size of each subband is one quarter of the size of the original image.

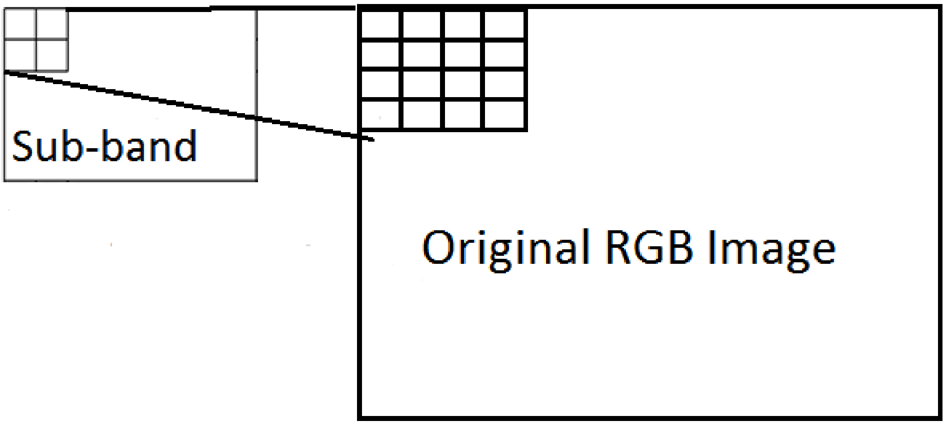

Figure 3.3: The corresponding 4x4 block section in the original image for every 2x2 block of the sub-band.

6. For each 2x2 block of H, V, and D sub-bands

   6.1 Compute its root-mean-square (RMS) values as rms(H), rms(V) and rms(D), respectively

   6.2 Compute the means of its corresponding 4x4 block of the 3 channels of the RGB image as mean(R), mean(G) and mean(B), respectively.

   6.3 Construct its 6-featured vector (rms(H), rms(V), rms(D), mean(R), mean(G),mean(B)).

7. For each integer value *k* ranging from 3 to *Max-k*

    7.1 Obtain clusters of 2x2 blocks with labels by clustering these feature vectors using the *k*-means clustering algorithm.

    7.2 Create unique labels for disconnected components of the same cluster.

    7.3 Apply the labels to the original image size by resizing the output matrix.

    7.4 Remove the padding.

    7.5 Obtain the segmented output corresponding to the *k* value

3.3. The Proposed Modified De-texturing Method

The algorithmic overview of the proposed modified de-texturing method is summarized as follows:

1. Get input image in the RGB color space.
2. Convert it to 4 other color spaces, which are HSV, YIQ, LUV, and LAB.
3. For each color space

    3.1 Quantize the gray levels ranging from 0 to 255 to five quantization levels of 1 to 5

    3.2 Quantize the combination of the three channels, with each channel having values ranging from 1 to 5, to a single channel with values ranging from 1 to 125 by multiplying the corresponding quantized values at each pixel location in each of the three channels.

    3.3 From this single channel matrix, construct a 125 bin histogram for a 5x5 window around each pixel. This histogram will act as the pixel's feature vector.

4. For every pixel, concatenate the feature vectors obtained from the above step for all color spaces to create a feature descriptor of size 625 (e.g., 125*5).

5. Apply principal component analysis (PCA) on the descriptors of all pixels to reduce the dimensionality of the feature vectors.
6. Use K-means on the dimensionally reduced features to cluster the descriptors of all the pixels, for *k* = 8.
7. Create a labeled matrix based on the obtained clusters.
8. Convert this labeled matrix into an RGB image by replacing each cluster's label with the average R, G, and B values of the corresponding cluster in the original input image.

Thus, we have a de-textured input image, which is transformed to the YIQ color space before applying the SRM algorithm on it. In our proposed system, we used five color spaces, namely, RGB, HSV, YIQ, LUV and LAB, to extract features. These five color spaces provide complementary information. Specifically, HSV, standing for hue, saturation and value, is used as it decouples chromatic information from shadow information. In YIQ color space, *Y* encodes luminance information and *I* and *Q* encode chrominance information. The LAB color space or the CIE L a*b* color space as it is officially called, is used as it approximates human vision, especially in human perception of lightness or darkness of an image. LUV color space approximates human vision in its perception of color changes. Finally, RGB color space is used for its similarity to human perception of complex colors. Although M. Mignotte in [11] suggests to use XYZ color space in addition to the above color spaces, we decided to drop it after our experiments showed an improvement in segmentation results upon removing XYZ images in getting the de-textured images. In our proposed de-texturing method, we applied PCA to reduce the dimensionality of the feature vector, which captures the few largest variance (i.e., the most distinguishable features) totaling to at least 80% of the variance, to facilitate the clustering process.

Figure 3.4 presents the block diagram of the proposed detextured-based SRM method. The same de-textured image is transformed to the LUV color space before applying the SIMPLIcity's segmentation algorithm on it. Figure 3.5 presents the block diagram of the proposed de-textured-based SIMPLIcity method.

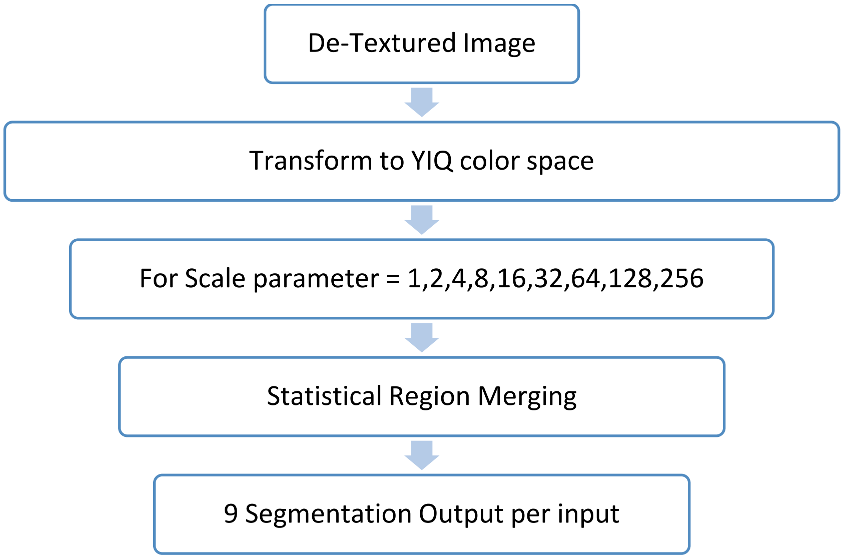
De-Textured Image
Transform to YIQ color space
For Scale parameter = 1,2,4,8,16,32,64,128,256
Statistical Region Merging
9 Segmentation Output per input

Figure 3.4: De-Textured-YIQ-SRM (Statistical Region Merging).

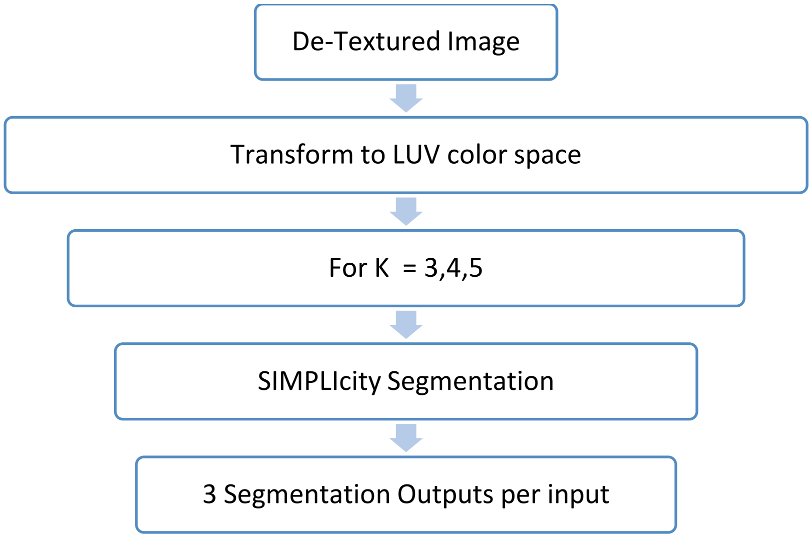
De-Textured Image
Transform to LUV color space
For K = 3,4,5
SIMPLIcity Segmentation
3 Segmentation Outputs per input

Figure 3.5: De-Textured-LUV-SIMPLIcity Segmentation.

4. Experimental Results

4.1. Dataset and Metrics

Berkeley Segmentation Dataset has been widely used as a benchmark for measuring the effectiveness of a segmentation algorithm. All benchmarks are computed under two scenarios:

a. Optimal Dataset Scale (ODS) – Since we get multiple segmentation outputs, by varying the values of the scale parameter Q in SRM or by varying the values of *k* in SIMPLIcity, we fix one segmentation scale for the whole dataset that has overall best segmentation results and compute the benchmark on these results.

b. Optimal Individual Scale (OIS) – We select the segmentation that gives the best results for each image, and calculate an aggregate benchmark for these outputs.

There are two different ways of measuring the effectiveness – boundary benchmarks and region benchmarks. Boundary benchmarks measure the strength of region contour with respect to the contours of the ground truth. Measures used are:

a. Precision: It is the probability that a contour pixel is a true boundary pixel. It is computed as

   Precision $= \frac{true\ positives}{true\ positives\ + false\ positives}$, where true positives are pixels accurately labeled as boundary pixels and false positives are pixels inaccurately labeled as boundary pixels.

b. Recall: It is the probability of the detection of boundary pixels. It is computed as,

   Recall $= \frac{true\ positives}{true\ positives\ + false\ negatives}$, where false negatives are boundary pixels inaccurately labeled as non-boundary pixels.

c. F-measure: It is a single measure to incorporate both recall and precision. The higher the F-measure, the greater the balance between precision and recall. It is computed as

$$F = \frac{2*Precision*Recall}{Precision+Recall}$$

Region benchmarks measure the strength of the regions using the following measures:

a. Segmentation Covering (SC) – It measures the overlap of regions of the segmentation output and the regions of the ground truth. The higher the SC value, the better the quality of segmentation.

b. Variation of Information (VI) – It measures the distance between output's segmentation and ground truth's segmentation. The lower the VI value, the better the segmentation results.

c. Probability Rand Index (PRI) – It measures the accuracy of labels assigned to pairs of pixels in a cluster. A higher PRI value indicates more accurate segmentation.

4.2. Experimental Setup

The de-texturing code and SRM and SIMPLIcity segmentation methods were written in separate MATLAB files. The datasets BSDS300 and BSDS500 [13] have found wide acceptance as benchmarks for evaluating different segmentation algorithms. BSDS300 contains 200 training images and 100 testing images. BSDS500 uses BSDS300 as the training dataset and adds 200 new testing images. All the testing images encompass a wide variety of geographic features, objects, plants and animals.

As our proposed technique is an unsupervised method that does not require any prior training, we directly ran it on the testing images. For SRM, 9 different scales were used for each image, instead of iteratively changing the scale. For SIMPLIcity segmentation, we varied the values of *k* from 3 to 5. As a result, for each input image, we get 9 different segmentation results from the De-texturing-YIQ-SRM process and 3 different segmentation results from the De-texturing-LUV-SIMPLIcity process.

These outputs are stored as label matrices. These label matrices are then passed to the files provided by the Berkeley database to compute the corresponding measures for the boundary benchmarks and the region benchmarks.

4.3. Results

Figure 4.1 shows a sample image, its ground truth images, and the segmentation results of De-texturing-YIQ-SRM and De-texturing-LUV-SIMPLIcity.

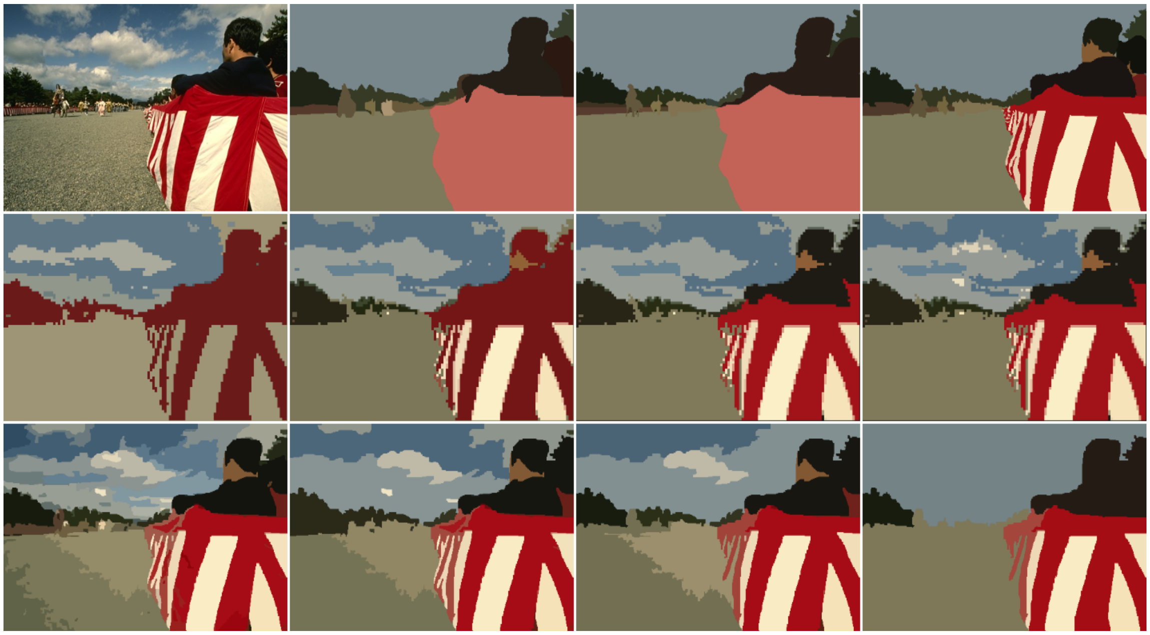

Figure 4.1: Comparison of the ground truth segmentation results and the segmentation results obtained from the two proposed segmentation methods. The top row shows the original image and 3 ground truth segmentation results manually labeled by human subjects. The second row shows De-texturing-LUV-SIMPLIcity segmentation results for four values of k, i.e., 2, 3, 4, and 5. The third row shows De-Texturing-YIQ-SRM segmentation results for four scale values of Q, i.e., 1, 4, 5, and 7.

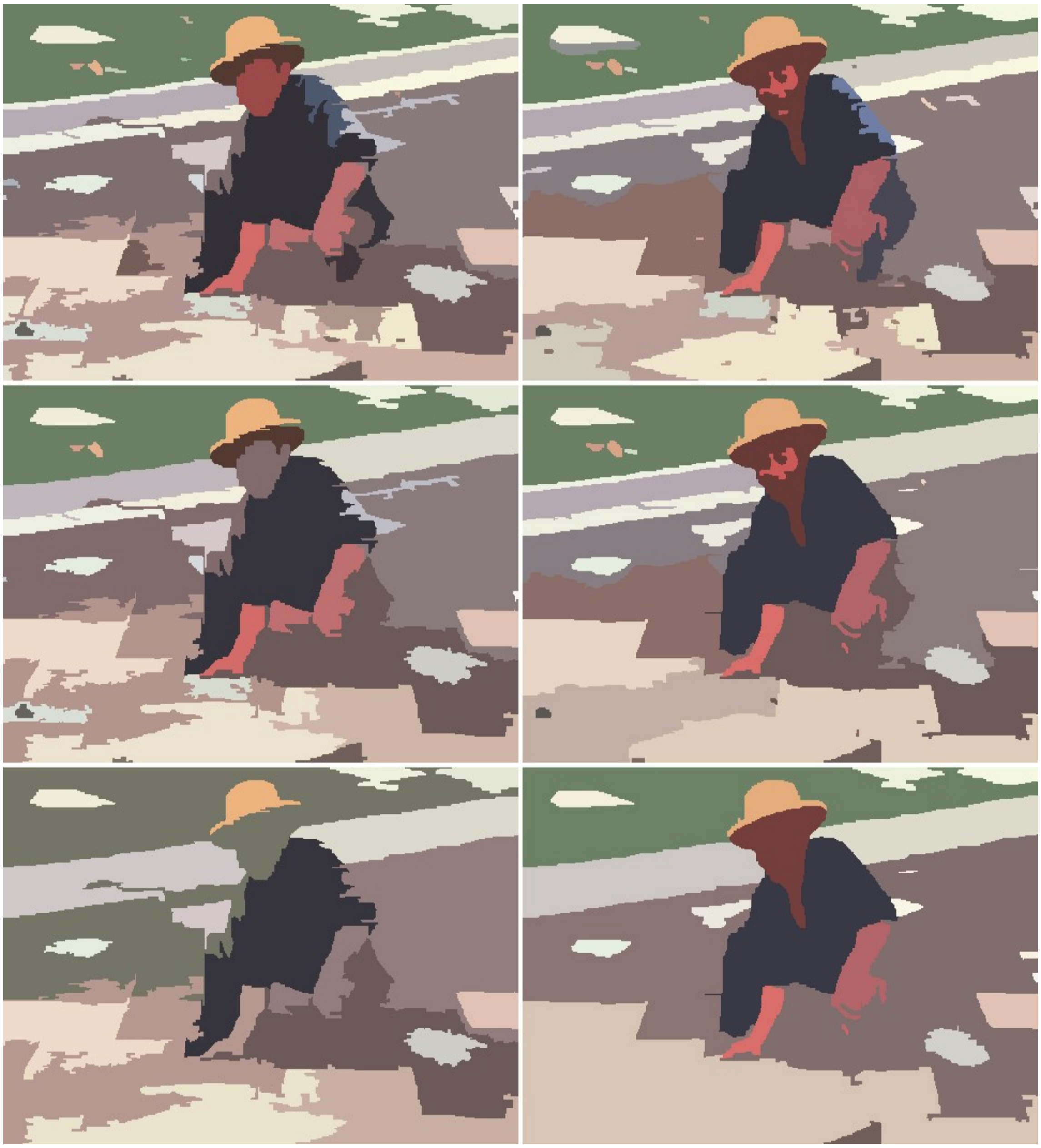

Figure 4.2: Comparison of the outputs of the SRM-RGB segmentation algorithm (shown on the left-hand side) and the outputs of the De-Texturing-YIQ-SRM segmentation algorithm (shown on the right-hand side) for different values of Q's, where the top row presents the results for Q= 4, the middle row presents the results for Q = 5, and the bottom row presents the results for Q=6.

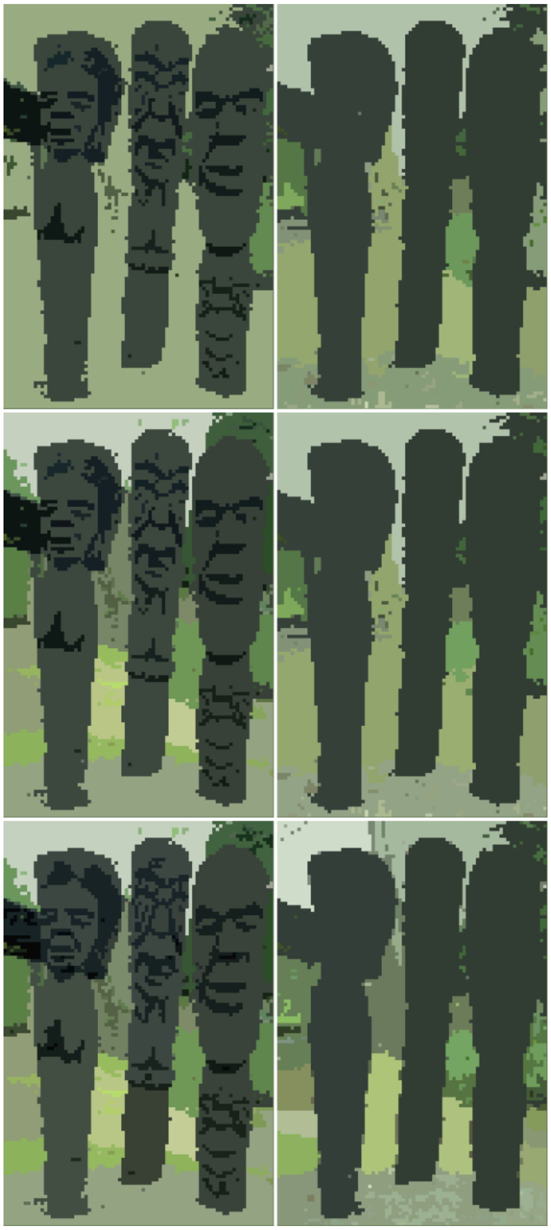

Figure 4.3: The comparison of the outputs of the SIMPLIcity -RGB segmentation algorithm (shown on the left-hand side) and the outputs of the De-Texturing-LUV- SIMPLIcity segmentation algorithm (shown on the right-hand side) for different values of *k*'s, where the top row presents the results for *k* = 3, the middle row presents the results for *k*= 4, and the bottom row presents the results for *k*= 5.

Figure 4.2 shows a visual comparison of SRM segmentation results against De-texturing-YIQ-SRM segmentation results for another picture. Figure 4.3 shows a visual comparison of SIMPLIcity segmentation results against De-texturing-LUV- SIMPLIcity segmentation results for a different picture. Table 4.1 lists the values of the three boundary benchmark measures computed from the segmentation results obtained by applying different segmentation methods on the databases

BSD300 and BSD500. The first row in the table lists the results for the best segmentation method and the last row in the table lists the results for the worst segmentation method. It clearly shows that the proposed detexturing-YIQ-SRM approach respectively improves the YIQ-SRM by 0%, 1.6%, and 22% on BSDS300 database for the measures of ODS, OIS, and AP, and respectively improves the YIQ-SRM by 0%, 0%, and 26% on BSDS500 database for the measures of ODS, OIS, and AP. It also respectively improves the original RGB-SRM by 0%, 1.6%, and 29.4% on BSDS300 for the measures of ODS, OIS, and AP and respectively improves the original RGB-SRM by 1.6%, 0%, and 33.33% on BSDS500 for the measures of ODS, OIS, and AP. Similarly, the proposed detexturing-LUV-SIMPLIcity respectively improves the original LUV-SIMPLIcity by 3.8%, 1.8%, and 0% on BSDS300 database for the measures of ODS, OIS, and AP and respectively improves the original LUV-SIMPLIcity by 1.8%, 1.8%, and 16.7% on BSDS500 database for the measures of ODS, OIS, and AP.

Table 4.1: Comparison of segmentation results of different segmentation methods in terms of boundary benchmarks on datasets BSDS300 and BSDS500.

	BSDS300			BSDS500		
Algorithm	ODS	OIS	AP	ODS	OIS	AP
Human	0.79	0.79	-	0.8	0.8	NA
gPb-owt-ucm	0.71	0.74	0.73	0.73	0.75	0.73
gPb	0.7	0.72	-	0.71	0.74	0.65
Mean Shift	0.63	0.66	0.54	0.64	0.68	0.56
Ncuts	0.62	0.66	0.43	0.64	0.68	0.45
Detexturing-YIQ-SRM	0.6	0.64	0.44	0.63	0.65	0.48
YIQ-SRM	0.6	0.63	0.36	0.63	0.66	0.38
RGB-SRM	0.6	0.63	0.34	0.62	0.65	0.36
Canny-owt-ucm	0.58	0.63	0.58	0.6	0.64	0.58
Canny	0.58	0.62	0.58	0.58	0.63	0.58
Felz-Hutt	0.58	0.59	0.53	0.61	0.64	0.56
SWA	0.56	0.59	0.54	NA	NA	NA
Detexturing-LUV-Simplicity	0.54	0.57	0.11	0.55	0.58	0.14
LUV-Simplicity	0.52	0.56	0.11	0.54	0.57	0.12

Table 4.2 and Table 4.3 list the 3 region benchmarks, SC, PRI and VI when applying different segmentation methods on BSDS300 and BSDS500 databases, respectively. In measuring the three values, they are averaged over each segmentation in the output with respect to the ground truth. For PRI, there are segments that arguably present with the best possible accuracy, i.e., a value close to 1. And similarly for VI, few segments might present the least possible differences in clustering, i.e., a value close to 0. Since the best PRI or the best VI do not reflect the actual performance of the algorithm, they are not included in the comparison tables as suggested in the literature as well.

Table 4.2 lists region benchmark measures computed from the segmentation results obtained by applying different segmentation methods on the databases BSDS300. The first row in the table lists the results for the best segmentation method and the last row in the table lists the results for the worst segmentation method. It clearly shows that the proposed detexturing-YIQ-SRM approach respectively improves the YIQ-SRM by 2%, 3.5%, and 3% for the three measures of SC (ODS, OIS, and the best SC, respectively), by 0% and 0% for the two measures of PRI (ODS and OIS, respectively), and by 2% and 4% for the two measures of VI (ODS and OIS, respectively). It also improves the original RGB-SRM by 4%, 5%, and 6% for the three measures of SC (ODS, OIS, and the best SC, respectively), by 0% and 1.2% for the two measures of PRI (ODS and OIS, respectively), and by 7% and 8% for the two measures of VI (ODS and OIS, respectively). Similarly, the proposed detexturing-LUV-SIMPLIcity respectively improves the original LUV-SIMPLIcity by 4%, 4%, and 0% for the three measures of SC (ODS, OIS, and the best SC, respectively), by 0% and 0% for the two measures of PRI (ODS and OIS, respectively), and by 2% and 5% for the two measures of VI (ODS and OIS, respectively).

Table 4.2: Region benchmarks for the dataset BSDS300.

Algorithm	BSDS300						
	SC			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
Human	0.73	0.73	NA	0.87	0.87	1.16	1.16
gPb-owt-ucm	0.59	0.65	0.75	0.81	0.85	1.65	1.47
Mean Shift	0.54	0.58	0.66	0.78	0.80	1.83	1.63
Detexturing-YIQ-SRM	0.52	0.59	0.68	0.75	0.82	1.89	1.68
Felz-Hutt	0.51	0.58	0.68	0.77	0.82	2.15	1.79
YIQ-SRM	0.51	0.57	0.66	0.76	0.82	1.93	1.75
RGB-SRM	0.50	0.56	0.64	0.75	0.81	2.02	1.82
Detexturing-LUV-Simplicity	0.50	0.54	0.55	0.74	0.76	2.26	2.07
Canny-owt-ucm	0.48	0.56	0.66	0.77	0.82	2.11	1.81
SWA	0.47	0.55	0.66	0.75	0.80	2.06	1.75
Ncuts	0.44	0.53	0.66	0.75	0.79	2.18	1.84
LUV-Simplicity	0.48	0.52	0.55	0.74	0.76	2.30	2.17
Total Var.	0.57	NA	NA	0.78	NA	1.81	NA
T+B Encode	0.54	NA	NA	0.78	NA	1.86	NA
Av. Diss.	0.47	NA	NA	0.76	NA	2.62	NA
Chan Vese	0.49	NA	NA	0.75	NA	2.54	NA

Table 4.3 lists region benchmark measures computed from the segmentation results obtained by applying different segmentation methods on the databases BSDS500. The first row in the table lists the results for the best segmentation method and the last row in the table lists the results for the worst segmentation method. It clearly shows that the proposed detexturing-YIQ-SRM approach respectively improves the YIQ-SRM by 4%, 3.5%, and 4.5% for the three measures of SC (ODS, OIS, and the best SC, respectively), by 4% and 1% for the two measures of PRI (ODS and OIS, respectively), and by 1% and 4% for the two measures of VI (ODS and OIS, respectively). It also improves the original RGB-SRM by 10%, 9%, and 10% for the three measures of SC (ODS, OIS, and the best SC, respectively), by 2% and 3% for the two measures of PRI (ODS and OIS, respectively), and by 8% and 11% for the two measures of VI (ODS and OIS, respectively). Similarly, the proposed detexturing-LUV-SIMPLIcity respectively improves the original LUV-SIMPLIcity by 2%

and 3.4% for the two measures of VI (ODS and OIS, respectively). In dataset BSDS500 under region benchmarks, we can see that the proposed detexturing-LUV-SIMPLIcity respectively improves the original LUV-SIMPLIcity predominantly in the VI measures compared to SC or PRI. This could be attributed to the fact the *L* component of the LUV color space, which the input image is transformed into, is not really dependent on textures for accurate segmentation. But it also indicates that the de-texturing definitely improves upon the placement of the segments with respect to the ground truth, which can be observed in the reduction of the VI (ODS and OIS) measure.

Table 4.3: Region benchmarks for the dataset BSDS500.

	BSDS500						
Algorithm	SC			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
Human	0.72	0.72	NA	0.88	0.88	1.17	1.17
gPb-owt-ucm	0.59	0.65	0.74	0.83	0.86	1.69	1.48
Mean Shift	0.54	0.58	0.66	0.79	0.81	1.85	1.64
Detexturing-YIQ-SRM	0.53	0.59	0.69	0.79	0.83	1.91	1.69
Felz-Hutt	0.52	0.57	0.69	0.8	0.82	2.21	1.87
YIQ-SRM	0.51	0.57	0.66	0.76	0.82	1.93	1.75
Detexturing-LUV-Simplicity	0.5	0.53	0.56	0.77	0.78	2.32	2.07
LUV-Simplicity	0.5	0.53	0.56	0.77	0.78	2.36	2.14
Canny-owt-ucm	0.49	0.55	0.66	0.79	0.83	2.19	1.89
RGB-SRM	0.48	0.54	0.63	0.77	0.81	2.08	1.88
Ncuts	0.45	0.53	0.67	0.78	0.8	2.23	1.89

Figure 4.4 shows precision-recall curves of the proposed De-Texturing-YIQ-SRM method and the original RGB-SRM method for the datasets BSD300 and BSD500, respectively. The green background curves represent the graph for fixed F-measure values. The green dot is the F-measured derived from segmentations made by multiple human subjects. It clearly shows the proposed Detexturing-YIQ-SRM achieves better segmentation results than the original RGB-SRM

on both databases since its precision-recall curves cover larger areas than the precision-recall curve of the original RGB-SRM. Figure 4.5 shows precision-recall curves of De-Texturing-LUV-SIMPLIcity and LUV-SIMPLIcity for the datasets BSD300 and BSD500, respectively. It clearly shows the proposed Detexturing-LUV-SIMPLIcity achieves better segmentation results than the original LUV-SIMPLIcity on both databases since its precision-recall curves cover larger areas than the precision-recall curves of the original LUV-SIMPLIcity. However, the area under both the curves in both the cases is small because we only considered *k*-values of 3 to 5. Any higher *k* values will result in finer segments.

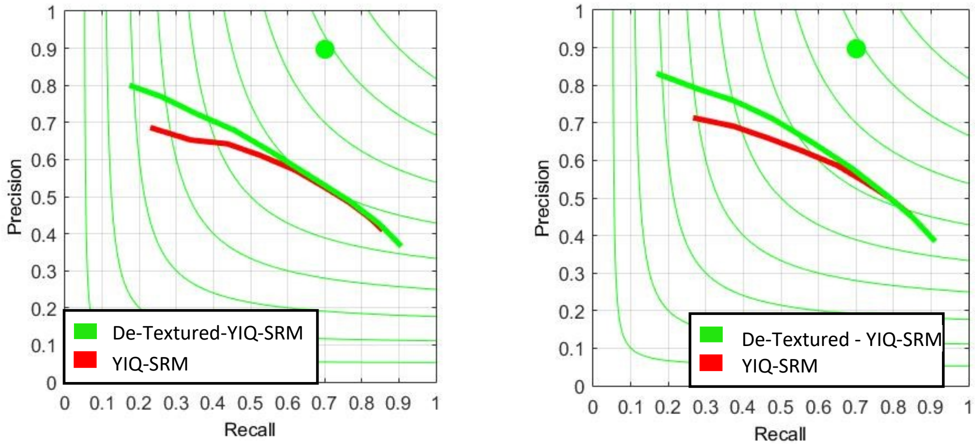

Figure 4.4: Comparison of precision and recall curves for the proposed De-Textured-YIQ-SRM method and the RGB-SRM method for datasets BSD300 and BSD500, respectively.

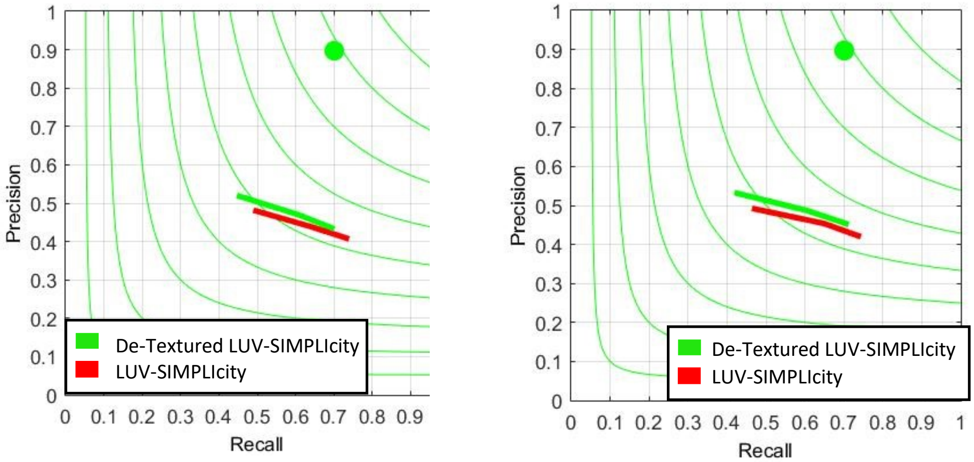

Figure 4.5: Comparison of precision and recall curves for the proposed De-Textured-LUV-SIMPLIcity method and the LUV-SIMPLIcity method for datasets BSD300 and BSD500, respectively.

5. Conclusion

While SRM and Simplicity-segmentation are capable standalone algorithms, they still have a lot more room to improve. SRM specifically does not make use of other color spaces. In our method, we show that SRM performs its best while using the YIQ color space of the image as input. Irrespective of the color space, the algorithms produce better results when applying the detexturing algorithm on the inputs to obtain the pre-processed inputs. De-texturing is initially introduced as a simple denoising technique. In our study, we employed the detexturing algorithm as the pre-processing step to obtain the pre-processed inputs for the segmentation algorithms. Our extensive experimental results on the Berkeley Segmentation Dataset clearly show that the segmentation results obtained with detexturing are better than the segmentation results obtained without detexturing in term of Segmentation Covering, Variation of Information, Probability Rand Index, and F-Measures under any of the three applicable conditions such as ODS, OIS, and best, when employing either SRM or Simplicity-segmentation as the segmentation method.

References

[1] R. Nock and F. Nielsen, “Statistical region merging,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1452–1458, Nov. 2004.

[2] J. Z. Wang, Jia Li, and G. Wiederhold, “SIMPLIcity: semantics-sensitive integrated matching for picture libraries,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 947–963, 2001.

[3] H. Lu, G. Fang, X. Shao, and X. Li, “Segmenting human from photo images based on a coarse-to-fine scheme,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 3, pp. 889–899, Jun. 2012.

[4] C. L. Zitnick and S. B. Kang, “Stereo for image-based rendering using image over-segmentation,” *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 49–65, Jul. 2007.

[5] A. R. Weeks and G. E. Hague, “&lt;title&gt;Color segmentation in the HSI color space using the K-means algorithm&lt;/title&gt;,” 1997, pp. 143–154.

[6] Chen Sagiv, N. A. Sochen, and Y. Y. Zeevi, “Integrated active contours for texture segmentation,” *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1633–1646, Jun. 2006.

[7] J. Shi, S. Belongie, T. Leung, and J. Malik, “Image and video segmentation: the normalized cut framework,” in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, vol. 1, pp. 943–947.

[8] R. Adams and L. Bischof, “Seeded region growing,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 6, pp. 641–647, Jun. 1994.

[9] P. Soille, “Constrained connectivity for hierarchical image partitioning and simplification,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1132–1145, Jul. 2008.

[10] J. G. Daugman, “Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters,” *J. Opt. Soc. Am. A*, vol. 2, no. 7, 1985.

[11] M. Mignotte, “Segmentation by Fusion of Histogram-Based $K$-Means Clusters in Different Color Spaces,” *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 780–787, May

2008.

[12] M. Mignotte, “A de-texturing and spatially constrained K-means approach for image segmentation,” 2011.

[13] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detection and Hierarchical Image Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.